Muhammad Arham Khan
21701848
Section 1
Matlab Assignment 2

I will be using image0.jpg for this part as my initial image:

Part 1: RGB Components

a)

Comment: The image basically just shows a binary representation of the regions in the image that have a red component intensity of higher than 140. So, fruits or elements in the image with R>140 are white while the others are black.

b)

Comment: The image basically just shows a binary representation of the regions in the image that have a green component intensity of higher than 140. So, fruits or elements in the image with G>140 are white while the others are black.

c)

Comment: The image basically just shows a binary representation of the regions in the image that have a blue component intensity of higher than 140. So, fruits or elements in the image with B>140 are white while the others are black.

d)

Comment: The image basically just shows a binary representation of the regions in the image that have a red component value of higher than 140, a green component value of

higher than 140 and blue component value of less than 30. So, such regions are white while the others are black.

e)

Part 2: Mean Filtering

a)

```
function [J] = func1( I, M)
    [rows, cols] = size(I);
    fprintf('%d by %d\n', rows, cols);

    J = rand(rows,cols);

    N = (sqrt( M) - 1) / 2;

    for row=1:rows
        for col=1:cols

            sumTemp = 0.0;
            for r=(row-N):(row+N)
                for c=(col-N):(col+N)
                    if( r > 0) && (c > 0) && ( r <= rows) && ( c <= cols)
                        sumTemp = double(sumTemp) + double( I(r, c));
                    end
                end
            end
            J(row, col) = double(sumTemp) / double(M) / 255.0;
        end
    end
end
```

b)

c)

M = 9

M = 25

M = 121

Comment: The visual effect observed is that of a blur effect because as the value of M increases, the blurriness of the image increases too (reducing the noise in the image). As the value of M is increased, the details in the image are reduced as the image becomes less sharp and blurred (due to the mean filtering of individual pixels). The effect is the same in both dimensions because the region of selection is a perfect square of width (2N + 1) so there is no obvious or expected change in the effect. The image becomes darker towards the edges due to the pixel correction values of 0 being added to the mean calculation matrix. If the value of M increases, the darkness increases towards the edges too.

d) The Gaussian noise is increased to a value of 1024 (obtained by replacing the 8 in the provided equation with 32).

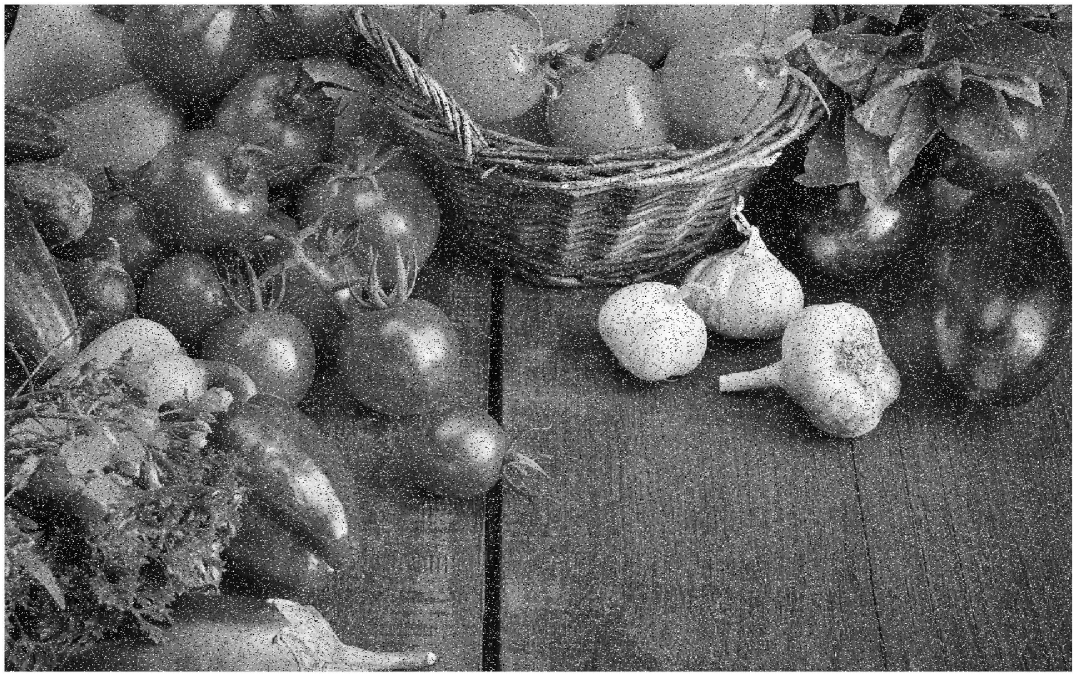
M = 9

M = 25

M = 121

Comment: same observations as in part c

e) Image obtained by applying salt and pepper noise to the image titled “gray.jpg”



