**EE 391**

**Homework 2**

**Mert Özerdem**

**ID: 21300835**

**Section: 03**

# Q1)

figure 1: original picture

## a)

From the figure 2 below we can see that some pixels are white and some are black. White pixels represents where contrast of red color is higher than 140. White pixels is not affected by green and blue color values. So we can say that no matter the color of the pixel in the original picture if value(contrast) of red in the original pixel is higher than 140 we color as white(1), otherwise to black (0). So picture below indicates where red color value of pixels is high.And by looking at the face of the animal in the original picture we can say white color of an original image does have higher contrast of red value.



figure 2: white and black where red value is higher than 140

## b)

From the figure 3 below we can see that some pixels are white and some are black. White pixels represents where contrast of green color is higher than 140. White pixels is not affected by red and blue color values. So we can say that no matter the color of the pixel in the original picture if value(contrast) of green in the original pixel is higher than 140 we color as white(1), otherwise to black (0). So picture below indicates where green color value of pixels is high. And by looking at the face of the animal in the original picture we can say white color of an original image does have higher contrast of green value.



figure 3: white and black where green value is higher than 140

## c)

From the figure 4 below we can see that some pixels are white and some are black. White pixels represents where contrast of blue color is higher than 140. White pixels is not affected by red and green color values. So we can say that no matter the color of the pixel in the original picture if value(contrast) of blue in the original pixel is higher than 140 we color as white(1), otherwise to black (0). So picture below indicates where blue color value of pixels is high.And by looking at the face of the animal in the original picture we can say white color of an original image does have higher contrast of blue value.



figure 4: white and black where blue value is higher than 140

## d)

From the figure 5 below we can see that some pixels are white and some are black. White pixels represents where contrast of red color is higher than 140, contrast of green color is higher than 140, and contrast of blue color is less than 30. So we can say that if pixel of the original picture validates conditions above it is painted white(1) and black(0) otherwise.So below picture indicates combination of colors which seems like color yellow.



figure 5: red >140, green > 140, and blue < 30 shows yellow parts

## e)

Below figure 6 is generated by taking average value of red, green, blue value of individual pixels. We can see that from picture below averaging colors gave us gray version of the original picture.



figure 6: (red+green+blue)/3 averaged image

# Q2)

## a)

The mean filter takes an M value form the user, then this filter uses it to calculate the mean value of the pixel with the N neighbors. This way filter smoothens the picture. [see appendix 2.a]

## b)

Function corrupts the image with variance 64.

figure 7: Gauss noised image with variance 64

## c)

This filter smoothes the image and gives it a blurry effect. The blur effect on the images increased as M increased. Nevertheless, images were smoothened by the filter when we increased the value of M. This happens because the mean value is assigned to the pixel with its MxM neighbor. MxM matrix/N for each pixel of the original image. M= 9 and M= 25 fixed noise and smoothened the image but M=121 broke the picture and blurred unnecessarily. Details in the image also faded away while we are increasing M value. Edges of the images get smoother and blend with the background. this is happening because all pixels get closer to each other while we increase M value. There is no difference in vertical and horizontal dimensions, each dimension effected the same from the filtering. Visually each dimension affected with the same type of visual smoothening.

