Digital Image Processing
Project 2

Omar Rawashdeh
Harshal Raut
Utsav Shah

September 27, 2016

Contents

1 Introduction **4**
1.1 Image Enhancement 4

2 Problem 1 **5**
2.1 Discussion and Objective 5
2.2 Filter and Parameter Used 5
2.3 Results and Remarks 6
2.4 MATLAB Source Code 7

3 Problem 2 **9**
3.1 Discussion and Objective 9
3.2 Filter and Parameter Used 9
3.3 Results and Remarks 10
3.4 MATLAB Source Code 12

4 Problem 3 **14**
4.1 Discussion and Objective 14
4.2 Filter and Parameter Used 14
4.3 Results and Remarks 15
4.4 MATLAB Source Code 19

5 Problem 4 **20**
5.1 Discussion and Objective 20
5.2 Filter and Parameter Used 20
5.3 Results and Remarks 21
5.4 MATLAB Source Code 25

6 Problem 5 **26**
6.1 Discussion and Objective 26
6.2 Filter and Parameter Used 26
6.3 Results and Remarks 27

7 Main Source Code **28**

List of Figures

1 Original Image 6
2 Gray Scale Logarithmic Transform Image 7
3 Blurred Image 10
4 Cleared Image Using Median Filter 11
5 Edge Extraction Noisy Image Without Scaling 15
6 Edge Extraction Noisy Image With Scaling 16
7 Edge Extraction No Noise Image Without Scaling 17
8 Edge Extraction No Noise Image With Scaling 18
9 Sharpened Gray Image using Sharp Mask 21
10 Sharpened Colored Image using Sharp Mask 22
11 Sharpened Gray Image using Unsharp Mask 23
12 Sharpened Colored Image using Unsharp Mask 24
13 Applying Two Sharpen Masks in Sequence 27
14 Applying One Sharpen Mask 28

1 Introduction

1.1 Image Enhancement

Image Enhancement falls into two categories:

- Enhancement in Spatial Domain: The term spatial domain refers to the Image Plane itself which is direct manipulation of pixels.

- Enhancement in Frequency Domain: Frequency Domain processing techniques are based on modifying the Fourier transform of an image.

The value of a pixel with coordinates (x,y) in the enhanced image is the result of performing some operation on the pixels in the neighbourhood of (x,y) in the input image, F. Neighbourhoods can be any shape, but usually they are rectangular.[1]

The term Spatial Domain refers to the aggregate of the pixels composing an image(1). The Spatial Domain methods are the procedures that operate directly on these pixels.

It is denoted by

$$g(x, y) = T[f(x, y)]$$

where

$$g(x, y) : ProcessedImage; T : OperatoronImage; f(x, y) : InputImage$$

2 Problem 1

Select a gray level transformation method and apply it to images and discuss your finding.

2.1 Discussion and Objective

The visual appearance of an image is generally characterized by two properties: brightness and contrast. Brightness refers to the overall intensity level and is therefore inuenced by the individual gray-level (intensity) values of all the pixels within an image. Since a bright image (or sub-image) has more pixel gray-level values closer to the higher end of the intensity scale, it is likely to have a higher average intensity value. Contrast in an image is indicated by the ability of the observer to distinguish separate neighboring parts within an image. This ability to see small details around an individual pixel and larger variations within a neighborhood is provided by the spatial intensity variations of adjacent pixels, between two neighboring sub-images, or within the entire image. Thus, an image may be bright (due to, for example, overexposure or too much illumination) with poor contrast if the individual target objects in the image have optical characteristics similar to the background. At the other end of the scale, a dark image may have high contrast if the background is signicantly different from the individual objects within the image, or if separate areas within the image have very different reectance properties.[2]

2.2 Filter and Parameter Used

A function is written to Logarthmic Transformation on the Grayscale Image.

