Applied Computer Vision (CS-696)
Homework Assignment 3

Name – Dhaval Harish Sharma **Red ID – 824654344**

Case 1: Template matching with fixed scale: Match the cropped template to the query image.
1) Implement the sliding window method for matching a template image to a query image.
2) Try to use three different metrics Zero-mean correlation, SSD and NCC.
3) Calculate the localization errors for each metric.

Solution:
Template Matching:
• Template matching is an algorithm for finding the location of given template image in the original Image. There are several methods of measuring the similarity that can be implemented for template matching to find exact location of the template image in the original image.

• Some of these techniques are as follows:
1. Correlation.
2. Zero mean correlation.
3. Sum Square Difference.
4. Normalized Cross Correlation.

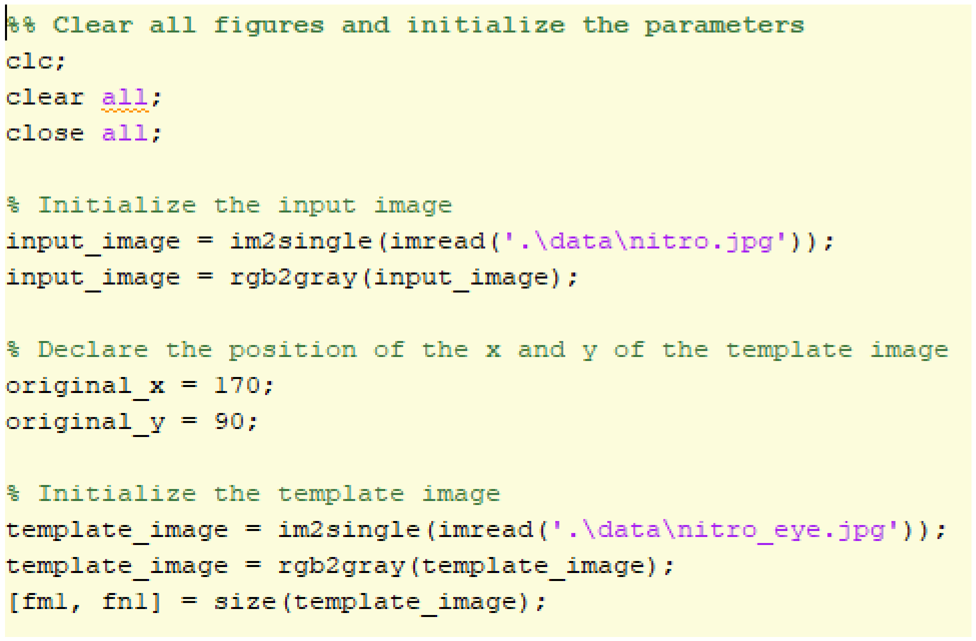
• After finding template in the original image, Euclidean distance between the original center of template and found template is calculated.

• It gives localization error in this method.

Algorithm:
Firstly, we clear all the figures. Then, we initialize the input image and x, y positions of the template image. Also, the template image is declared. We then process the images using the measures of similarity explained before. The first one is the Zero Mean Correlation. In case of Zero Mean Correlation, we first pad the template image in order to make the size even. Then, we find the mean of the total filter. We convolve the figure with the filter using ZMC and return the output image. We find the error corresponding to the output image. The resultant image is then output to the user along with the threshold image. The next measure is Sum of Squared Differences. The same process as before is followed in case of Sum of Squared Differences but with a changed measure. Inside the sum_of_sq_diff() function, instead of finding the mean, we pad the query image. To find the Normalized Cross Correlation, we used the in-built function called “normxcorr2”. It takes the input image and template image as its input and returns the output image using Normalized Cross Correlation measure. We then, print the corresponding errors in the output.

The corresponding codes and the output images for the above case are as follows:

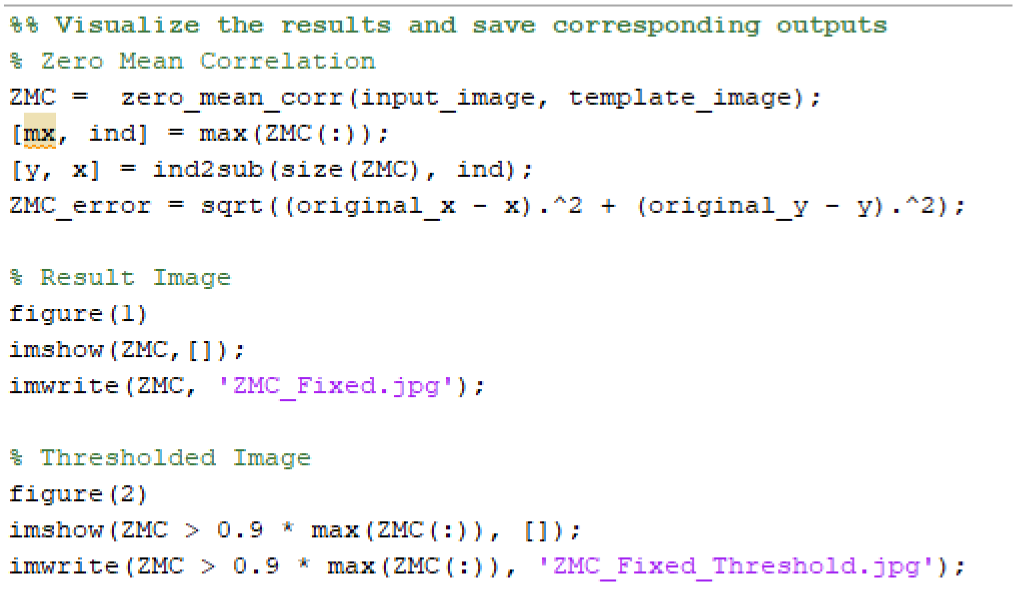
The initialization of the images are as follows:



```
%% Clear all figures and initialize the parameters
clc;
clear all;
close all;

% Initialize the input image
input_image = im2single(imread('.\data\nitro.jpg'));
input_image = rgb2gray(input_image);

% Declare the position of the x and y of the template image
original_x = 170;
original_y = 90;

% Initialize the template image
template_image = im2single(imread('.\data\nitro_eye.jpg'));
template_image = rgb2gray(template_image);
[fml, fnl] = size(template_image);
```

We visualize the results and save the corresponding outputs as follows:

1) Zero Mean Correlation



```
%% Visualize the results and save corresponding outputs
% Zero Mean Correlation
ZMC =  zero_mean_corr(input_image, template_image);
[mx, ind] = max(ZMC(:));
[y, x] = ind2sub(size(ZMC), ind);
ZMC_error = sqrt((original_x - x).^2 + (original_y - y).^2);

% Result Image
figure(1)
imshow(ZMC,[]);
imwrite(ZMC, 'ZMC_Fixed.jpg');

% Thresholded Image
figure(2)
imshow(ZMC > 0.9 * max(ZMC(:)), []);
imwrite(ZMC > 0.9 * max(ZMC(:)), 'ZMC_Fixed_Threshold.jpg');
```

2) Sum of Squared Difference

```
% Sum of Squared Difference
SSD = sum_of_sq_diff(input_image,template_image);
[mx, ind] = max(SSD(:));
[y, x] = ind2sub(size(SSD), ind);
SSD_error = sqrt((original_x - x).^2 + (original_y - y).^2);

% Result Image
figure(3);
imshow(SSD,[]);
imwrite(SSD, 'SSD_Fixed.jpg');

% Thresholded Image
figure(4);
imshow(SSD > 0.9 * max(SSD(:)), []);
imwrite(SSD > 0.9 * max(SSD(:)), 'SSD_Fixed_Threshold.jpg');
```

