Name: Mian Usman Naeem Kakakhel
Id: 21701015
Section: 02
Matlab Assignment 2
17/04/2020

Note: All code is given at the end.

PART 1) a)

All pixels where the red component was greater than 140 was represented as black and the rest was represented as white.

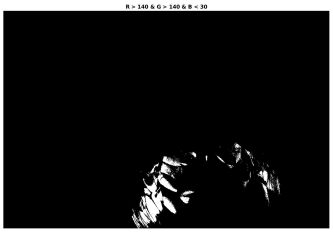
b)

All pixels where the green component was greater than 140 was represented as black and the rest was represented as white.

c)

All pixels where the blue component was greater than 140 was represented as black and the rest was represented as white.

d)



All pixels where the red component was greater than 140, green component was greater than 140 and blue was less than 30 was represented as black and the rest was represented as white.

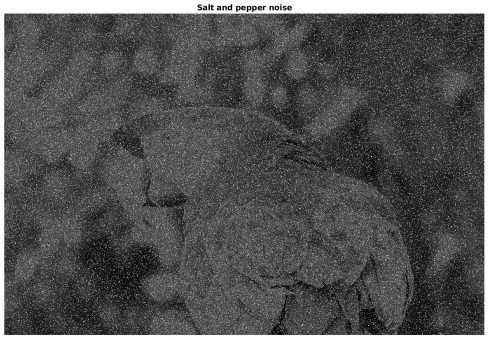
e)

Gray image.

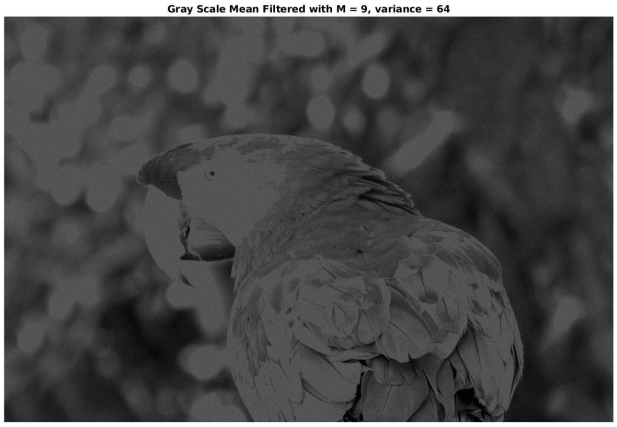
PART 2) Mean Filtering

b) Corrupted with gaussian noise of var = 64 and 1024.

Corrupted with salt and pepper noise

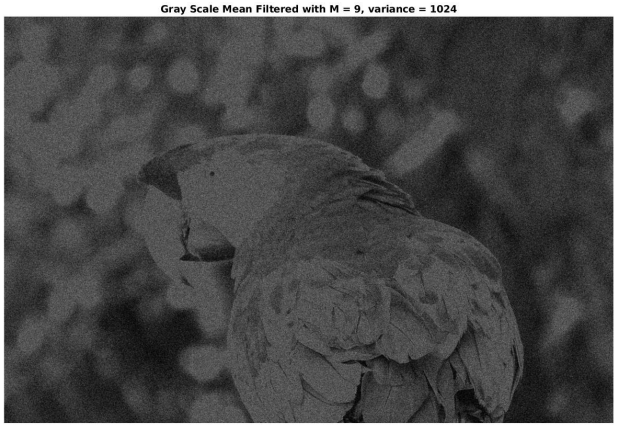


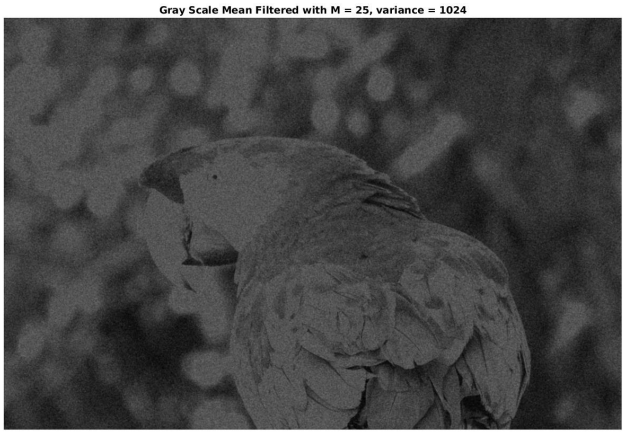
c) Mean Filter with M = 9, 25, 121, var = 64