Comment: The noise adds black and white pixels (hence the name salt and pepper) across the image in random positions and hence makes the image polluted with a lot of noice that is either black or white in color.

f)

M = 9

M = 25

M = 121

Comment: The filter reduces the noise in the salt and pepper image obtained earlier in part(e) by clearing out the noisy pixels and making the image more blurred and much less detailed in the process. The details in the image seem to be lost due to the mean function being applied to the pixels and as the value of M increases, the image appears to be less noisy but more hazy (blurred). The effect is the same in both dimensions because the region of selection is a perfect square of width (2N + 1) so there is no obvious or expected change in the effect. As with the previous filter, the image appears to have a dark blackish vignette around the edges due to the 0 values added to the mean function for the correction of M pixels near the edges. The black vignette increases as the value of M increases.

Part 3: Median Filtering

a)

```
function [J] = func2( I, M)
    [rows, cols] = size(I);
    fprintf('%d by %d\n', rows, cols);

    J = rand(rows,cols);

    N = (sqrt( M) - 1) / 2;

    for row=1:rows
        for col=1:cols

            %Making a temporary matrix with 0 values
            len = (N * 2) + 1;
            tempMatrix = zeros(len,len);

            if (row - N) < 1; Xstart = 1; else; Xstart = row - N; end
            if (row + N) > rows; Xend = rows; else; Xend = row + N; end
            if (col - N) < 1; Ystart = 1; else; Ystart = col - N; end
            if (col + N) > cols; Yend = cols; else; Yend = col + N; end
```

```
            xTemp = 1;
            for a=Xstart:Xend
                yTemp = 1;
                for b=Ystart:Yend
                    tempMatrix(xTemp, yTemp) = I(a, b);
                    yTemp = yTemp + 1;
                end
                xTemp = xTemp + 1;
            end


            %Finding median from temporary matrix
            medRow = double(median( tempMatrix(1:len,1:len)));
            medFinal = median( medRow);
            fprintf("%f, ", medFinal);
            J(row, col) = double(medFinal) / 255.0;

        end
    end
end
```

b)

c)

M = 9

M = 25

M = 121

Comment: The visual effect observed seems to be that of a smooth texture as the original image was a bit noisy and polluted. Unlike the previous filter in part 2, median filter basically smoothens the image out although some details are lost, the image appears to be sharp in general with solid boundaries. As expected, as we increase the M, the amount of smoothing effect applied to the image increases as well and the image obtained appears to be clearer due to the larger median selection region available. The effect is the same in both dimensions because the region of selection is a perfect square of width (2N + 1) so there is no obvious or expected change in the effect. As we close to the edges of the image in this filter, a darker black vignette is observed in the images due to increasing number of zeros in the sample of size M and the median inclines towards 0. With increasing M, the black vignette region increases as well.

d) The Gaussian noise is increased to a value of 1024 (obtained by replacing the 8 in the provided equation with 32).

M = 9

M = 25

M = 121

Comment: same observations as in part c

e) Image obtained by applying salt and pepper noise to the image titled “gray.jpg”

Comment: The noise adds black and white pixels (hence the name salt and pepper) across the image in random positions and hence makes the image polluted with a lot of noise that is either black or white in color.

f)

M = 9

M = 25

M = 121

Comment: Median filtering function in majority has a clearing effect on the image as it reduces the noise although it does blur the image a bit, in general it performs well in clearing the salt and pepper noise from the image. The noise in the image decreases as the value of M increases (as expected because the median calculation gets a higher number of pixels and a more general pixel correction is obtained). The effect is the same in both dimensions because the region of selection is a perfect square of width (2N + 1) so there is no obvious or expected change in the effect. The image gets a darker vignette near the edges of the image due to 0 values added to the calculation matrix for pixel size correction. The vignette size and darkness increases as the value of M increases.

Part 4: Edge Detection with the Sobel Operator

Comments: The sobel operator edge detection basically detects edge lines in the image and highlights them against a darker background.