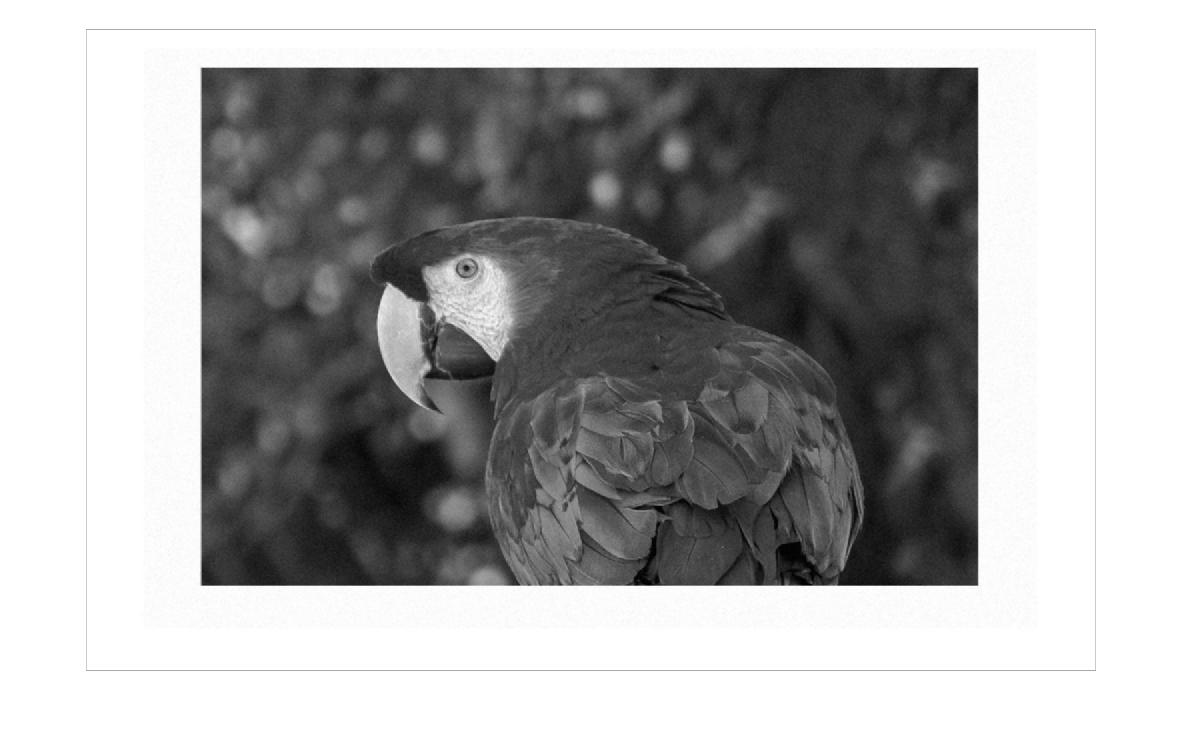
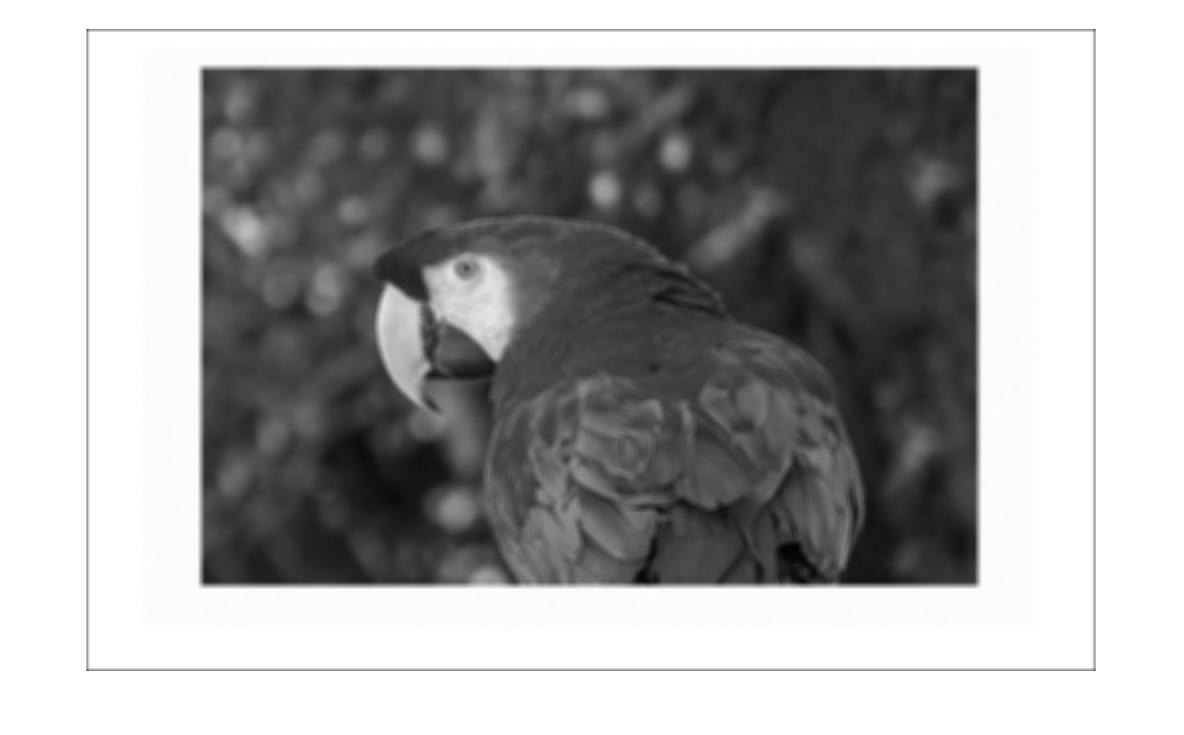