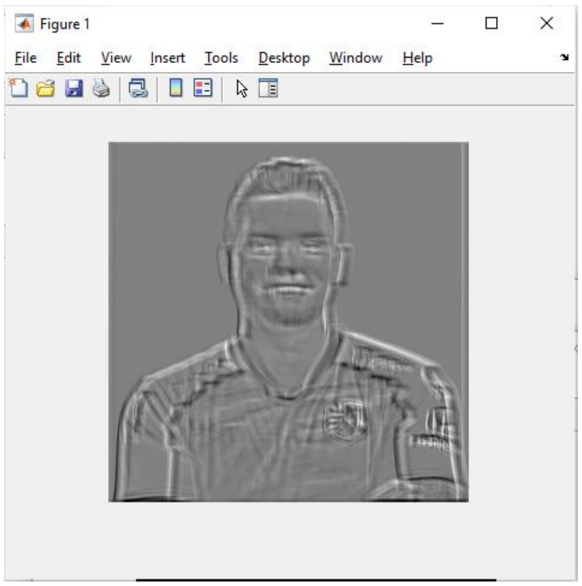
3) Normalized Cross Correlation

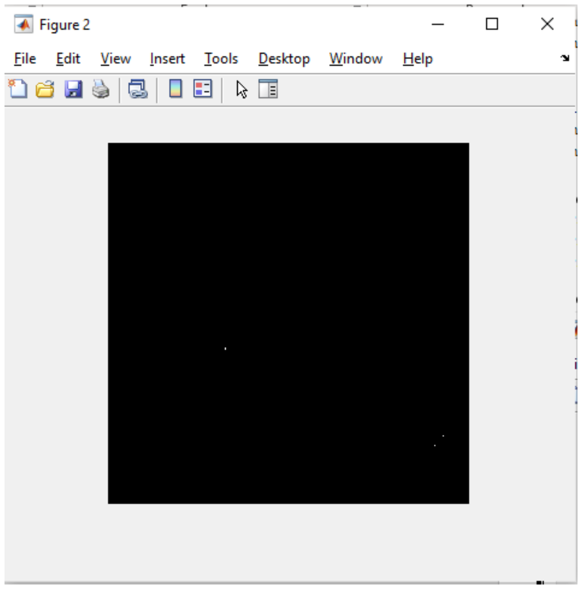
```
% Normalized Cross Correlation
NCC = normxcorr2(template_image,input_image);

[mx, ind] = max(NCC(:));
[y, x] = ind2sub(size(NCC), ind);
NC_error = sqrt((original_x - x).^2 + (original_y - y).^2);

% Result Image
figure(5)
imshow(NCC,[]);
imwrite(NCC, 'NCC_Fixed.jpg');

% Thresholded Image
figure(6)
imshow(NCC > 0.9 * max(NCC(:)), []);
imwrite(NCC > 0.9 * max(NCC(:)), 'NCC_Fixed_Threshold.jpg');
```

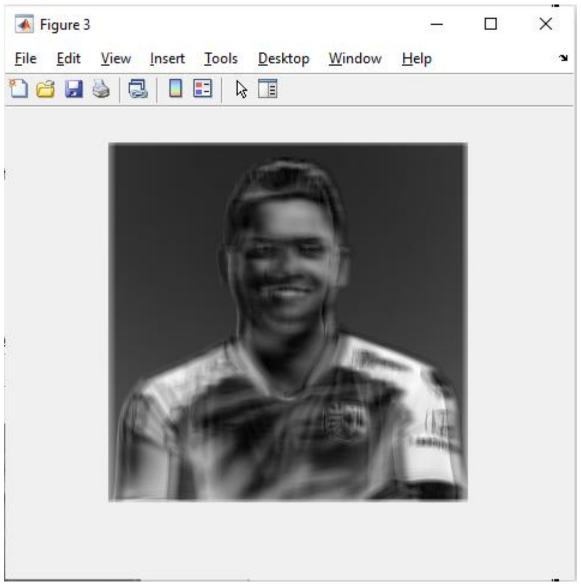
The output in the command window is as follows:

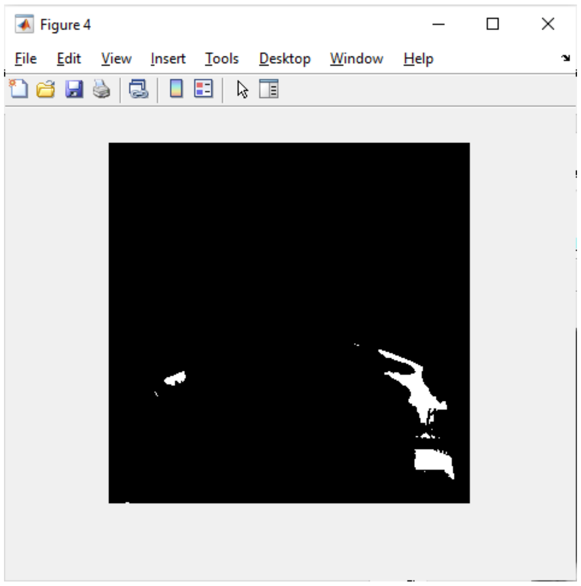
```
Error from ZMC - 108.374351
Error from SSD - 134.536240
Error from NC  - 32.449961
```

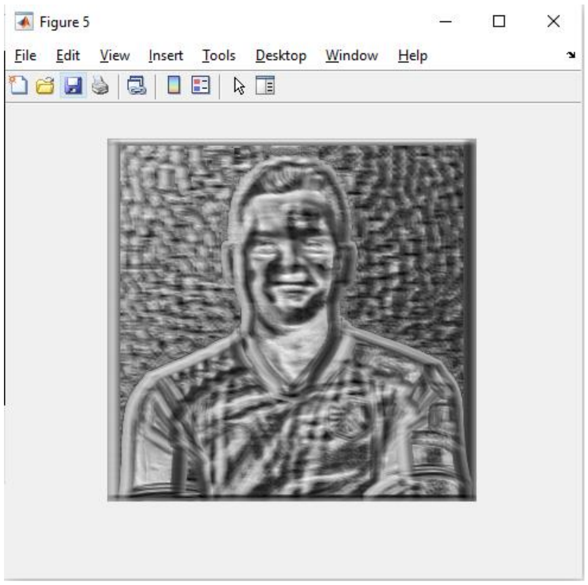
The output images are as follows:

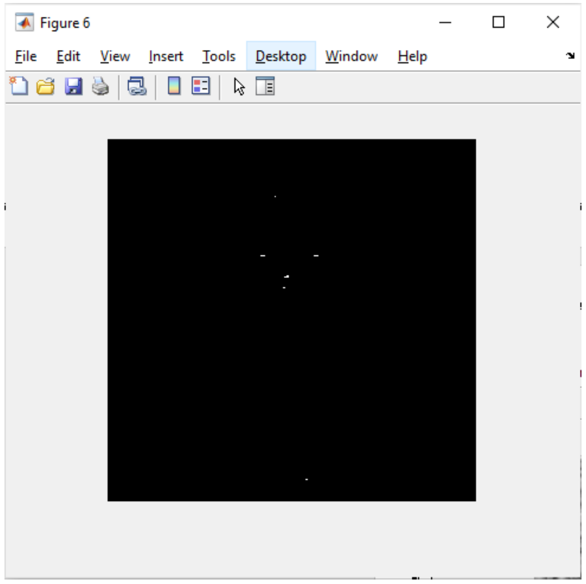












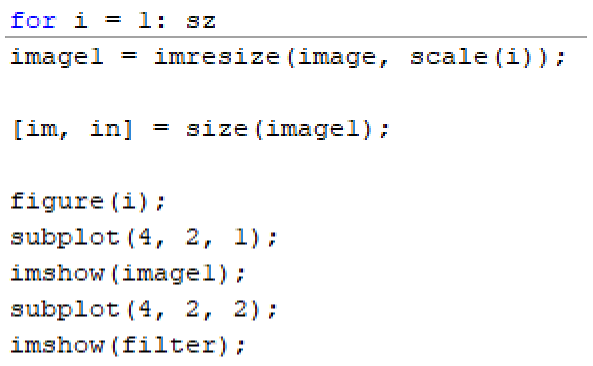
Case 2: Template matching with pyramid representation.
1) Resize the template images with multiple scaling factors (e.g., 0.5, 1.0, 2.0) to get the pyramid representation of the template image.
2) Implement the sliding window method for matching each of the pyramid images (scaled image) to the query image.
3) Use the metrics Zero-mean correlation, SSD and NCC.
4) Calculate the localization errors for each metric.

