Bilkent University

Department of Computer Engineering

EEE 391 - Basics of Signals and Systems
MATLAB Assignment 2

Name: Munib Emre Sevilgen
ID Number: 21602416
Section: 2

Part I:

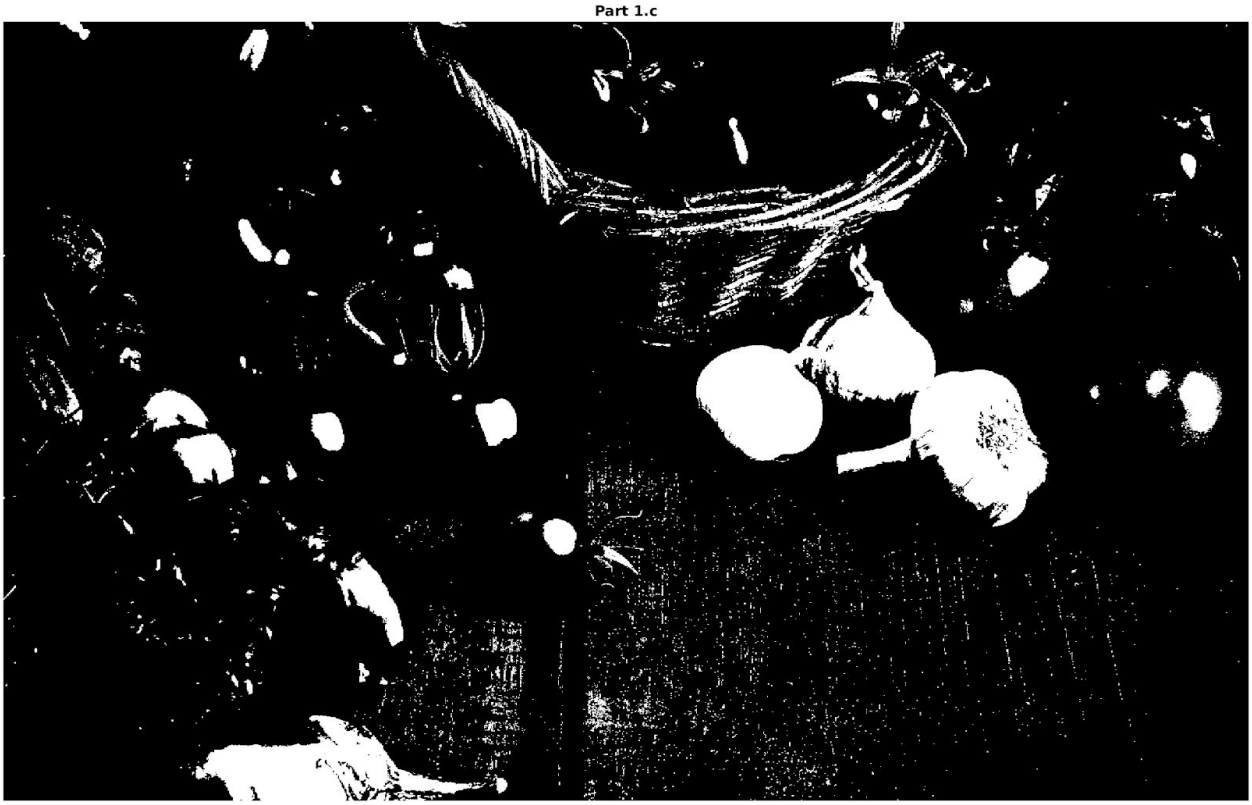
(a)

By the filter that is the image matrix elements have R values greater than 140, we make the pixels which have high red channel value white and others black. Therefore, we get the red and the bright colors. Because bright colors have high values at each channel red, green, and blue.

(b)

By the filter that is the image matrix elements has G values greater than 140, we make the pixels which have high green channel value white and others black. Therefore, we get green and bright colors. Because bright colors have high values at each channel red, green, and blue.

(c)



By the filter that is the image matrix elements has B values greater than 140, we make the pixels which have high blue channel value white and others black. Therefore, we get the bright colors. Because bright colors have high values at each channel red, green, and blue. However, there is not any pixel that has a blue color in the image.

(d)

By the filter that is the image matrix elements have R values greater than 140, G values greater than 140, and B values less than 30, we make the pixels which have high red and green but less blue channel values white and others black. Therefore, we get the yellow color which is the mixture of red and green and the bright green colors which are close to the yellow.

Part 1.e

Part II:

(i) The filter tries to reduce the noise at the image by looking for itself and the neighbor pixels. In this process, it calculates the mean of all the pixels in a matrix NxN which contains the current pixel in the middle of the matrix. By this method, the resulting image matrix will be in a way that the difference of the neighbor elements in the image matrix will be not much.