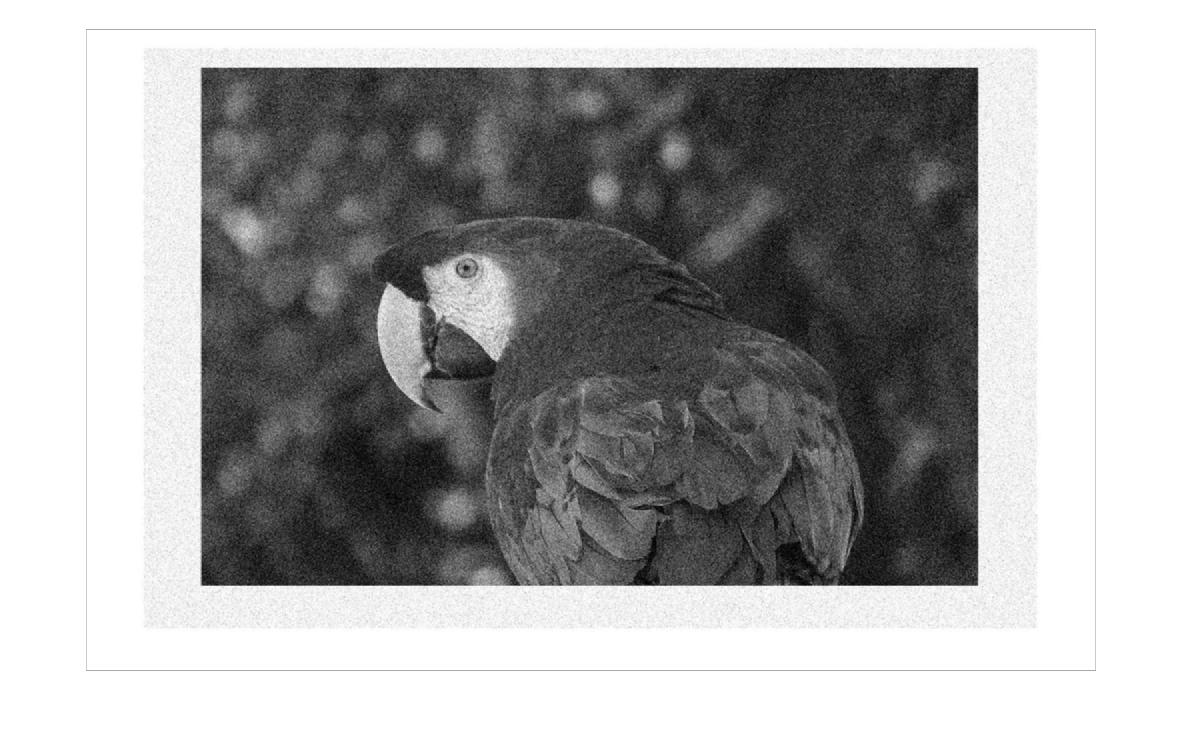
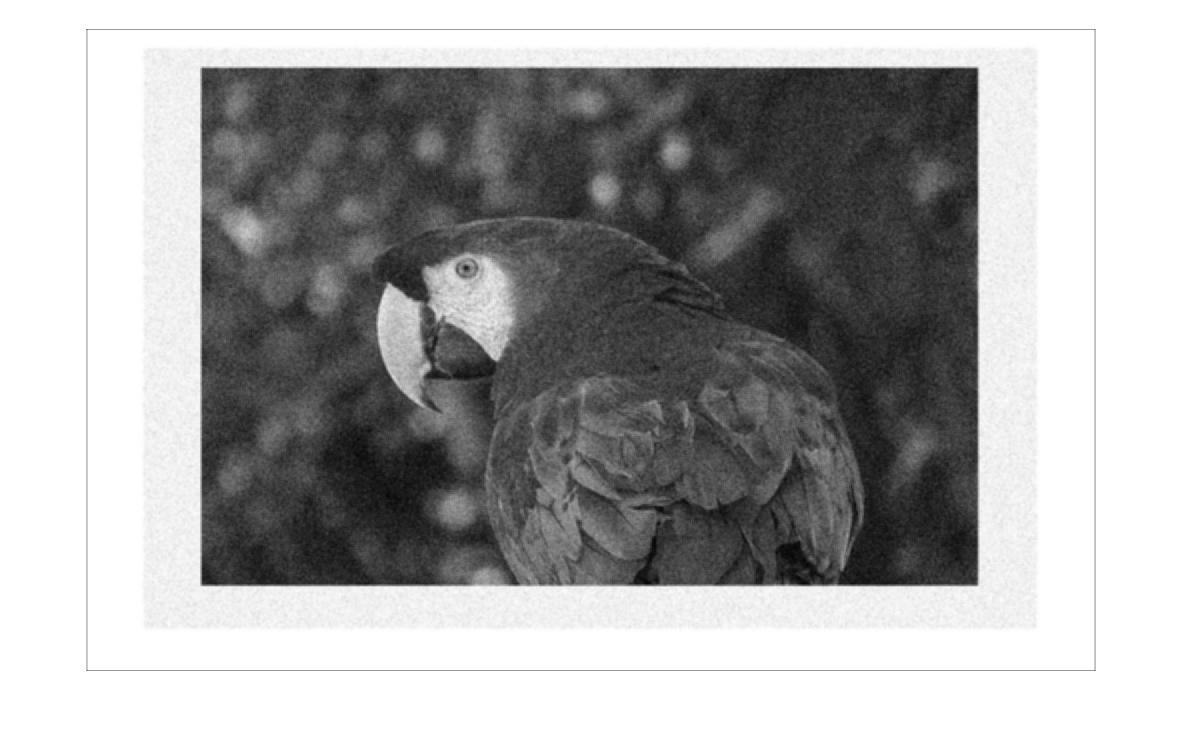
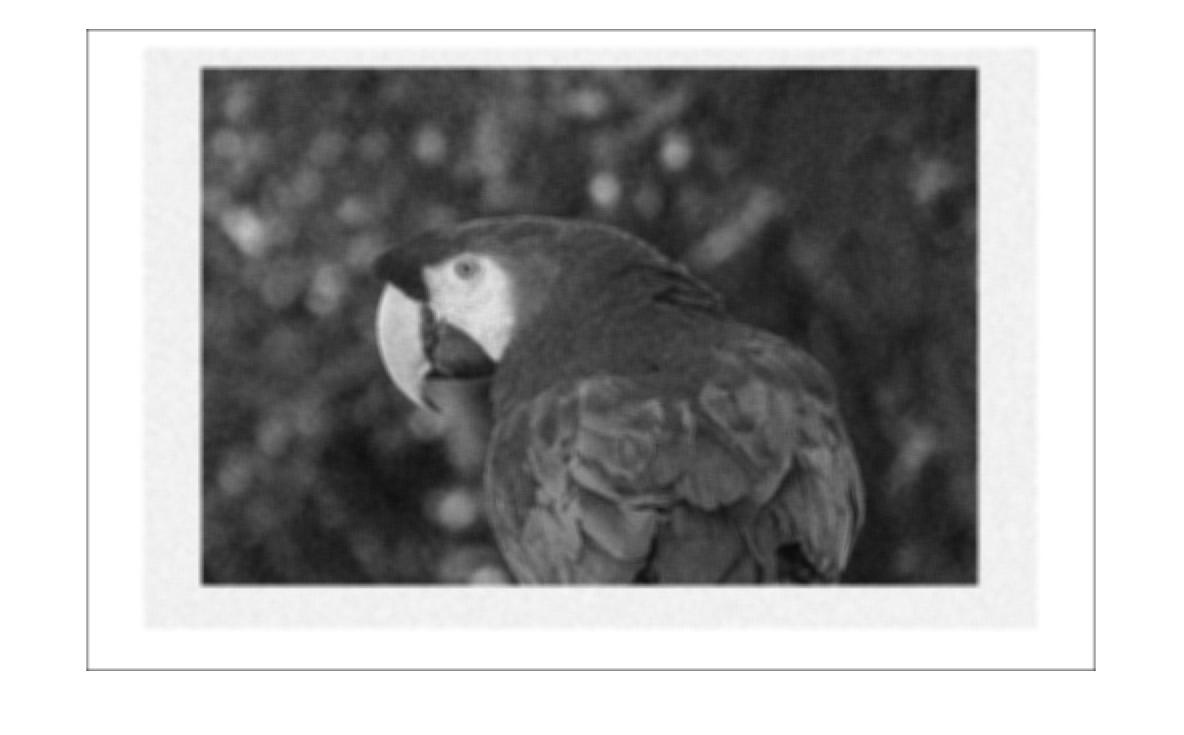
figure 8: Mean filter with M = 9