This filter causes a blur effect. As the value of M increases, the picture becomes more blur and reduces the noise with it. The details of the picture on the other hand are diminished as the Value of M increases. Since we are filtering a pixel according to all of its neighbours, the visual effect in the n and m dimension is not different and are affected equally. Since a blur effect is taking place, the edges are kind of blending into the rest of the picture and do not seem like edges when M is increased.

d) Mean Filter with M = 9, 25, 121, var = 1024



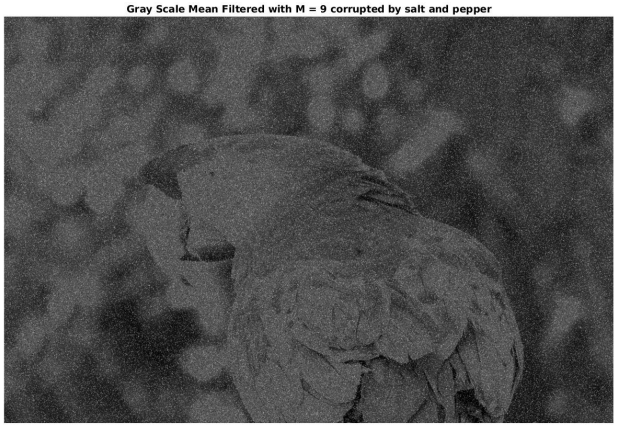


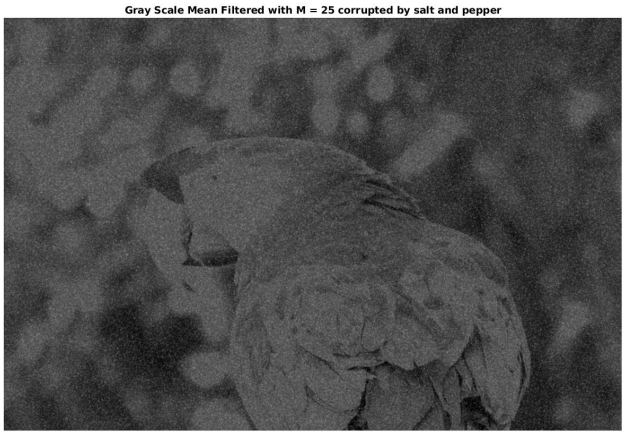
This filter causes a blur effect. As the value of M increases, the picture becomes more blur and reduces the noise with it. The details of the picture on the other hand are diminished as the Value of M increases. Since we are filtering a pixel according to all of its neighbours, the visual effect in the n and m dimension is not different and are affected equally. Since a blur effect is taking place, the edges are kind of blending into the rest of the picture and do not seem like edges when M is increased. Since the Noise was much greater than the previous part, the noise is not removed as effectively.

e) Salt and pepper

Basically we create a matrix of the size of imgray and then initialize it with random values between 0 and 1. The values in that matrix where the value is <= 1/16 is shown as black on imgray and the values in that matrix where the value is <= 15/16 is shown as white.

f) Mean Filter with M = 9, 25, 121, Salt and Pepper





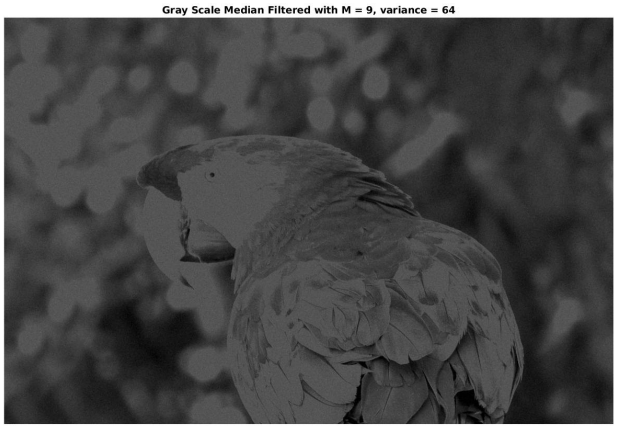
This filter causes a blur effect. As the value of M increases, the picture becomes more blur and reduces the noise with it. The details of the picture on the other hand are diminished as the Value of M increases.