2.3 Results and Remarks

Figure 1: Original Image

Figure 2: Gray Scale Logarithmic Transform Image

2.4 MATLAB Source Code

The following function was written to apply logarthmic transform

$$GrayLevelLogarithmicTransform$$

```
function OutputImage = GrayLevelTransform_Log(InputImage)
%GRAYLEVELTRANSFORM_LOG Summary of this function goes
    here
%   Detailed explanation goes here
    NewImageMat = zeros(size(InputImage, 1), size(
        InputImage, 2));
    c = 255/log(256);
    for j=1:size(InputImage, 2);
        for i=1:size(InputImage, 1)
            NewImageMat(i, j) = c*log(1 + double(
                InputImage(i, j)));
        end
```

```
    end
    DoubleNewImage = mat2gray(NewImageMat, [0 255]);
    OutputImage = im2uint8(DoubleNewImage);
end
```

3 Problem 2

Select at least two spatial filters and apply them to noisy images and discuss your results.

3.1 Discussion and Objective

In blurring, an image is simply blurred using a low pass filter. An image looks more sharp or more detailed if humans are able to perceive all the objects and their shapes correctly in it. For example, an image with a face, looks clear when humans are able to identify eyes, ears nose, lips, forehead, etc. very clear. This shape of an object is due to its edges. So in blurring we simply reduce the edge content and makes the transition form one color to the other very smooth.[3]

Blurring can be achieved by many ways. The common type of filters that are used to perform blurring are.

- Mean filter
- Weighted average filter
- Gaussian filter

3.2 Filter and Parameter Used

Two methods used:

- Blurring image using low pass filter
- Clearing image using median filter to remove salt and pepper noise

$$\text{Blurring Mask}= \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

3.3 Results and Remarks

Figure 3: Blurred Image

Figure 4: Cleared Image Using Median Filter

3.4 MATLAB Source Code

The following function was written to blur image

SpatialFilter2BlurImage

```matlab
function BlurredImage = SpatialFilter2_BlurImage(Image, Mask)
%BLURIMAGE Summary of this function goes here
%   Detailed explanation goes here
    if (mod(size(Mask, 1), 2) == 0) || (mod(size(Mask, 2), 2) == 0)
        error('it is recommended to use a mask with odd width and height');
    end

    SumOfWeights = sum(Mask(:));
    NewMask = Mask/SumOfWeights;

    m = (size(Mask, 2)-1)/2;
    n = (size(Mask, 1)-1)/2;

    BlurredImageMat = Image;
    for j=m+1:size(Image, 2)-m
        for i=n+1:size(Image, 1)-n
            Values_X_Weights = double(ReadImagePart(Image, i, j, n, m)).*NewMask;
            BlurredImageMat(i, j) = fix(sum(Values_X_Weights(:)));
        end
    end

    DoubleBlurredImage = mat2gray(BlurredImageMat, [0 255]);
    BlurredImage = im2uint8(DoubleBlurredImage);
end
```

The following function was written to remove noise

$$SpatialFilter1RemoveNoiseStatistical$$

```
function ClearedImage = SpatialFilter1_RemoveNoise_Statistical(NoiseImage, n)
%REMOVENOISE Summary of this function goes here
%   Detailed explanation goes here
    ClearedImage = NoiseImage;
    for j=n+1:size(NoiseImage, 2)-n
        for i=n+1:size(NoiseImage, 1)-n
            Values = ReadImagePart(NoiseImage, i, j, n, n);
            ClearedImage(i, j) = median(Values(:));
        end
    end
end
```

4 Problem 3

Select two edge extraction methods and apply them to both noise and no noise image and discuss your result.

4.1 Discussion and Objective

A high-pass filter can be used to make an image appear sharper. These filters emphasize fine details in the image – exactly the opposite of the low-pass filter. High-pass filtering works in exactly the same way as low-pass filtering; it just uses a different convolution kernel. In the example below, notice the minus signs for the adjacent pixels. If there is no change in intensity, nothing happens. But if one pixel is brighter than its immediate neighbors, it gets boosted.[4]

Unfortunately, while low-pass filtering smooths out noise, high-pass filtering does just the opposite: it amplifies noise. You can get away with this if the original image is not too noisy; otherwise the noise will overwhelm the image. High-pass filtering can also cause small, faint details to be greatly exaggerated. An over-processed image will look grainy and unnatural, and point sources will have dark donuts around them. So while high-pass filtering can often improve an image by sharpening detail, overdoing it can actually degrade the image quality significantly.