figure 9: Mean filter with M = 25figure 9: Mean filter with M = 121

## d)

This is basically the same thing but with more corrupted image. So everything i said about part c is valid in here. This filter smoothes the image and gives it a blurry effect. The blur effect on the images increased as M increased. Nevertheless, images were smoothened by the filter when we increased the value of M. This happens because the mean value is assigned to the pixel with its MxM neighbor. MxM matrix/N for each pixel of the original image. M= 9 and M= 25 fixed noise and smoothened the image but M=121 broke the picture and blurred unnecessarily. Details in the image also faded away while we are increasing M value. Edges of the images get smoother and blend with the background. this is happening because all pixels get closer to each other while we increase M value. There is no difference in vertical and horizontal dimensions, each dimension effected the same from the filtering. Visually each dimension affected with the same type of visual smoothening.

figure 10: variance 1024 gauss noised image

figure 11: Mean filter with M = 9 to variance 1024 gauss noised imagefigure 12: Mean filter with M = 25 to variance 1024 gauss noised imagefigure 13: Mean filter with M = 121 to variance 1024 gauss noised image

## e)

Adding salt noise to image make it look like an old picture, which get corrupted in time. This noise adds 0 and 255 valued pixels to the image with a random chance. so image is corrupted by hand with the addition of white and black pixels. Each dimension affected same. Edges affected as same as any part of the image.