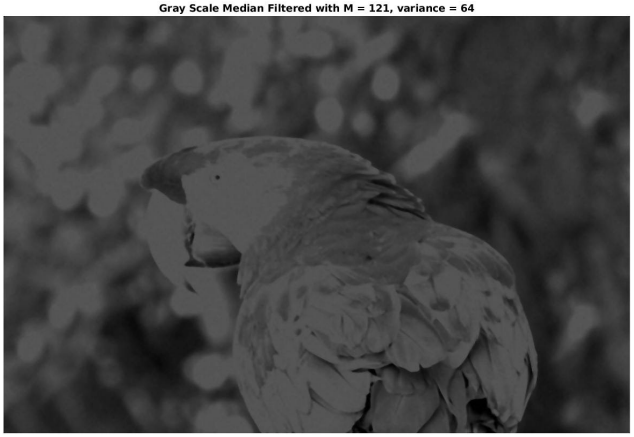
Since we are filtering a pixel according to all of its neighbours, the visual effect in the n and m dimension is not different and are affected equally. Since a blur effect is taking place, the edges are kind of blending into the rest of the picture and do not seem like edges when M is increased. Salt and Pepper noise was not reduced by Mean Filtering effectively.

PART 3) Median Filtering

b) The Corrupted images are the same as in part 2.

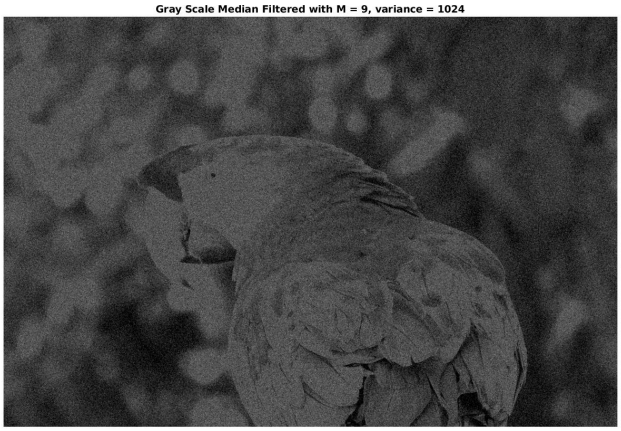
c) Median Filter with M = 9, 25, 121, var = 64

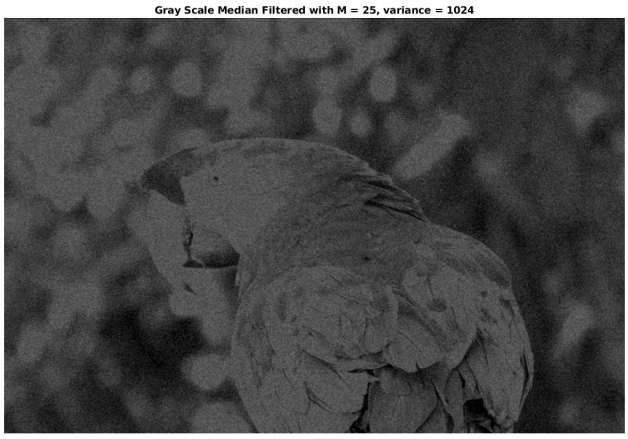




This filter causes a blur effect but the blur effect is not as strong as in the mean filtering. As the value of M increases, the picture becomes more blur and reduces the noise with it. The details of the picture on the other hand are diminished as the Value of M increases the details seem to diminish more than mean filtering. Since we are filtering a pixel according to all of its neighbours, the visual effect in the n and m dimension is not different and are affected equally. Since a blur effect is taking place, the edges are kind of blending into the rest of the picture but since the blur effect is not as strong as the mean filter, the edges are still quite distinguishable.