4.2 Filter and Parameter Used

$$\mathrm{M} = \begin{bmatrix} 0 & -1/4 & 0 \\ 2 & -1/4 & -1/4 \\ 0 & -1/4 & 0 \end{bmatrix}$$

4.3 Results and Remarks

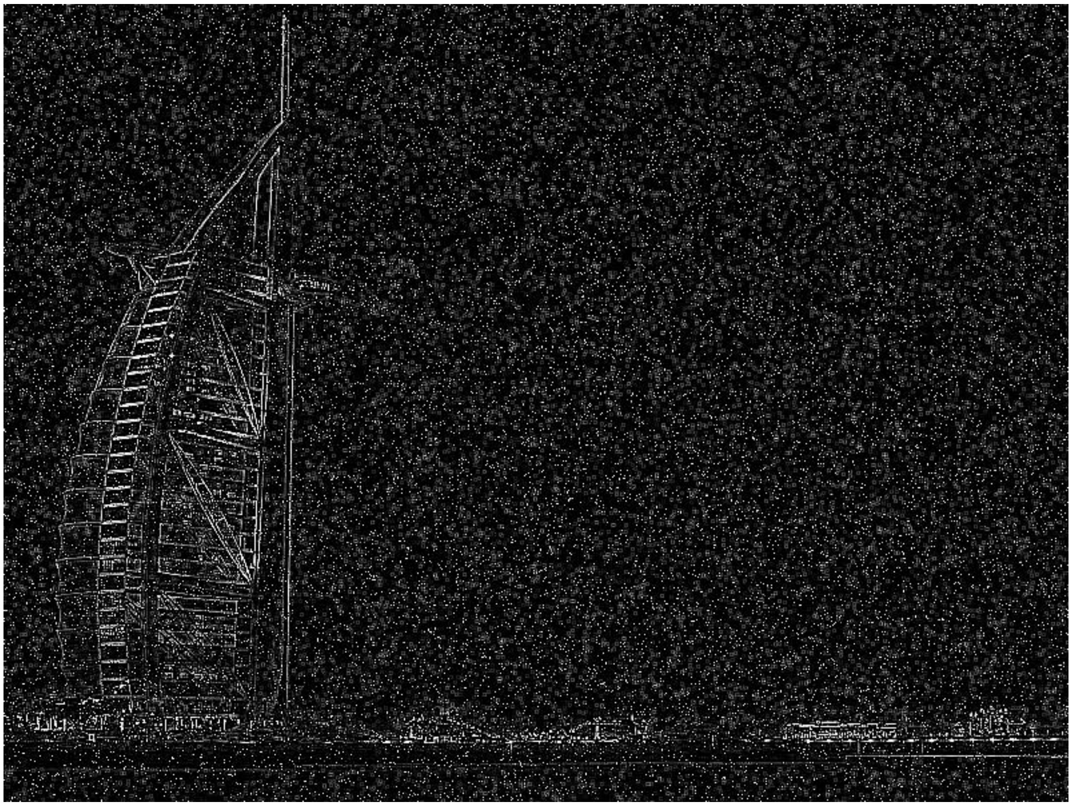

Figure 5: Edge Extraction Noisy Image Without Scaling

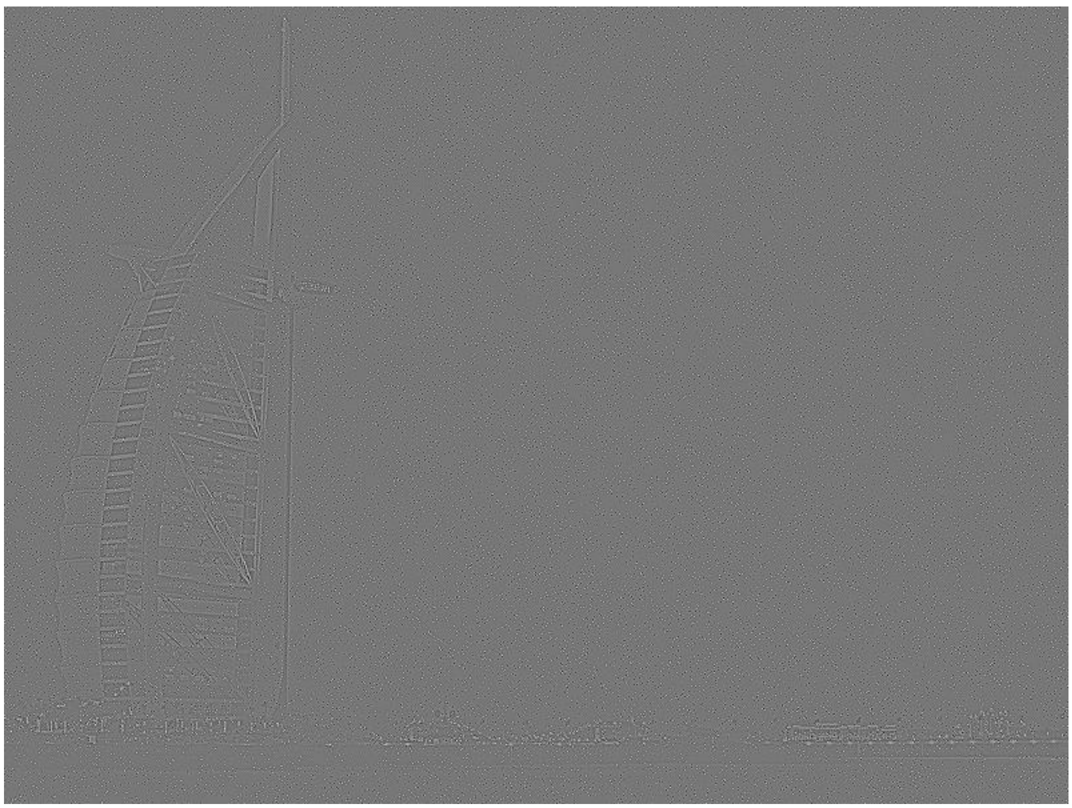

Figure 6: Edge Extraction Noisy Image With Scaling

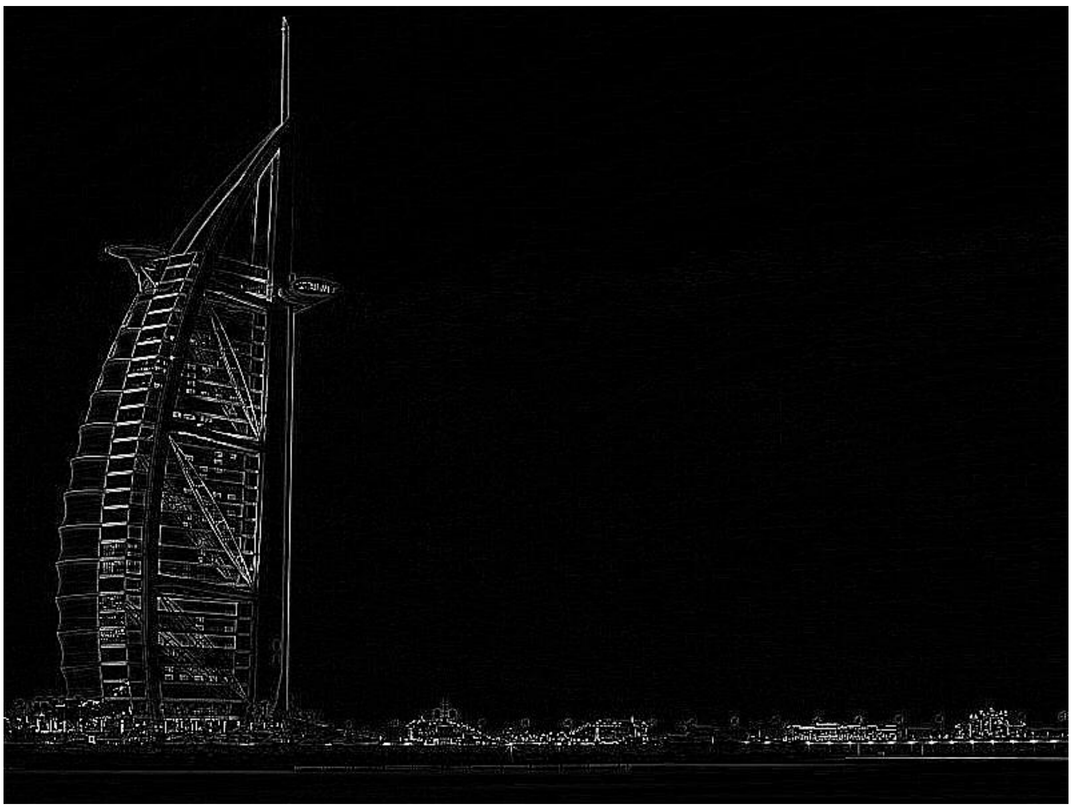

Figure 7: Edge Extraction No Noise Image Without Scaling

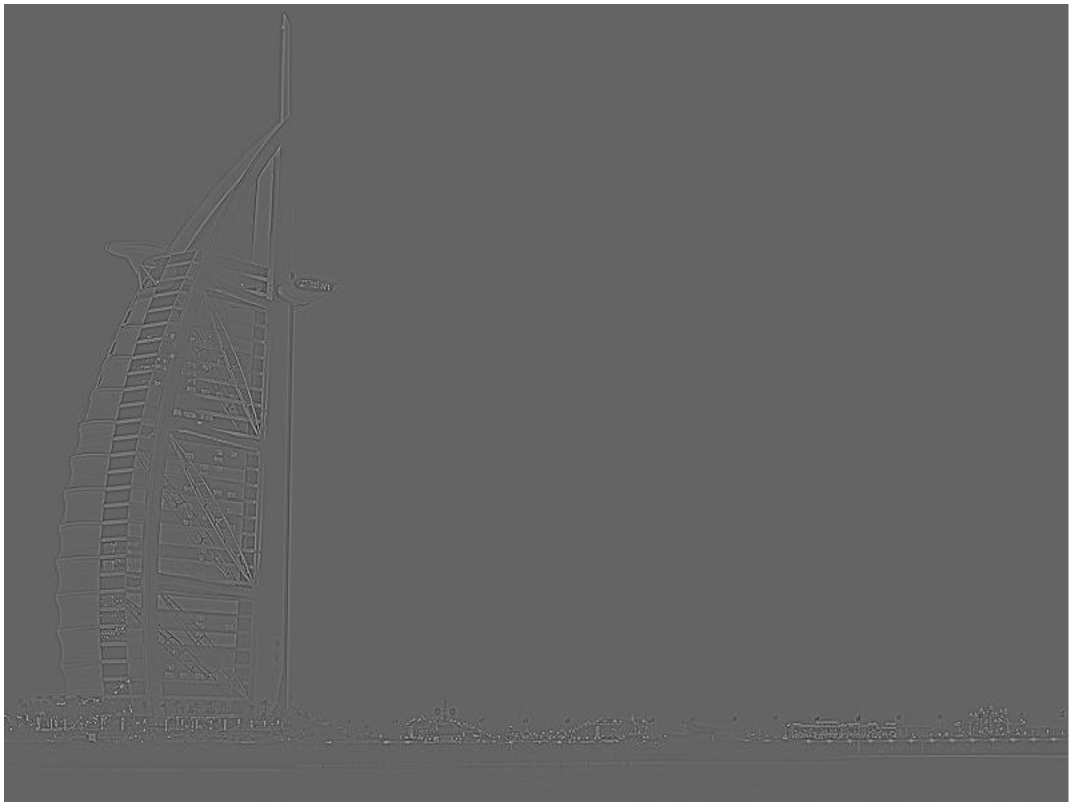

Figure 8: Edge Extraction No Noise Image With Scaling

4.4 MATLAB Source Code

Sharpen

```
function OutPutImage = Sharpen(Image, Mask, Add_X_Y)
%SHARPEN Summary of this function goes here
%   Detailed explanation goes here
    if (mod(size(Mask, 1), 2) == 0) || (mod(size(Mask, 2), 2) == 0)
        error('it is recommended to use a mask with odd width and height');
    end

    m = (size(Mask, 2)-1)/2;
    n = (size(Mask, 1)-1)/2;