figure 14: salt noised image

## f)

Mean filtering did not give great result as it did in gauss noised image. Image is again smoothened but filter could not managed to get rid of the effects created by salt noise. So overall quality of the image decreased much faster. This filter smoothes the image and gives it a blurry effect. The blur effect on the images increased as M increased. Nevertheless, images were smoothened by the filter when we increased the value of M. This happens because the mean value is assigned to the pixel with its MxM neighbor. MxM matrix/N for each pixel of the original image. M= 9 and M= 25 fixed noise and smoothened the image but M=121 broke the picture and blurred unnecessarily. Details in the image also faded away while we are increasing M value. Edges of the images get smoother and blend with the background. this is happening because all pixels get closer to each other while we increase M value. There is no difference in vertical and horizontal dimensions, each dimension effected the same from the filtering. Visually each dimension affected with the same type of visual smoothening.

figure 15: salt noised image,mean filter M=9figure 16: salt noised image,mean filter M=25figure 17: salt noised image,mean filter M=121

# Q3)

## a)

[see appendix 3.a]

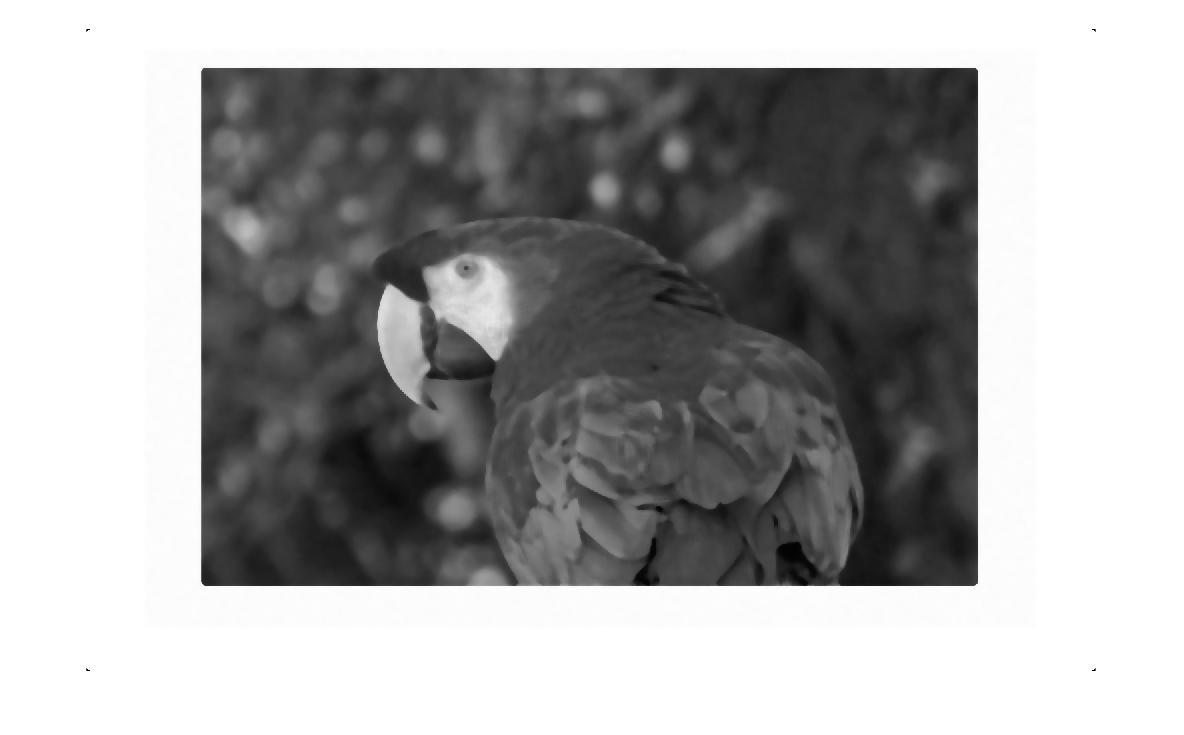
## b)

same as part 2-bfigure 18 :

## c)

fil

figure 19:

figure 20:figure 21:

## d)

asdasdfigure 22: variance 1024 gauss noised imagefigure 23: figure 24:figure 25:

## e)

same as part 2-e

figure 26:

## f)

sasa

figure 27: figure 28: figure 29:

# Q4)

Sobel operation detects edges of the images by using matrix multiplication where 3x3 matrix with minus and plus sides is used. This matrix multiplication gives higher values when image shows a change in the pixel value composition. this way edges, where values of the pixels changes and creates uneven matrix, can be detected and scored higher. I did sobel filtering in this part of the homework and found edges where value of the pixels differ. From below figures you can see that sobel filter detected the edges of the original image and edges marked with white color while rest is black. I also made only red channel image and filtered it with sobel filter to eliminate excess red, green,blue pixels.