(ii) In the mean process, the difference between the neighbor elements in the image matrix is reduced. Therefore, the details in the image are also reduced. When the M increases, the neighbor matrix that the filter calculates the mean also increases. Therefore, there are more similar elements, so the loss of the details increases. The image will be more blurry.

(iii) There are not any differences between the dimensions. Because the filter applies the mean calculation to a square part of an image.

(iv) At the edges, the pixels are darker than the original image because we assume that the values lying outside of the image are zero. When M increases, the darker part also increases, because the size of the part that the filter calculates the mean is also increased.

(b)

(c)

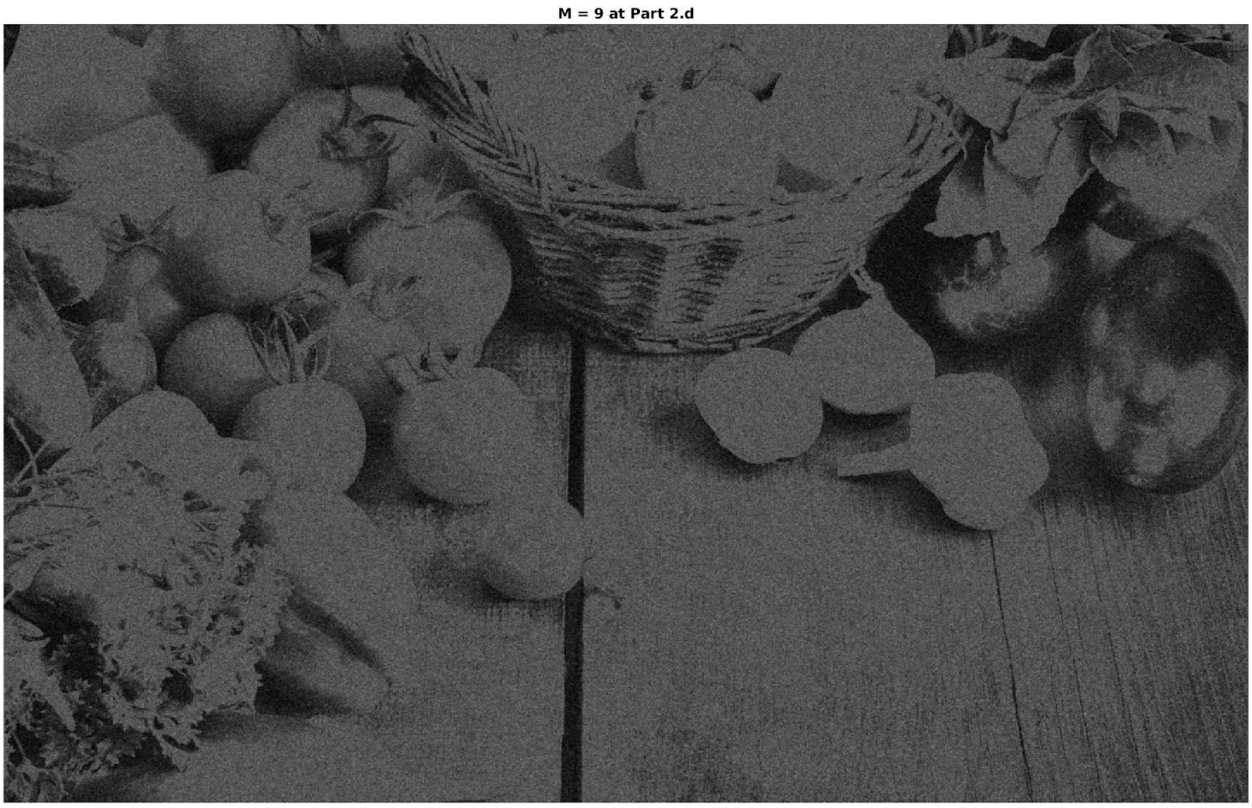
When we apply the mean filter with the value 9, the noise of the image is reduced but not totally avoided. The details of the image are much protected than the other values of the M. Moreover, with the value 25, the noise is much reduced and it is better than the image with the value 9. The details are also reasonable. However, with the value 121, the loss of details is very high and the image is blurry but the noise is completely reduced.

M = 25 at Part 2.c

M = 121 at Part 2.c

(d)

The noise level is much greater than the image in Part (b) with the Gaussian noise variance 1024. After that noise, even the M value 121 is not sufficient to reduce the noise completely.



M = 25 at Part 2.d

M = 121 at Part 2.d

(e)

This noise makes approximately 1/16 of the pixels completely white and approximately 1/16 of the pixels completely black randomly. The random matrix with the same size as the original image is created. The pixels of the original image at the indexes, which are specified by looking at the random matrix values, are changed. Therefore, we see some black and white pixels in the corrupted image. The black ones can be noticed at the brighter parts, and the white ones can be noticed at the darker parts of the image.