    OutPutImageMat = zeros(size(Image, 1)-2*n, size(Image, 2)-2*m);
    for j=m+1:size(Image, 2)-m
        for i=n+1:size(Image, 1)-n
            Values_X_Weights = double(ReadImagePart(Image, i, j, n, m)).*Mask;
            if Add_X_Y
                OutPutImageMat(i-n, j-m) = Image(i,j) + fix(sum(Values_X_Weights(:)));
            else
                OutPutImageMat(i-n, j-m) = fix(sum(Values_X_Weights(:)));
            end
        end
    end

    DoubleImage = mat2gray(OutPutImageMat, [0 255]);
    OutPutImage = im2uint8(DoubleImage);
end
```

5 Problem 4

Select two image sharpening methods and apply them to different kind of images and discuss your findings.

5.1 Discussion and Objective

- Sharpen grayscale image using sharp mask
- Sharpen colored image using sharp mask
- Sharpen grayscale image using unsharp mask
- Sharpen colored image using unsharp mask

5.2 Filter and Parameter Used

High pass filter is used to sharpen the image with sharp and unsharp masks.

5.3 Results and Remarks

Figure 9: Sharpened Gray Image using Sharp Mask

Figure 10: Sharpened Colored Image using Sharp Mask

Figure 11: Sharpened Gray Image using Unsharp Mask

Figure 12: Sharpened Colored Image using Unsharp Mask

5.4 MATLAB Source Code

ApplyHighPassFilter

```
function OutPutImage = ApplyHighPassFilter(Image, Mask, Scale)
%APPLYFILTER Summary of this function goes here
%   Detailed explanation goes here
    if (mod(size(Mask, 1), 2) == 0) || (mod(size(Mask, 2), 2) == 0)
        error('it is recommended to use a mask with odd width and height');
    end

    m = (size(Mask, 2)-1)/2;
    n = (size(Mask, 1)-1)/2;

    OutPutImageMat = zeros(size(Mask, 1)-n, size(Mask, 2)-m);
    for j=m+1:size(Image, 2)-m
        for i=n+1:size(Image, 1)-n
            Values_X_Weights = double(ReadImagePart(Image, i, j, n, m)).*Mask;
            OutPutImageMat(i-n, j-m) = fix(sum(Values_X_Weights(:)));
        end
    end

    if Scale
        DoubleImage = mat2gray(OutPutImageMat, [min(OutPutImageMat(:)) max(OutPutImageMat(:))]);
    else
        DoubleImage = mat2gray(OutPutImageMat, [0 255]);
    end

    OutPutImage = im2uint8(DoubleImage);
end
```

6 Problem 5

Demonstrate that in using two separate filters in a cascading operation, a single filter can be derived with the same resulting output

6.1 Discussion and Objective

Sharpen an image using two masks then sharpen it again using a mask made using both of the original masks

6.2 Filter and Parameter Used

$$\text{Mask1} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\text{Mask2} = \begin{bmatrix} -1 & 0 & -1 \\ 0 & 5 & 0 \\ -1 & 0 & -1 \end{bmatrix};$$

$$\text{Mask3} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & -5 & -3 & -5 & 1 \\ 0 & -3 & 25 & -3 & 0 \\ 1 & -5 & -3 & -5 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