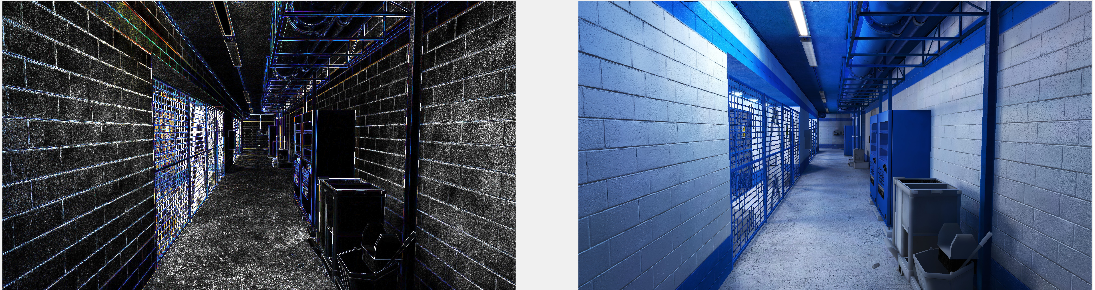
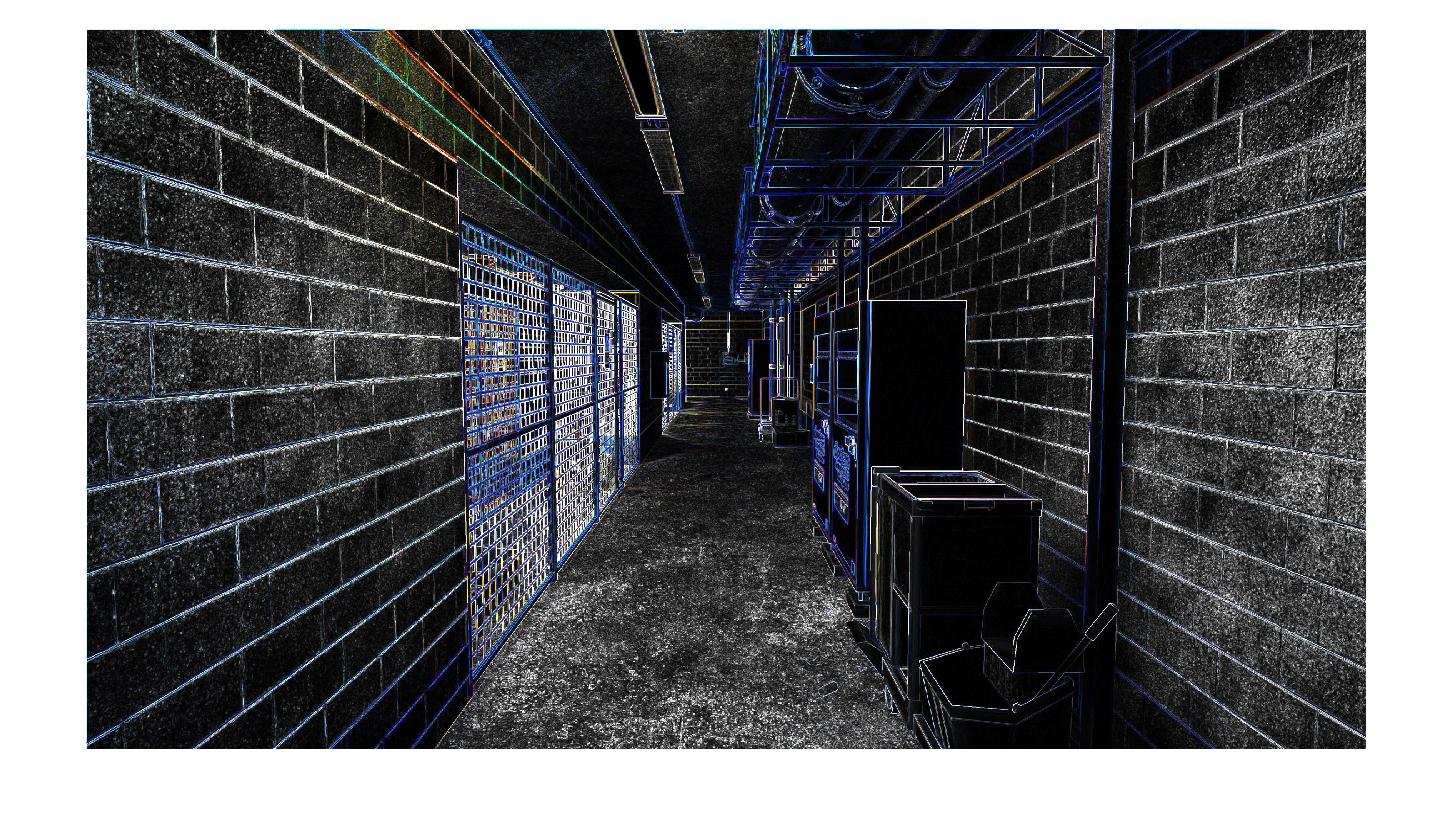


figure : comparison image



figure : Original image

figure : filtered imagefigure: only red channel image

# Appendix

## 1.a red contrast > 140

A=imread('image3.jpg');