(f)

The mean filter can reduce the noise a little bit but not completely even with the M value 121. Because the calculating mean is not sufficient to get rid of the pixels that cause the noise.

M = 25 at Part 2.f

M = 121 at Part 2.f

Part III:

(i) The filter tries to reduce the noise at the image by looking for itself and the neighbor pixels. In this process, it calculates the median of all the pixels in a matrix NxN which contains the current pixel in the middle of the matrix. By this method, the resulting image can avoid the singular pixel noises like bit errors.

(ii) In the median process, the details are lost because of the mean calculation. It calculates the mean of a square NxN matrix which contains the current pixel in the middle, therefore it can miss the details at this matrix. Because it takes just one value which is the mean of the values. When M increases the size of the square matrix that the filter calculates mean by is increases, the loss of the details also increases.

(iii) There are not any differences between the dimensions. Because the filter applies the mean calculation to a square part of an image.

(iv) At the edges, the pixels are completely black because we assume that the values lying outside of the image are zero. When M increases, the black part also increases, because the size of the part that the filter calculates the mean is also increased.

(c)

When we apply the median filter to the image at Part II (b) with the value 9, the noise of the image is reduced but not totally avoided. The details of the image are much protected than the other values of the M. Moreover, with the value 25, the noise is much reduced and it is better than the image with the value 9. The details are also reasonable. However, with the value 121, the loss of details is very high and the image is blurry but the noise is completely reduced.

M = 9 at Part 3.c

M = 25 at Part 3.c

(d)

The noise level is much greater than the image in Part II (b) with the Gaussian noise variance 1024. After that noise, even the M value 121 is not sufficient to reduce the noise completely.

M = 9 at Part 3.d

M = 25 at Part 3.d

M = 121 at Part 3.d

(f)

The median filter can reduce most of the noise with the M value 9 at the image of Part II (e). With the M value 25, the noise is completely reduced and the loss of the details is reasonable. However, with the M value 121, the loss of the details is very large. The median filter is much better than the mean filter to reduce the salt and pepper noises.

M = 9 at Part 3.f

M = 25 at Part 3.f

M = 121 at Part 3.f

Part IV:

The vertical and horizontal edges are almost perfectly detected by the Sobel operator. However, if there are shadows at the edges, the sharp difference of the edges is reduced, so they are not detected as can be seen at the top right of the image. Moreover, the surface patterns of the wood are also detected as an edge because there are also sharp differences.

G

Codes:

Part I:

```
clc; clear all; close all;

% Get the image file
id = 21602416;
rem = mod(id, 4);
fname = ['image',num2str(rem),'.jpg'];
A = imread(fname);

% Part 1.a
n = size(A, 1);
m = size(A, 2);
Y = zeros(n, m, 'logical');

for row = 1:n
  for col = 1:m
    if A(row, col, 1) > 140
      Y(row, col) = 1;
    end
  end
end

figure;
imshow(Y);
title('Part 1.a');
drawnow;

% Part 1.b
n = size(A, 1);
m = size(A, 2);
Y = zeros(n, m, 'logical');

for row = 1:n
  for col = 1:m
    if A(row, col, 2) > 140
      Y(row, col) = 1;
    end
  end
end
```

```
figure;
imshow(Y);
title('Part 1.b');
drawnow;

% Part 1.c
n = size(A, 1);
m = size(A, 2);
Y = zeros(n, m, 'logical');

for row = 1:n
  for col = 1:m
     if A(row, col, 3) > 140
        Y(row, col) = 1;
     end
  end
end

figure;
imshow(Y);
title('Part 1.c');
drawnow;

% Part 1.d
n = size(A, 1);
m = size(A, 2);
Y = zeros(n, m, 'logical');

for row = 1:n
  for col = 1:m
     if A(row, col, 1) > 140 && A(row, col, 2) > 140 && A(row, col, 3) < 30
        Y(row, col) = 1;
     end
  end
end

figure;
imshow(Y);
title('Part 1.d');
drawnow;

%Part 1.e
n = size(A, 1);
```

```
m = size(A, 2);
Y = zeros(n, m, 'uint8');

for row = 1:n
  for col = 1:m
     Y(row, col) = (A(row, col, 1) + A(row, col, 2) + A(row, col, 3))/3;
  end
end

figure;
imshow(Y, [0 255]);
title('Part 1.e');
drawnow;
imwrite(Y, 'gray.jpg');
```