Solution:
Algorithm:
This case works exactly same as the previous algorithm. The only changes are that, we initialize the scale factors along with the other declarations. Then, we loop over various scale factors, rescaling the template images and calculating the output images.

The corresponding codes and the output images for the above case are as follows:

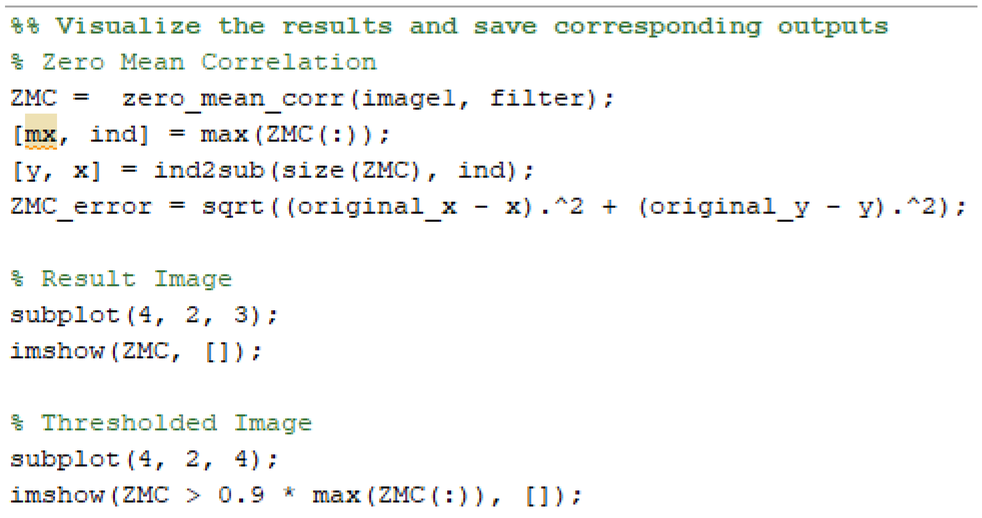
The initialization of the parameters are as follows:
```
%% Clear all figures and initialize the parameters
clc;
clear all;
close all;

% Initialize the scale factors
scale = [2 1 0.5];
sz = size(scale, 2);

% Initialize the input image
image = im2single(imread('.\data\nitro.jpg'));
image = rgb2gray(image);

% Declare the position of the x and y of the template image
original_x = 170;
original_y = 90;

% Initialize the template image
template_image = im2single(imread('.\data\nitro_eye.jpg'));
template_image = rgb2gray(template_image);
[fm1, fn1] = size(template_image);

% Initializing the filter
filter = imresize(template_image, 0.5);
[fm, fn] = size(filter);
```

We rescale the images:



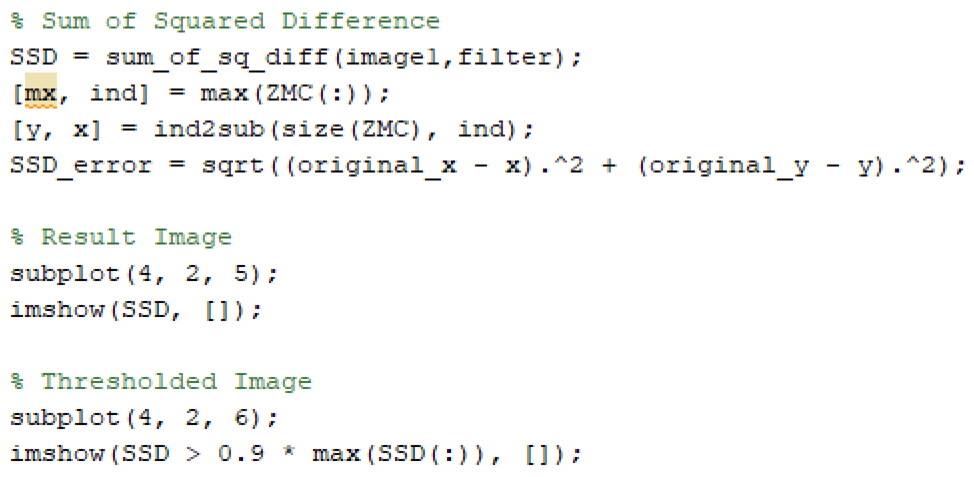
```
for i = 1: sz
image1 = imresize(image, scale(i));

[im, in] = size(image1);

figure(i);
subplot(4, 2, 1);
imshow(image1);
subplot(4, 2, 2);
imshow(filter);
```

1) Zero Mean Correlation:



```
%% Visualize the results and save corresponding outputs
% Zero Mean Correlation
ZMC =  zero_mean_corr(image1, filter);
[mx, ind] = max(ZMC(:));
[y, x] = ind2sub(size(ZMC), ind);
ZMC_error = sqrt((original_x - x).^2 + (original_y - y).^2);

% Result Image
subplot(4, 2, 3);
imshow(ZMC, []);

% Thresholded Image
subplot(4, 2, 4);
imshow(ZMC > 0.9 * max(ZMC(:)), []);
```

2) Sum of Squared Difference:



```
% Sum of Squared Difference
SSD = sum_of_sq_diff(image1,filter);
[mx, ind] = max(ZMC(:));
[y, x] = ind2sub(size(ZMC), ind);
SSD_error = sqrt((original_x - x).^2 + (original_y - y).^2);

% Result Image
subplot(4, 2, 5);
imshow(SSD, []);

% Thresholded Image
subplot(4, 2, 6);
imshow(SSD > 0.9 * max(SSD(:)), []);
```

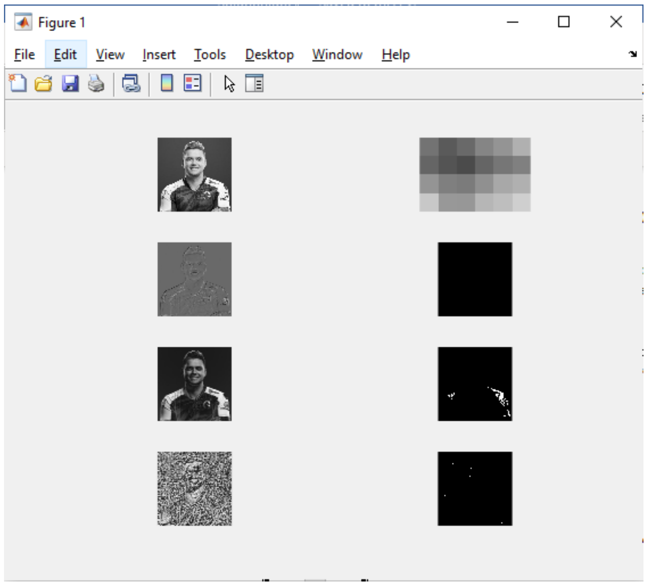
3) Normalized Cross Correlation:

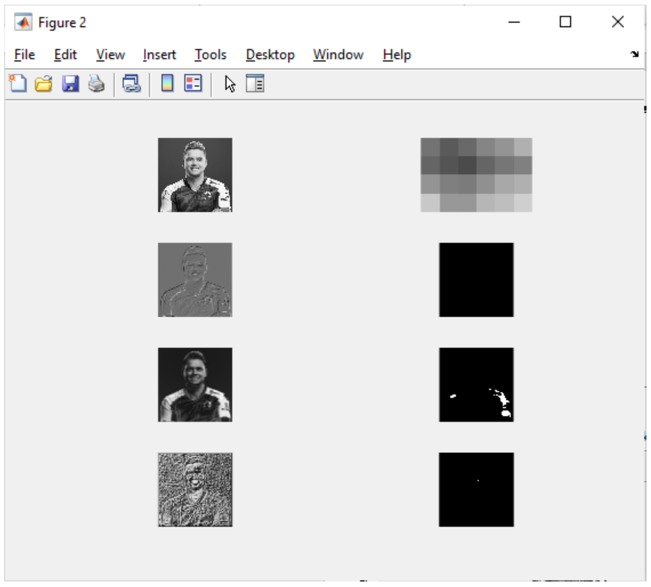
```
% Normalized Cross Correlation
NC = normxcorr2(filter, image1);

[mx, ind] = max(NC(:));
[y, xp] = ind2sub(size(NC), ind);
NC_error = sqrt(((original_x*scale(i)) - xp).^2 + ((original_y*scale(i)) - y).^2);

% Result Image
subplot(4, 2, 7);
imshow(NC, []);

% Thresholded Image
subplot(4,2,8);
imshow(NC > 0.9 * max(NC(:)),[]);
```

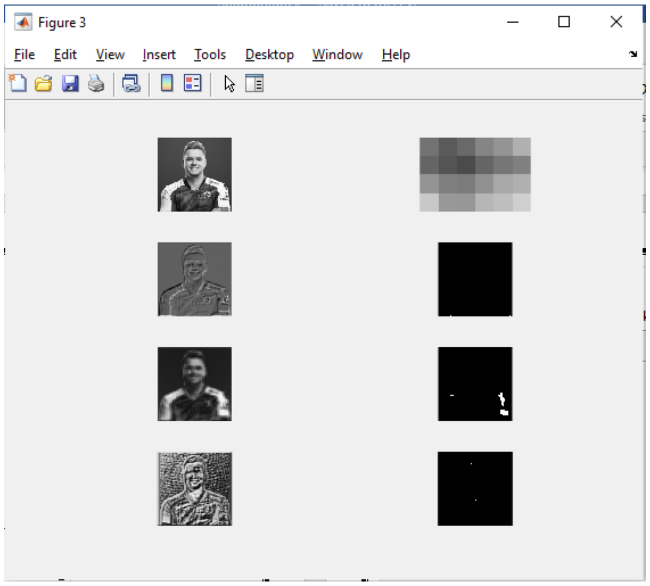
The output in the command window is as follows:

```
Sacling Factor - 2.000000
Error from ZMC - 527.638134
Error from SSD - 527.638134
Error from NC  - 109.931797
Sacling Factor - 1.000000
Error from ZMC - 243.016460
Error from SSD - 243.016460
Error from NC  - 55.443665
Sacling Factor - 0.500000
Error from ZMC - 63.694584
Error from SSD - 63.694584
Error from NC  - 16.643317
```

The output images are as follows:

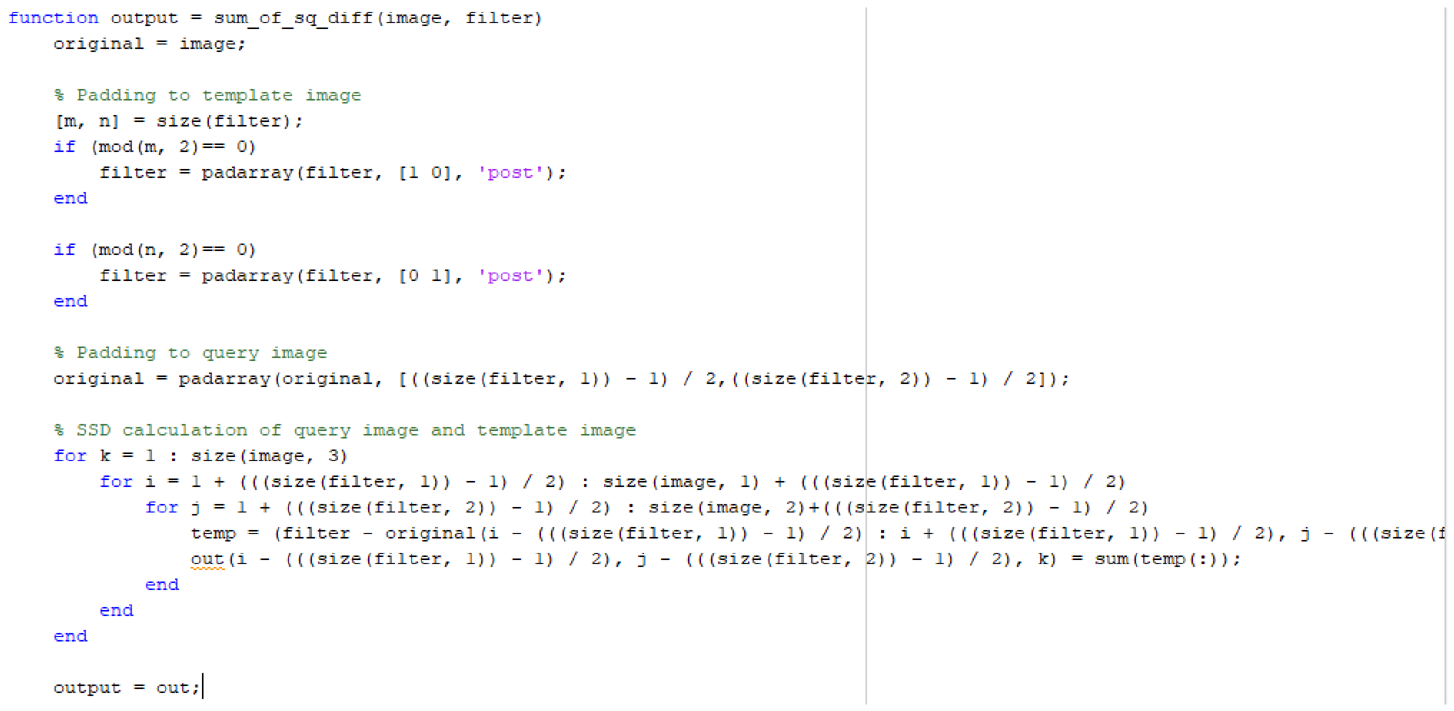






The functions zero_mean_corr and sum_of_sq_diff are as follows:

```
function output = zero_mean_corr(image, filter)
    original = image;

    % Padding to the template image
    [m, n] = size(filter);
    if (mod(m, 2) == 0)
        filter = padarray(filter, [1 0], 'post');
    end

    if (mod(n, 2) == 0)
        filter = padarray(filter, [0 1], 'post');
    end

    % Finding the mean of the filter
    mn = mean(filter(:));
    filter = filter - mn;

    % ZMC calculation of query image and template image
    original = padarray(original, [((size(filter, 1)) - 1) / 2, ((size(filter, 2)) - 1) / 2]);
    for k = 1: size(image, 3)
        for i = 1 + (((size(filter, 1)) - 1) / 2) : size(image, 1) + (((size(filter, 1)) - 1) / 2)
            for j = 1 + (((size(filter, 2)) - 1) / 2) : size(image, 2) + (((size(filter, 2)) - 1) / 2)
                temp = original(i - (((size(filter, 1)) - 1) / 2) : i + (((size(filter, 1)) - 1) / 2), j - (((size(filter, 2))
                out(i - (((size(filter, 1)) - 1) / 2), j - (((size(filter, 2)) - 1) / 2), k) = sum(temp(:));
            end
        end
    end

output = out;
```



```
function output = sum_of_sq_diff(image, filter)
    original = image;

    % Padding to template image
    [m, n] = size(filter);
    if (mod(m, 2)== 0)
        filter = padarray(filter, [1 0], 'post');
    end

    if (mod(n, 2)== 0)
        filter = padarray(filter, [0 1], 'post');
    end

    % Padding to query image
    original = padarray(original, [((size(filter, 1)) - 1) / 2,((size(filter, 2)) - 1) / 2]);

    % SSD calculation of query image and template image
    for k = 1 : size(image, 3)
        for i = 1 + (((size(filter, 1)) - 1) / 2) : size(image, 1) + (((size(filter, 1)) - 1) / 2)
            for j = 1 + (((size(filter, 2)) - 1) / 2) : size(image, 2)+(((size(filter, 2)) - 1) / 2)
                temp = (filter - original(i - (((size(filter, 1)) - 1) / 2) : i + (((size(filter, 1)) - 1) / 2), j - (((size(f
                out(i - (((size(filter, 1)) - 1) / 2), j - (((size(filter, 2)) - 1) / 2), k) = sum(temp(:));
            end
        end
    end

    output = out;
```

The most challenging part of this algorithm was to calculate the different measures with respect to different scales of image. In some cases, the localization of the template images fails because of the different scales used. As for increased scale, the pixels are repeated which creates a different view whereas in case of decreased scale, some of the pixels are removed resulting in a different view.

I thresholded the output image in order to see the proper output of the resulting image. This helps us to identify the most similar regions in the image, neglecting the least similar regions.

In conclusion, the similarity measures are an important tool for template matching. There is no such best measure from all the available measures. It depends on the available scenario. The SSD is faster but sensitive to overall intensity. In case of Normalized Cross-Correlation, it is slower but is invariant to the local average intensity and contrast. But, neither of these techniques represent the modern sophisticated techniques that are really good at template matching.

The output for the other images is as follows:
(1) Image 1:

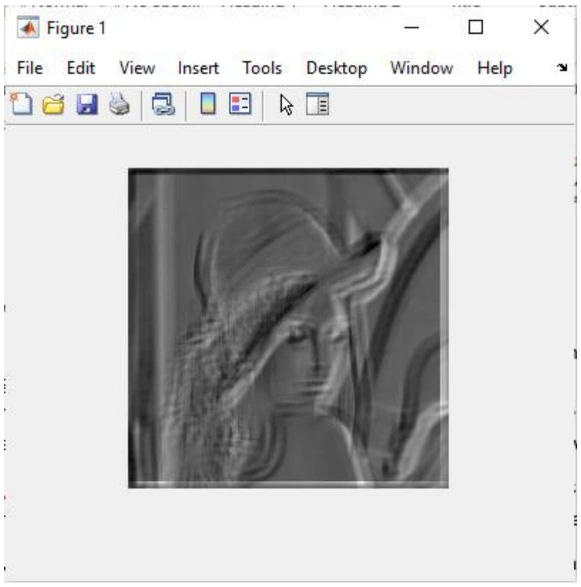


Figure 1
File Edit View Insert Tools Desktop Window Help

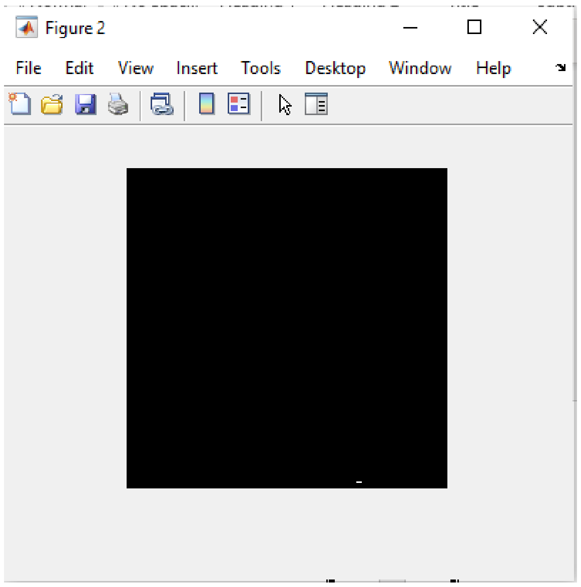


Figure 2
File Edit View Insert Tools Desktop Window Help

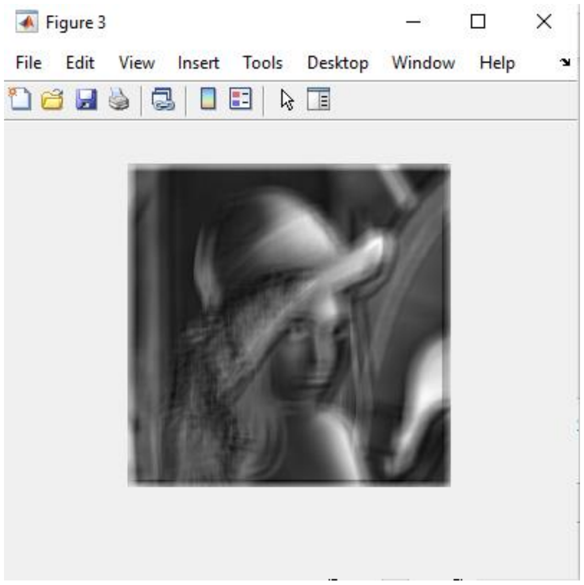


Figure 3
File Edit View Insert Tools Desktop Window Help

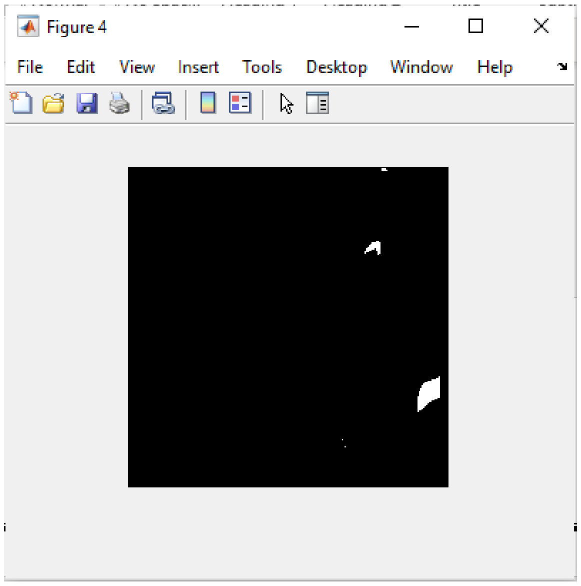


Figure 4
File Edit View Insert Tools Desktop Window Help

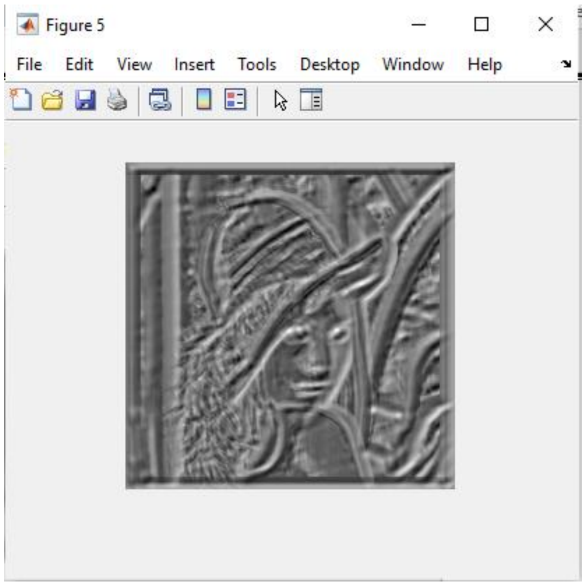


Figure 5
File Edit View Insert Tools Desktop Window Help

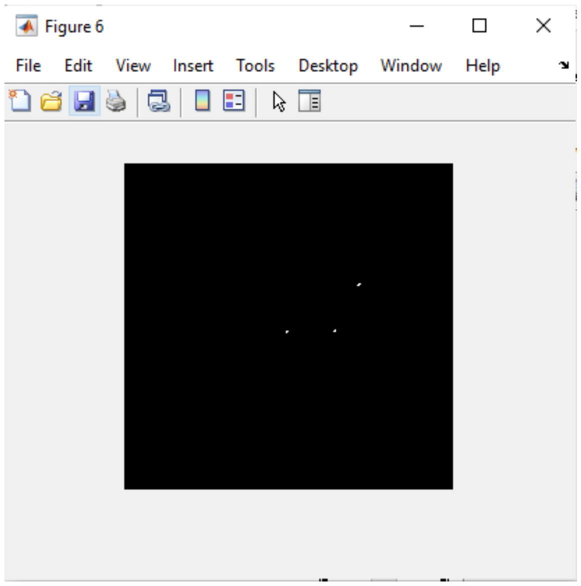


Figure 6
File Edit View Insert Tools Desktop Window Help

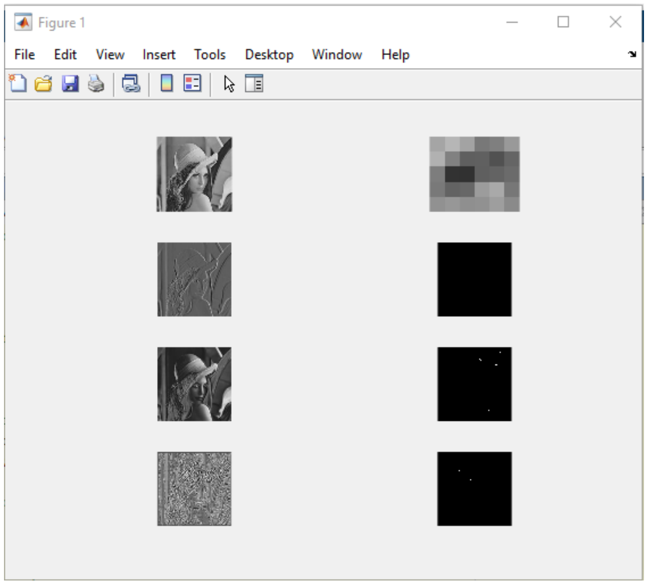


Figure 1
File Edit View Insert Tools Desktop Window Help

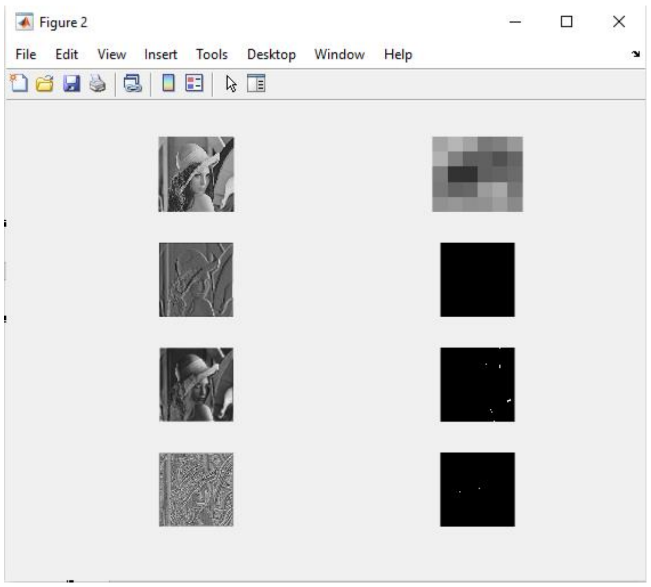


Figure 2
File Edit View Insert Tools Desktop Window Help

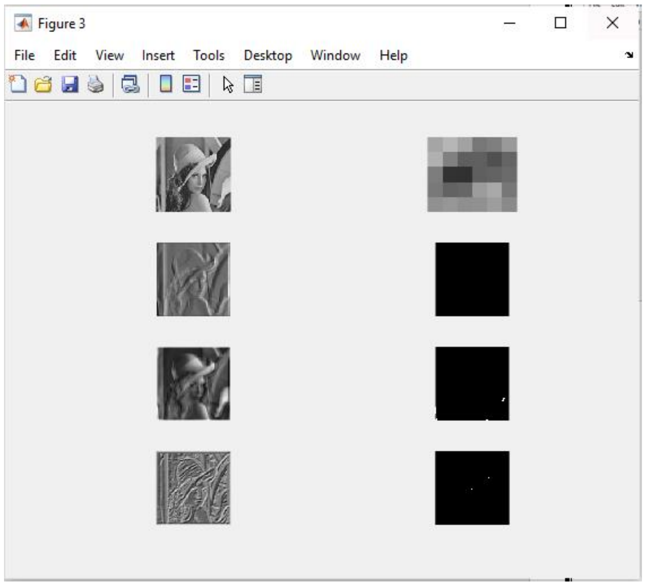
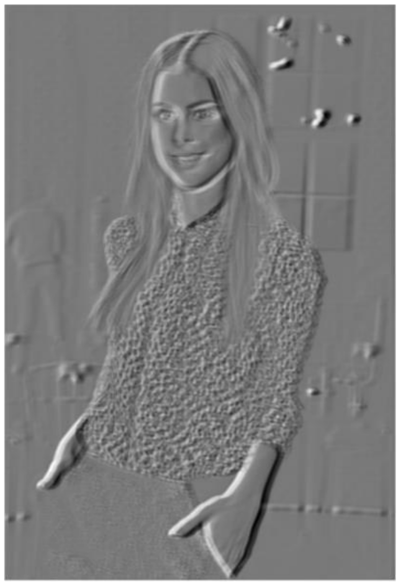
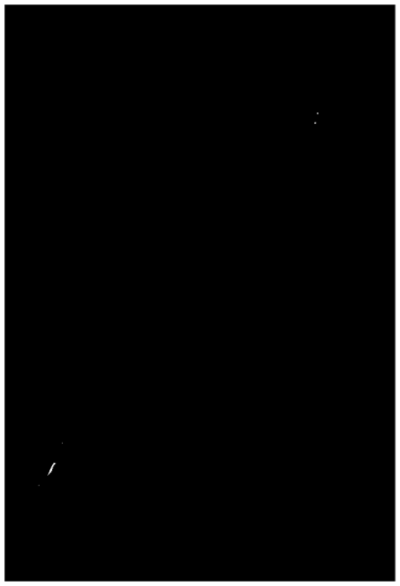
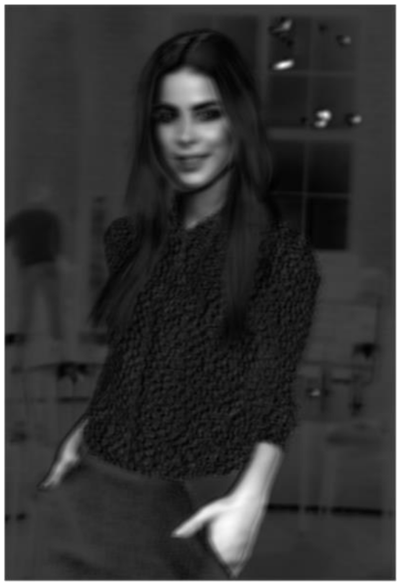