%%onlyreds = A(:, :, 1);

%imshow(redChannel)

B = zeros(1554,2331);

for i = 1:2331

for j = 1:1554

if A(j,i,1) > 140

B(j,i) = 1;

%disp('job done')

end

end

end

C=B;

imshow(B)

## 1.b green contrast > 140

A=imread('image3.jpg');

%%onlyreds = A(:, :, 1);

%imshow(redChannel)

B = zeros(1554,2331);

for i = 1:2331

for j = 1:1554

if A(j,i,2) > 140

B(j,i) = 1;

%disp('job done')

end

end

end

C=B;

imshow(B)

## 1.c blue contrast > 140

A=imread('image3.jpg');

%%onlyreds = A(:, :, 1);

%imshow(redChannel)

B = zeros(1554,2331);

for i = 1:2331

for j = 1:1554

if A(j,i,3) > 140

B(j,i) = 1;

%disp('job done')

end

end

end

C=B;

imshow(B)

## 1.d combination

A=imread('image3.jpg');

%%onlyreds = A(:, :, 1);

%imshow(redChannel)

B = zeros(1554,2331);

for i = 1:2331

for j = 1:1554

if A(j,i,1) > 140 && A(j,i,2) > 140 && A(j,i,3) < 30

B(j,i) = 1;

%disp('job done')

end

end

end

C=B;

imshow(B)

## 1.e Gray image

A=imread('image3.jpg');

[a,b,c] = size(A);

for i = 1:a

for j = 1:b

d = double(A(i,j,1))+ double(A(i,j,2)) + double(A(i,j,3));

B(i,j) = double(d)/3;

end

end

imshow(B)

## 2.a Mean filter Function

function [meanFilteredImage] = meanFilter(I,M)

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

[i,j]=size(I);

temp = 0;

for n = 1 : i

for m = 1 : j

for a = (n-N) : (n+N)

for b = (m-N) : (m+N)

if a < 1 || b < 1 || a > i || b > j

temp = 0 + temp;

else

temp = I(a,b) + temp;

end

end

end

I(n,m) = temp\*(1/M);

temp = 0;

end

end

meanFilteredImage = I;

end

## 2.b Gauss noise with variance 64

A=imread('gray.jpg');

imgray = A(:,:,1);

gaussnoise = 8\*randn(size(imgray,1), size(imgray,2));

imgaussnoise = uint8(double(imgray) + gaussnoise);

imshow(imgaussnoise)

## 2.c.1 Mean filter with M=9

A=imread('imgaussnoise.jpg');

a = A(:, :, 1);

C = meanFilter(double(a),9);

imshow(uint8(C))

## 2.c.1 Mean filter with M=25

A=imread('imgaussnoise.jpg');

a = A(:, :, 1);

C = meanFilter(double(a),25);

imshow(uint8(C))

## 2.c.1 Mean filter with M=121

A=imread('imgaussnoise.jpg');

a = A(:, :, 1);

C = meanFilter(double(a),121);

imshow(uint8(C))

## 2.d.1 variance of gauss noise is changed to 1024

A=imread('gray.jpg');

imgray = A(:,:,1);

gaussnoise = 32\*randn(size(imgray,1), size(imgray,2));

imgaussnoise = uint8(double(imgray) + gaussnoise);

imshow(imgaussnoise)

## 2.d.2 Mean filter with M=9

A=imread('imgaussnoiseVariance1024.jpg');

a = A(:, :, 1);

C = meanFilter(double(a),9);

imshow(uint8(C))

## 2.d.3 Mean filter with M=25

A=imread('imgaussnoiseVariance1024.jpg');

a = A(:, :, 1);

C = meanFilter(double(a),25);

imshow(uint8(C))

## 2.d.4 Mean filter with M=121

A=imread('imgaussnoiseVariance1024.jpg');

a = A(:, :, 1);

C = meanFilter(double(a),121);

imshow(uint8(C))

## 2.e salt noise image

A=imread('gray.jpg');

imgray = A(:, :, 1);

imsaltnoise = imgray;

noisypixels = rand( size(imgray,1), size(imgray,2) );

imsaltnoise( find( noisypixels <= ( 1 / 16 ) ) ) = 255;

imsaltnoise( find( noisypixels >= ( 15 / 16 ) ) ) = 0;

%C = meanFilter(double(imsaltnoise),121);

imshow(imsaltnoise)

## 2.f.1 salt noise image mean filter M=9

A=imread('imsaltnoise.jpg');

a = A(:, :, 1);

C = meanFilter(double(a),9);

imshow(uint8(C))

## 2.f.1 salt noise image mean filter M=25

A=imread('imsaltnoise.jpg');

a = A(:, :, 1);

C = meanFilter(double(a),25);

imshow(uint8(C))