Part II:

```
clc; clear all; close all;

% Get the image file
imgray = imread('gray.jpg');
n = size(imgray, 1);
m = size(imgray, 2);

% Part 2.b
gaussnoise = 8*randn(n, m);
imgaussnoise = uint8(double(imgray) + gaussnoise);
figure;
imshow(imgaussnoise, [0 255]);
title('Corrupted image at Part 2.b');
drawnow;

% Part 2.c
Ms = [9 25 121];
displayForMs(Ms, imgaussnoise, '2.c');

% Part 2.d
gaussnoise = 32*randn(n, m);
imgaussnoise = uint8(double(imgray) + gaussnoise);
figure;
imshow(imgaussnoise, [0 255]);
title('Corrupted image at Part 2.d');
drawnow;
```

```
Ms = [9 25 121];
displayForMs(Ms, imgaussnoise, '2.d');

% Part 2.e
imsaltnoise = imgray;
noisypixels = rand( size(imgray,1), size(imgray,2) );
imsaltnoise(noisypixels <= (1 / 16)) = 255;
imsaltnoise(noisypixels >= (15 / 16)) = 0;
figure;
imshow(imsaltnoise, [0 255]);
title('Salt and pepper type noise at Part 2.e');
drawnow;

% Part 2.f
Ms = [9 25 121];
displayForMs(Ms, imsaltnoise, '2.f');


function displayForMs(Ms, image, part)
   for i = 1: size(Ms, 2)
      J = meanfilter(Ms(1, i), image);
      figure;
      imshow(J, [0 255]);
      tit_str = ['M = ', num2str(Ms(1, i)), ' at Part ', part];
      title(tit_str);
      drawnow;
   end
end

% Part 2.a
function J = meanfilter(M, I)
   row_size = size(I, 1);
   col_size = size(I, 2);
   N = (sqrt(M) - 1)/2;
   J = zeros(row_size, col_size, 'uint8');
   for n = 1:row_size
      for m = 1:col_size
         sum = 0;
         for i = n-N:n+N
            if i >= 1 && i <= row_size
               for j = m-N:m+N
                  if j >= 1 && j <= col_size
                     sum = sum + uint32(I(i, j));
```

```
                end
            end
        end
    end
    J(n,m) = sum/M;
  end
 end
end
```

Part III:

```
clc; clear all; close all;

% Get the image file
imgray = imread('gray.jpg');
n = size(imgray, 1);
m = size(imgray, 2);

% Part 3.b
gaussnoise = 8*randn(n, m);
imgaussnoise = uint8(double(imgray) + gaussnoise);
figure;
imshow(imgaussnoise, [0 255]);
title('Corrupted image at Part 3.b');
drawnow;

% Part 3.c
Ms = [9 25 121];
displayForMs(Ms, imgaussnoise, '3.c');

% Part 3.d
gaussnoise = 32*randn(n, m);
imgaussnoise = uint8(double(imgray) + gaussnoise);
figure;
imshow(imgaussnoise, [0 255]);
title('Corrupted image at Part 3.d');
drawnow;

Ms = [9 25 121];
displayForMs(Ms, imgaussnoise, '3.d');

% Part 3.e
imsaltnoise = imgray;
noisypixels = rand( size(imgray,1), size(imgray,2) );
```

```
imsaltnoise(noisypixels <= (1 / 16)) = 255;
imsaltnoise(noisypixels >= (15 / 16)) = 0;
figure;
imshow(imsaltnoise, [0 255]);
title('Salt and pepper type noise at Part 3.e');
drawnow;

% Part 3.f
Ms = [9 25 121];
displayForMs(Ms, imsaltnoise, '3.f');

function displayForMs(Ms, image, part)
   for i = 1: size(Ms, 2)
      J = medianfilter(Ms(1, i), image);
      figure;
      imshow(J, [0 255]);
      tit_str = ['M = ', num2str(Ms(1, i)), ' at Part ', part];
      title(tit_str);
      drawnow;
   end
end

function J = medianfilter(M, I)
   row_size = size(I, 1);
   col_size = size(I, 2);
   N = (sqrt(M) - 1)/2;
   J = zeros(row_size, col_size, 'uint8');

   paddedImg = zeros(row_size + 2*N, col_size + 2*N);
   paddedImg(N+1:N+row_size, N+1:N+col_size) = I;
   for n = 1:row_size
      for m = 1:col_size
         med = median(paddedImg(n:(n+N+N), m:(m+N+N)), 'all');
         J(n,m) = med;
      end
   end
end
```

Part IV:

```
clc; clear all; close all;

% Get the image file
id = 21602416;
```

```
rem = mod(id, 4);
fname = ['image',num2str(rem),'part4.jpg'];
I = imread(fname);

f = [1 0 -1; 2 0 -2; 1 0 -1];
for i = 1:3
    Gx = conv2(f, I(:,:,i));
    Gy = conv2(f', I(:,:,i));
    G(:,:,i) = uint8(sqrt(Gx.*Gx + Gy.*Gy));
end

figure;
imshow(I);
title('I');
drawnow;

figure;
imshow(G);
title('G');
drawnow;
```