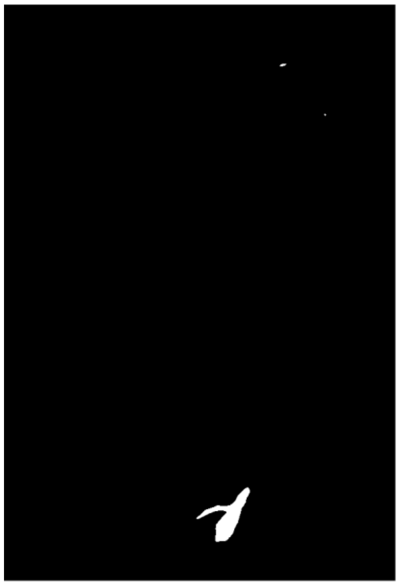
Figure 3
File Edit View Insert Tools Desktop Window Help

(2) Image 2:









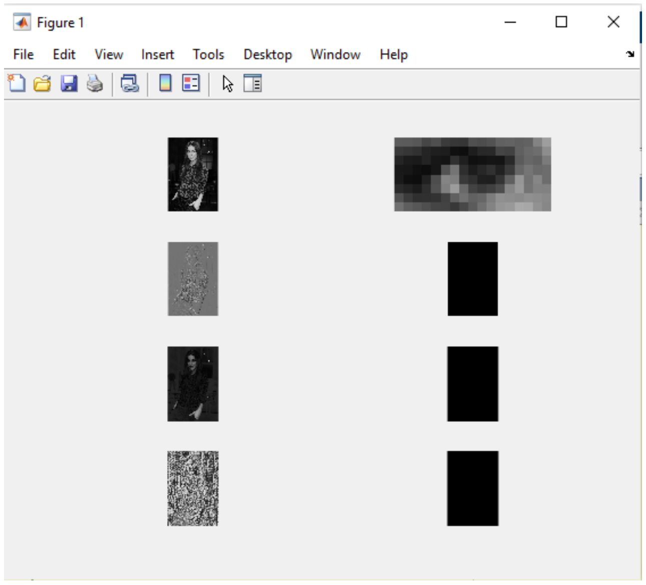


Figure 1
File Edit View Insert Tools Desktop Window Help

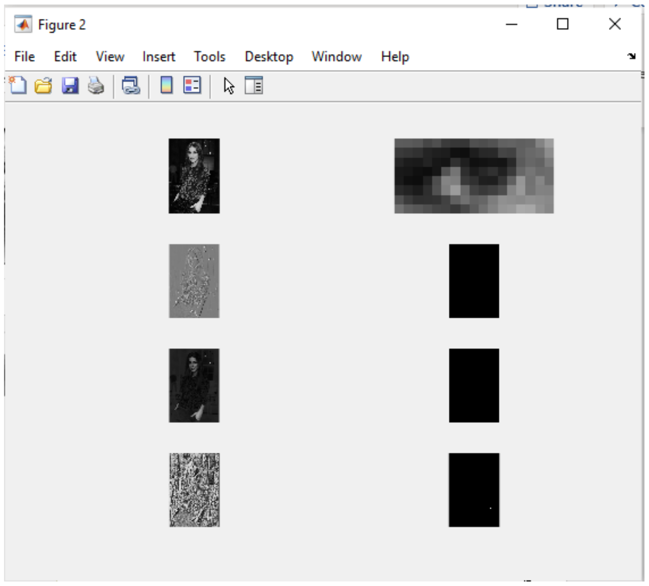



Figure 2
File Edit View Insert Tools Desktop Window Help

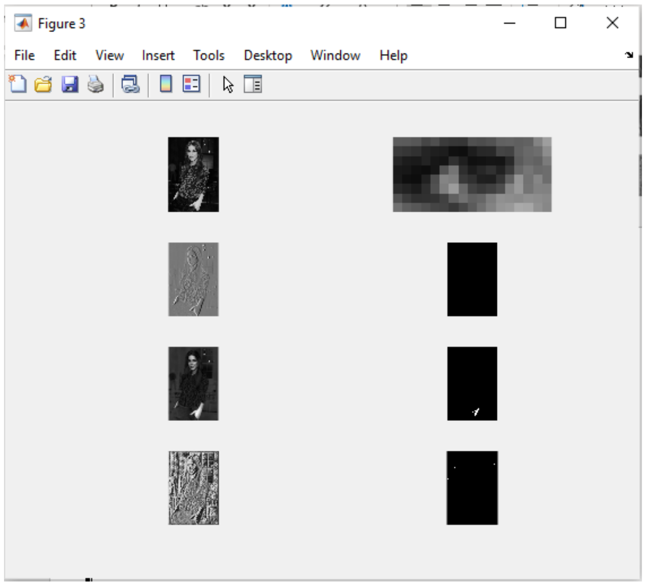



Figure 3
File Edit View Insert Tools Desktop Window Help

(3) Image 3:

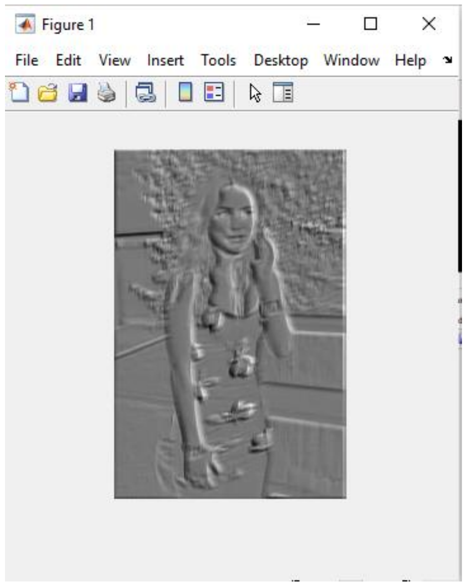


Figure 1
File Edit View Insert Tools Desktop Window Help

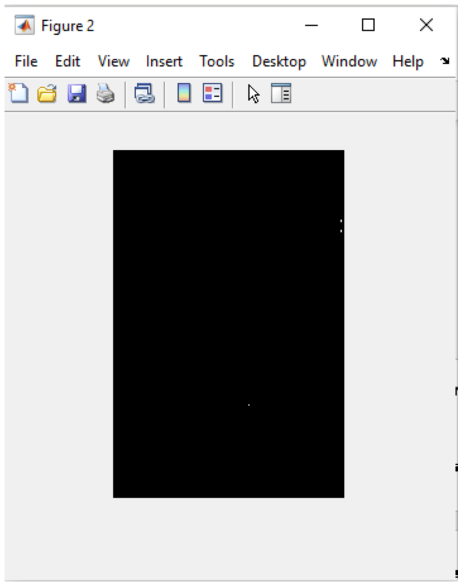


Figure 2
File Edit View Insert Tools Desktop Window Help

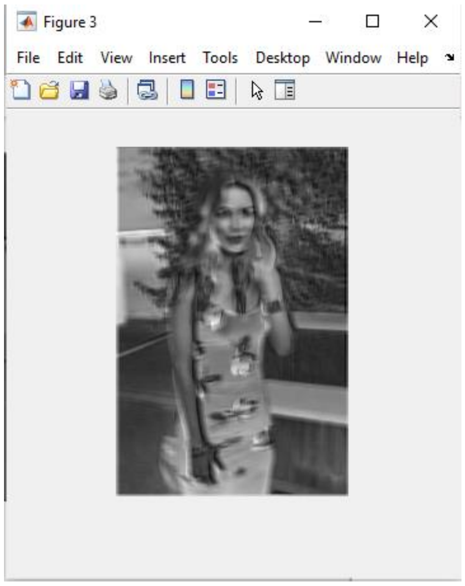


Figure 3
File Edit View Insert Tools Desktop Window Help

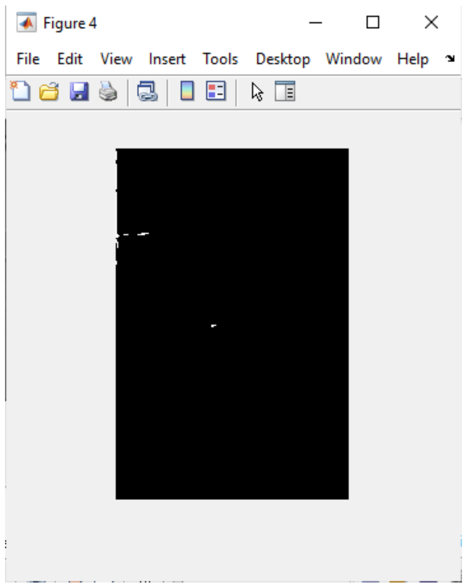


Figure 4
File Edit View Insert Tools Desktop Window Help

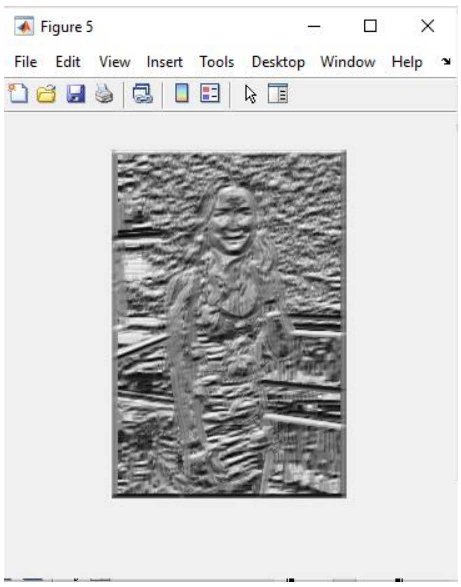


Figure 5
File Edit View Insert Tools Desktop Window Help

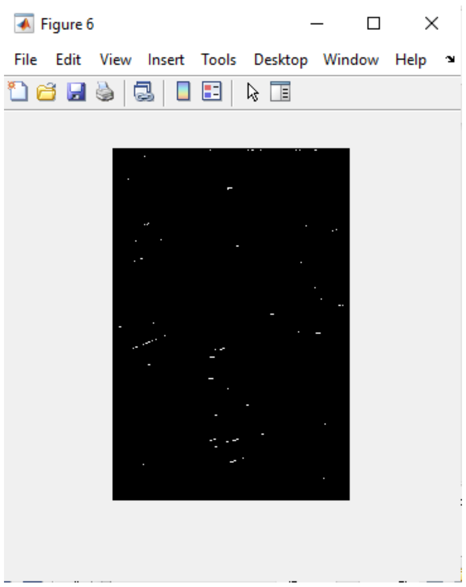


Figure 6
File Edit View Insert Tools Desktop Window Help

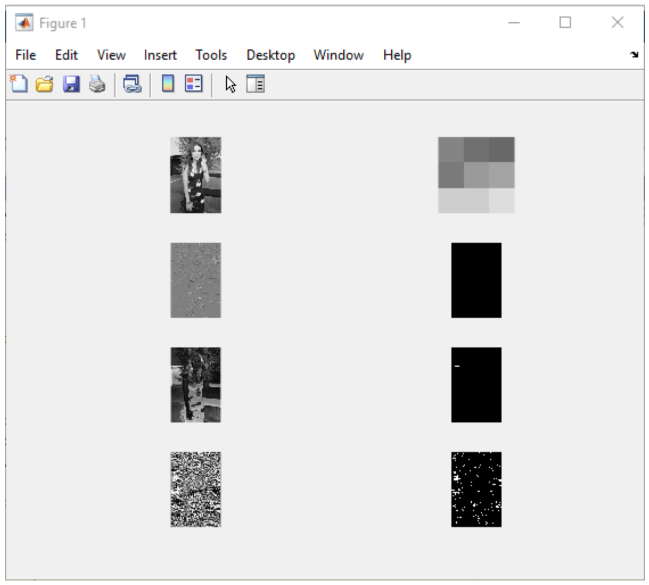


Figure 1
File Edit View Insert Tools Desktop Window Help

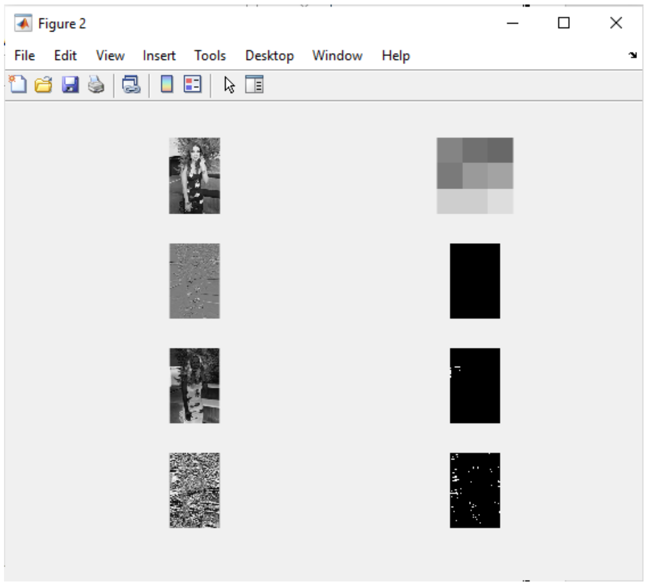


Figure 2
File Edit View Insert Tools Desktop Window Help

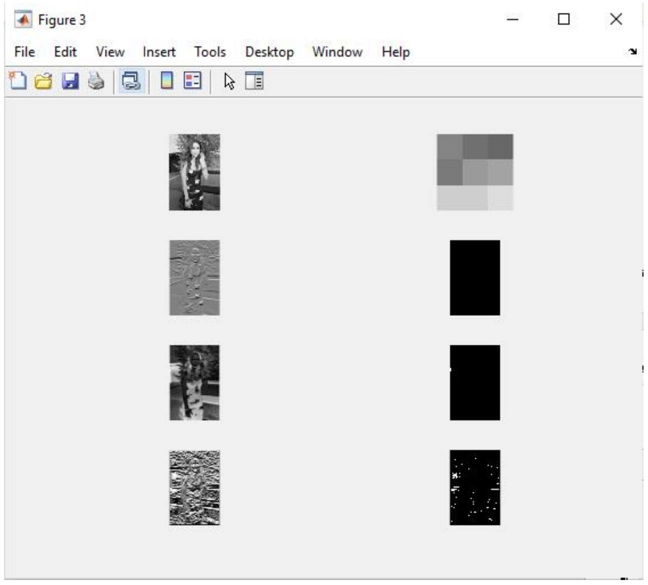


Figure 3
File Edit View Insert Tools Desktop Window Help