## 2.f.1 salt noise image mean filter M=121

A=imread('imsaltnoise.jpg');

a = A(:, :, 1);

C = meanFilter(double(a),121);

imshow(uint8(C))

## 3.a Median filter

function [medianFilteredImage] = medianFilter(I,M)

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

[i,j]=size(I);

for n = 1 : i

for m = 1 : j

arrayLength = 1;

for a = (n-N) : (n+N)

for b = (m-N) : (m+N)

if a < 1 || b < 1 || a > i || b > j

Arr(arrayLength) = 0;

arrayLength = arrayLength + 1;

else

Arr(arrayLength) = I(a,b);

arrayLength = arrayLength + 1;

end

end

end

sortedArr = sort(Arr);

medianValue = sortedArr(round(M/2));

ABC(n,m) = medianValue;

end

end

medianFilteredImage = ABC;

end

## 3.b Gauss noise with variance 64

A=imread('gray.jpg');

imgray = A(:,:,1);

gaussnoise = 8\*randn(size(imgray,1), size(imgray,2));

imgaussnoise = uint8(double(imgray) + gaussnoise);

imshow(imgaussnoise)

## 3.c.1 Median filter M = 9

A=imread('imgaussnoise.jpg');

a = A(:, :, 1);

C = medianFilter(a,9);

imshow(uint8(C))

## 3.c.2 Median filter M = 25

A=imread('imgaussnoise.jpg');

a = A(:, :, 1);

C = medianFilter(a,25);

imshow(uint8(C))

## 3.c.3 Median filter M = 121

A=imread('imgaussnoise.jpg');

a = A(:, :, 1);

C = medianFilter(a,121);

imshow(uint8(C))

## 3.d.1 variance of gauss noise is changed to 1024

A=imread('gray.jpg');

imgray = A(:,:,1);

gaussnoise = 32\*randn(size(imgray,1), size(imgray,2));

imgaussnoise = uint8(double(imgray) + gaussnoise);

imshow(imgaussnoise)

## 3.d.2 Median filter with M=9

A=imread('imgaussnoiseVariance1024.jpg');

a = A(:, :, 1);

C = medianFilter(a,9);

imshow(uint8(C))

## 3.d.3 Median filter with M=25

A=imread('imgaussnoiseVariance1024.jpg');

a = A(:, :, 1);

C = medianFilter(a,25);

imshow(uint8(C))

## 3.d.4 Median filter with M=121

A=imread('imgaussnoiseVariance1024.jpg');

a = A(:, :, 1);

C = medianFilter(a,121);

imshow(uint8(C))

## 3.e salt noise image

A=imread('gray.jpg');

imgray = A(:, :, 1);

imsaltnoise = imgray;

noisypixels = rand( size(imgray,1), size(imgray,2) );

imsaltnoise( find( noisypixels <= ( 1 / 16 ) ) ) = 255;

imsaltnoise( find( noisypixels >= ( 15 / 16 ) ) ) = 0;

%C = meanFilter(double(imsaltnoise),121);

imshow(imsaltnoise)

## 3.f.1 salt noise image median filter M=9

A=imread('imsaltnoise.jpg');

a = A(:, :, 1);

C = medianFilter(a,9);

imshow(uint8(C))

## 3.f.2 salt noise image median filter M=25

A=imread('imsaltnoise.jpg');

a = A(:, :, 1);

C = medianFilter(a,25);

imshow(uint8(C))

## 3.f.3 salt noise image median filter M=121

A=imread('imsaltnoise.jpg');

a = A(:, :, 1);

C = medianFilter(a,121);

imshow(uint8(C))

## 4 Sobel Filter

A=imread('image3part4.jpg');

[a,b,c] = size(A);

ARed= A(:, :, 1);

AGreen = A(:, :, 2);

ABlue = A(:, :, 3);

preGx = [1,0,-1;2,0,-2;1,0,-1];

preGy = [1,2,1;0,0,0;-1,-2,-1];

GxRed = conv2(ARed,preGx);

GxGrenn = conv2(AGreen,preGx);

GxBlue = conv2(ABlue,preGx);

GyRed = conv2(ARed,preGy);

GyGreen = conv2(AGreen,preGy);

GyBlue = conv2(ABlue,preGy);

GRed = sqrt(GxRed.^2 + GyRed.^2);

GGreen = sqrt(GxGrenn.^2 + GyGreen.^2);

GBlue = sqrt(GxBlue.^2 + GyBlue.^2);

G = zeros(a,b,c);

for i = 1:a

for j = 1:b

G(i,j,1) = GRed(i,j);

G(i,j,2) = GGreen(i,j);

G(i,j,3) = GBlue(i,j);

end

end

%figure;

%subplot('Position',[0.2 0.35 0.30 0.30]);

imshow(uint8(G));

%subplot('Position',[0.5 0.35 0.30 0.30]);

%imshow(A);