d) Median Filter with M = 9, 25, 121, var = 1024



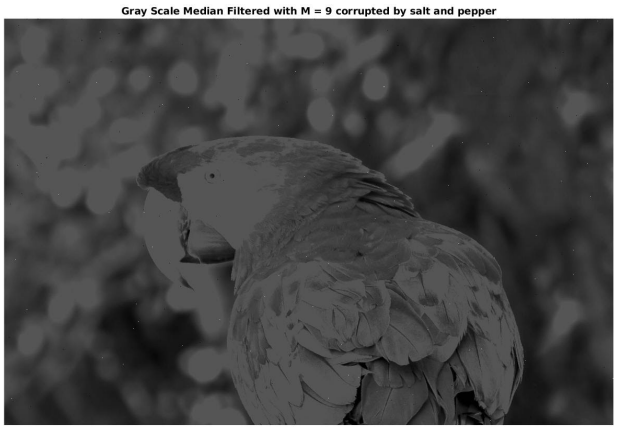


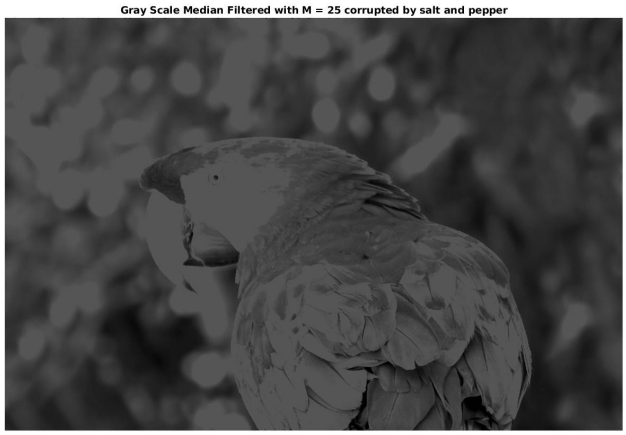
This filter causes a blur effect but the blur effect is not as strong as in the mean filtering. As the value of M increases, the picture becomes more blur and reduces the noise with it. The details of the picture on the other hand are diminished as the Value of M increases the details seem to diminish more than mean filtering. Since we are filtering a pixel according to all of its neighbours, the visual effect in the n and m dimension is not different and are affected equally. Since a blur effect is taking place, the edges are kind of blending into the rest of the picture but since the blur effect is not as strong as the mean filter, the edges are still quite distinguishable. Since the Noise was much greater than the previous part, the noise is not removed as effectively.

e) Salt and pepper

Basically we create a matrix of the size of imgray and then initialize it with random values between 0 and 1. The values in that matrix where the value is <= 1/16 is shown as black on imgray and the values in that matrix where the value is <= 15/16 is shown as white.