CODE:

```
% Part 4
I =imread("image0part4.jpg");

[rows, cols, width] = size(I);
fprintf('%d by %d by %d\n', rows, cols, width);

xMatrix = [1 0 -1; 2 0 -2; 1 0 -1];
yMatrix = [1 2 1; 0 0 0; -1 -2 -1];

rxConv = conv2( xMatrix, I(:,:, 1));
bxConv = conv2( xMatrix, I(:,:, 2));
gxConv = conv2( xMatrix, I(:,:, 3));

ryConv = conv2( yMatrix, I(:,:, 1));
byConv = conv2( yMatrix, I(:,:, 2));
gyConv = conv2( yMatrix, I(:,:, 3));

Gred = sqrt( ((rxConv .^ 2) + (ryConv .^ 2)));
Gblue = sqrt( ((bxConv .^ 2) + (byConv .^ 2)));
Ggreen = sqrt( ((gxConv .^ 2) + (gyConv .^ 2)));

[r, c] = size(Gred);
fprintf('%d by %d\n', r, c);

out = ones(r, c, width);

out(:,:,1) = Gred;
out(:,:,2) = Gblue;
out(:,:,3) = Ggreen;

out = uint8(out);

imshow(out)



%Part 3
imgray =imread("gray.jpg");
imsaltnoise = imgray;
noisypixels = rand( size(imgray,1), size(imgray,2) );
imsaltnoise( find( noisypixels <= ( 1 / 16 ) ) ) = 255;
imsaltnoise( find( noisypixels >= ( 15 / 16 ) ) ) = 0;


out = func2( imsaltnoise, 121);
imshow( out);

function [J] = func2( I, M)
    [rows, cols] = size(I);
    fprintf('%d by %d\n', rows, cols);

    J = rand(rows,cols);

    N = (sqrt( M) - 1) / 2;

    for row=1:rows
        for col=1:cols

            %Making a temporary matrix with 0 values
            len = (N * 2) + 1;
            tempMatrix = zeros(len,len);

            if (row - N) < 1; Xstart = 1; else; Xstart = row - N; end
            if (row + N) > rows; Xend = rows; else; Xend = row + N; end
            if (col - N) < 1; Ystart = 1; else; Ystart = col - N; end
            if (col + N) > cols; Yend = cols; else; Yend = col + N; end

            xTemp = 1;
            for a=Xstart:Xend
                yTemp = 1;
                for b=Ystart:Yend
                    tempMatrix(xTemp, yTemp) = I(a, b);
                    yTemp = yTemp + 1;
                end
                xTemp = xTemp + 1;
            end


            %Finding median from temporary matrix
            medRow = double(median( tempMatrix(1:len,1:len)));
            medFinal = median( medRow);
```

```
            J(row, col) = double(medFinal) / 255.0;

        end
    end
end

function [J] = func1( I, M)
    [rows, cols] = size(I);
    fprintf('%d by %d\n', rows, cols);

    J = rand(rows,cols);

    N = (sqrt( M) - 1) / 2;

    for row=1:rows
        for col=1:cols

            sumTemp = 0.0;
            for r=(row-N):(row+N)
                for c=(col-N):(col+N)
                    if( r > 0) && (c > 0) && ( r <= rows) && ( c <= cols)
                        sumTemp = double(sumTemp) + double( I(r, c));
                    end
                end
            end
            J(row, col) = double(sumTemp) / double(M) / 255.0;
        end
    end
end




%Part 2

[rows, cols, width] = size(I);
fprintf('%d by %d by %d\n', rows, cols, width);

out = rand(rows,cols);


for row=1:rows
    for col=1:cols

        %R pixel
        if A(row, col, 1) > 140
            out(row, col) = 1;
        else
            out(row, col) = 0;
        end

        %G pixel
        if A(row, col, 2) > 140
            out(row, col) = 1;
        else
            out(row, col) = 0;
        end

         B pixel
         if A(row, col, 3) > 140
             out(row, col) = 1;
         else
             out(row, col) = 0;
         end

         if ((A(row, col, 1) > 140) && (A(row, col, 2) > 140) && (A(row, col, 3) < 30))
             out(row, col) = 1;
         else
             out(row, col) = 0;
         end

 for r=1:rows
     for c=1:cols
         avg = (double(A(r, c, 1)) + double(A(r, c, 2)) + double(A(r, c, 3))) / 3.0;
         avg = double(avg) / 255.0;
         out(r, c) = avg;
     end
 end

Folder = '';
File   = 'gray.jpg';
imwrite(out, fullfile(Folder, File));
```

```
imshow(out);
    end
end
```