f) Median Filter with M = 9, 25, 121, Salt and Pepper



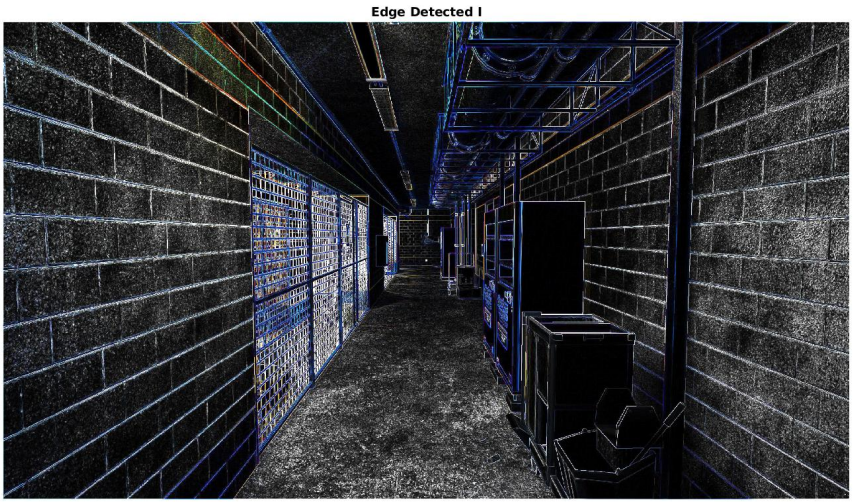


This filter causes a blur effect but the blur effect is not as strong as in the mean filtering. As the value of M increases, the picture becomes more blur and reduces the noise with it. The details of the picture on the other hand are diminished as the Value of M increases the details seem to diminish more than mean filtering. Since we are filtering a pixel according to all of its neighbours, the visual effect in the n and m dimension is not different and are affected equally. Since a blur effect is taking place, the edges are kind of blending into the rest of the picture but since the blur effect is not as strong as the mean filter, the edges

are still quite distinguishable. Median filtering seems to have reduced the salt and pepper noise quite effectively.

PART 4) Edge Detection

Edged Image



It seems that this detection method shows white where there seems to be an edge and dark colors where there is a smooth surface.

CODE Mean Filter)

```
function J = mean_filter(I,M)
    I = double(I);
    N = (sqrt(M) - 1) / 2;
    size_I = size(I);
    J = zeros(size_I(1), size_I(2));
    for m = 1:size_I(1)
        for n = 1:size_I(2)
            my_sum = 0;
            for i = m - N:m + N
                for j = n - N:n + N
                    if i >= 1 && j >= 1 && i <= size_I(1) && j <= size_I(2)
                        my_sum = my_sum + I(i,j);
                    end
                end
            end
            J(m,n) = my_sum / M;
        end
```

```
    end
    J = uint8(J);
end
```

CODE Median Filter)

```
function J = median_filter(I,M)
    I = double(I);
    N = (sqrt(M) - 1) / 2;
    size_I = size(I);
    J = zeros(size_I(1), size_I(2));
    for m = 1:size_I(1)
        for n = 1:size_I(2)
            my_median = zeros(M,1);
            cntr = 1;
            for i = m - N:m + N
                for j = n - N:n + N
                    if i >= 1 && j >= 1 && i <= size_I(1) && j <= size_I(2)
                        my_median(cntr) = I(i,j);
                    end
                    cntr = cntr + 1;
                end
            end
            J(m,n) = median(my_median);
        end
    end
    J = uint8(J);
end
```

CODE PART 1)

```
A = imread('image3.jpg');
size_A = size(A);

R_grey = zeros(size_A(1), size_A(2));
R_indices = find(A(:,:,1) > 140);
R_grey(R_indices) = 1;
R_img = imshow(R_grey);
caption = sprintf('R > 140');
title(caption, 'FontSize', 14);
drawnow;
```

```
waitfor(R_img);
G_grey = zeros(size_A(1), size_A(2));
G_indices = find(A(:,:,2) > 140);
G_grey(G_indices) = 1;
G_img = imshow(G_grey);
caption = sprintf('G > 140');
title(caption, 'FontSize', 14);
drawnow;

waitfor(G_img);
B_grey = zeros(size_A(1), size_A(2));
B_indices = find(A(:,:,3) > 140);
B_grey(B_indices) = 1;
B_img = imshow(B_grey);
caption = sprintf('B > 140');
title(caption, 'FontSize', 14);
drawnow;

waitfor(B_img);
RGB_grey = zeros(size_A(1), size_A(2));
RGB_indices = find(A(:,:,1) > 140 & A(:,:,2) > 140 & A(:,:,3) < 30);
RGB_grey(RGB_indices) = 1;
RGB_img = imshow(RGB_grey);
caption = sprintf('R > 140 & G > 140 & B < 30');
title(caption, 'FontSize', 14);
drawnow;

waitfor(RGB_img);
grey = zeros(size_A(1), size_A(2));
grey = (A(:,:,1) + A(:,:,2) + A(:,:,3)) / 3;
grey_img = imshow(grey);
caption = sprintf('Grey Scale');
title(caption, 'FontSize', 14);
drawnow;
imwrite(grey,'gray.jpg')
```

CODE PART 2)

```
imgray = imread('gray.jpg');
gaussnoise = 8*randn(size(imgray,1), size(imgray,2));
imgaussnoise = uint8(double(imgray) + gaussnoise);
im = imshow(imgaussnoise);
caption = sprintf('Corrupted Gray Scale with variance = 64');
title(caption, 'FontSize', 14);
drawnow;
%part c
waitfor(im);
M9_mat = mean_filter(imgaussnoise, 9);
M9_im = imshow(M9_mat);
caption = sprintf('Gray Scale Mean Filtered with M = 9, variance = 64');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M9_im);
M25_mat = mean_filter(imgaussnoise, 25);
M25_im = imshow(M25_mat);
caption = sprintf('Gray Scale Mean Filtered with M = 25, variance = 64');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M25_im);
M121_mat = mean_filter(imgaussnoise, 121);
M121_im = imshow(M121_mat);
caption = sprintf('Gray Scale Mean Filtered with M = 121, variance = 64');
title(caption, 'FontSize', 14);
drawnow;
% part d
waitfor(M121_im);
gaussnoise = 32*randn(size(imgray,1), size(imgray,2));
imgaussnoise = uint8(double(imgray) + gaussnoise);
im = imshow(imgaussnoise);
caption = sprintf('Corrupted Gray Scale with variance = 1024');
title(caption, 'FontSize', 14);
drawnow;

waitfor(im);
```

```
M9_mat = mean_filter(imgaussnoise, 9);
M9_im = imshow(M9_mat);
caption = sprintf('Gray Scale Mean Filtered with M = 9, variance = 1024');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M9_im);
M25_mat = mean_filter(imgaussnoise, 25);
M25_im = imshow(M25_mat);
caption = sprintf('Gray Scale Mean Filtered with M = 25, variance = 1024');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M25_im);
M121_mat = mean_filter(imgaussnoise, 121);
M121_im = imshow(M121_mat);
caption = sprintf('Gray Scale Mean Filtered with M = 121, variance = 1024');
title(caption, 'FontSize', 14);
drawnow;

%part e salt and pepper noise
waitfor(M121_im);
imsaltnoise = imgray;
noisypixels = rand( size(imgray,1), size(imgray,2) );
imsaltnoise( find( noisypixels <= ( 1 / 16 ) ) ) = 255;
imsaltnoise( find( noisypixels >= ( 15 / 16 ) ) ) = 0;
im = imshow(imsaltnoise);
caption = sprintf('Salt and pepper noise');
title(caption, 'FontSize', 14);
drawnow;
%part f filtering for salt and pepper
waitfor(im);
M9_mat = mean_filter(imsaltnoise, 9);
M9_im = imshow(M9_mat);
caption = sprintf('Gray Scale Mean Filtered with M = 9 corrupted by salt and pepper');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M9_im);
```

```
M25_mat = mean_filter(imsaltnoise, 25);
M25_im = imshow(M25_mat);
caption = sprintf('Gray Scale Mean Filtered with M = 25 corrupted by salt and pepper');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M25_im);
M121_mat = mean_filter(imsaltnoise, 121);
M121_im = imshow(M121_mat);
caption = sprintf('Gray Scale Mean Filtered with M = 121 corrupted by salt and pepper');
title(caption, 'FontSize', 14);
drawnow;
```

CODE PART 3)

```
imgray = imread('gray.jpg');
gaussnoise = 8*randn(size(imgray,1), size(imgray,2));
imgaussnoise = uint8(double(imgray) + gaussnoise);
im = imshow(imgaussnoise);
caption = sprintf('Corrupted Gray Scale with variance = 64');
title(caption, 'FontSize', 14);
drawnow;
%part c
waitfor(im);
M9_mat = median_filter(imgaussnoise, 9);
M9_im = imshow(M9_mat);
caption = sprintf('Gray Scale Median Filtered with M = 9, variance = 64');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M9_im);
M25_mat = median_filter(imgaussnoise, 25);
M25_im = imshow(M25_mat);
caption = sprintf('Gray Scale Median Filtered with M = 25, variance = 64');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M25_im);
M121_mat = median_filter(imgaussnoise, 121);
```

```
M121_im = imshow(M121_mat);
caption = sprintf('Gray Scale Median Filtered with M = 121, variance = 64');
title(caption, 'FontSize', 14);
drawnow;
% part d
waitfor(M121_im);
gaussnoise = 32*randn(size(imgray,1), size(imgray,2));
imgaussnoise = uint8(double(imgray) + gaussnoise);
im = imshow(imgaussnoise);
caption = sprintf('Corrupted Gray Scale with variance = 1024');
title(caption, 'FontSize', 14);
drawnow;

waitfor(im);
M9_mat = median_filter(imgaussnoise, 9);
M9_im = imshow(M9_mat);
caption = sprintf('Gray Scale Median Filtered with M = 9, variance = 1024');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M9_im);
M25_mat = median_filter(imgaussnoise, 25);
M25_im = imshow(M25_mat);
caption = sprintf('Gray Scale Median Filtered with M = 25, variance = 1024');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M25_im);
M121_mat = median_filter(imgaussnoise, 121);
M121_im = imshow(M121_mat);
caption = sprintf('Gray Scale Median Filtered with M = 121, variance = 1024');
title(caption, 'FontSize', 14);
drawnow;

%part e salt and pepper noise
waitfor(M121_im);
imsaltnoise = imgray;
noisypixels = rand( size(imgray,1), size(imgray,2) );
imsaltnoise( find( noisypixels <= ( 1 / 16 ) ) ) = 255;
```

```
imsaltnoise( find( noisypixels >= ( 15 / 16 ) ) ) = 0;
im = imshow(imsaltnoise);
caption = sprintf('Salt and pepper noise');
title(caption, 'FontSize', 14);
drawnow;
%part f filtering for salt and pepper
waitfor(im);
M9_mat = median_filter(imsaltnoise, 9);
M9_im = imshow(M9_mat);
caption = sprintf('Gray Scale Median Filtered with M = 9 corrupted by salt and pepper');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M9_im);
M25_mat = median_filter(imsaltnoise, 25);
M25_im = imshow(M25_mat);
caption = sprintf('Gray Scale Median Filtered with M = 25 corrupted by salt and pepper');
title(caption, 'FontSize', 14);
drawnow;

waitfor(M25_im);
M121_mat = median_filter(imsaltnoise, 121);
M121_im = imshow(M121_mat);
caption = sprintf('Gray Scale Median Filtered with M = 121 corrupted by salt and pepper');
title(caption, 'FontSize', 14);
drawnow;
```

CODE PART 4)

```
I = imread('image3part4.jpg');
GX_mat = [1 0 -1; 2 0 -2; 1 0 -1];
GY_mat = [1 2 1; 0 0 0; -1 -2 -1];

GX_R = conv2(GX_mat, I(:, :, 1));
GX_G = conv2(GX_mat, I(:, :, 2));
GX_B = conv2(GX_mat, I(:, :, 3));

GY_R = conv2(GY_mat, I(:, :, 1));
GY_G = conv2(GY_mat, I(:, :, 2));
GY_B = conv2(GY_mat, I(:, :, 3));
```

```
GX = zeros(size(GX_R,1), size(GX_R,2), 3);
GY = zeros(size(GY_R,1), size(GY_R,2), 3);

GX(:,:,1) = GX_R;
GX(:,:,2) = GX_G;
GX(:,:,3) = GX_B;

GY(:,:,1) = GY_R;
GY(:,:,2) = GY_G;
GY(:,:,3) = GY_B;

G = uint8(sqrt(GX.^2 + GY.^2));

I_img = imshow(I);
caption = sprintf('Original I');
title(caption, 'FontSize', 14);
drawnow;

waitfor(I_img);
G_img = imshow(G);
caption = sprintf('Edge Detected I');
title(caption, 'FontSize', 14);
drawnow;
```