6.3 Results and Remarks

Figure 13: Applying Two Sharpen Masks in Sequence

Figure 14: Applying One Sharpen Mask

7 Main Source Code

MainSourceCodeFile

```
%
    ##########################################################################
%##                                      Students  names  :
                                     ##
%##                                      Omar  Rawashdeh
                                         ##
%##                                        Harshal  Raut
                                           ##
%##                                          Utsav  Shah
                                             ##
%
    ##########################################################################
```

```matlab
% ############################################################
%##                          Digital Image Processing HW2                          ##
%##                         GrayScale & Spatial Filtering                          ##
%##                              Matlab R2016a was used                            ##
%##                                  v9.0.0.341360                                  ##
% ############################################################
% ############################################################
%%
%Read/Load example images
Image1 = imread('Images/1.jpg');
Image2 = imread('Images/2.jpg');
% Image3 = imread('Images/3.jpg');
Image4 = imread('Images/4.jpg');
Image5 = imread('Images/5.jpg');
Image6 = imread('Images/6.jpg');
Image7 = imread('Images/7.jpg');
Image8 = imread('Images/8.jpg');
Image9 = imread('Images/9.jpg');
Image10 = imread('Images/10.jpg');
Image11 = imread('Images/11.jpg');
Image12 = imread('Images/12.jpg');

warning('off', 'MATLAB:MKDIR:DirectoryExists');
mkdir('results');
mkdir(fullfile(pwd,'results'), 'Point1');
mkdir(fullfile(pwd,'results'), 'Point2');
mkdir(fullfile(pwd,'results'), 'Point3');
mkdir(fullfile(pwd,'results'), 'Point4');
mkdir(fullfile(pwd,'results'), 'Point5');
warning('on', 'MATLAB:MKDIR:DirectoryExists');

%<<<<<<<<<<< Preparing a grayscale Image and a noisy image >>>>>>>>>>>%
%==================================================
%convert colored image to grayscale
%--------------------------------------------------
```

```
%parameters :
%1) the colored image that we want to convert to gray
GrayImage = Convert2Gray(Image6);
%===================================================
imwrite(GrayImage, fullfile(pwd,'results','GrayScale.jpg'), 'jpg');


%===================================================
%Add random noise to an image
%---------------------------------------------------
%parameters :
%1) the image that we want to add noise to
%2) amount/value/amplitude of noise
%3) percentage of noise, if 0 no noise will be added if 100 all pixel will
%have noise added to them
NoisyGrayImage = AddNoise(GrayImage, 150, 10);
%===================================================
imwrite(NoisyGrayImage, fullfile(pwd,'results','NoisyGrayImage.jpg'), 'jpg');
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>%
%%



%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<POINT 1>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>%
%%
%===================================================
%apply the gray level Log Transformation on a grayscale image
%formula is: new level = constant*log(1+old level)
%---------------------------------------------------
%parameters :
%1) the Gray scale image that we want to modify
GrayTransformedImage = GrayLevelTransform_Log(GrayImage);
%===================================================
imwrite(GrayTransformedImage, fullfile(pwd,'results','Point1','GrayLevelTransformedImage.jpg'), 'jpg');
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>%
%%
```

```
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<POINT 2>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>%
%%
%==========================================================
%construct a mask then blur the Noisy image using a low pass filter (all weights are positive)
%----------------------------------------------------------
MaskBlurring = [1 2 1;
                2 3 2;
                1 2 1];
%parameters :
%1) the noisy image that we want to blur
%2) Blurring mask, all weights must be positive
BlurredImage = SpatialFilter2_BlurImage(NoisyGrayImage, MaskBlurring);
%==========================================================
imwrite(BlurredImage, fullfile(pwd,'results','Point2','BlurredImage.jpg'), 'jpg');

%==========================================================
%Clear the noise from an image using statistical mask(median).
%this function gives a better performance in removing the random noise
%compared to just blurring or taking average
%----------------------------------------------------------
%parameters :
%1) the noisy image that we want to remove the noise from
%2) n value =>> if you want the mask size to be 3x3 then enter n as 1
ClearedImage = SpatialFilter1_RemoveNoise_Statistical(NoisyGrayImage, 1);
%==========================================================
imwrite(ClearedImage, fullfile(pwd,'results','Point2','ClearedImageUsingMedianFilter.jpg'), 'jpg');
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>%

%%
```

```
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<POINT 3>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>%
%%
%==================================================================
%Edge Extraction using a Mask of values 0 -1 0; -1 4 -1; 0 -1 0
%(highpass filter) with scaling
%------------------------------------------------------------------
%
MaskEdgeExtraction1 = [0 -1 0;
                        -1 4 -1;
                        0 -1 0];
%Parameters :
%1) the GrayScale Image that we want to extract edges from using HighPass
%filter
%2) the Mask
%3) Boolean Value indicating whether we want to scale values to 0 255 after
%applying the filter
EdgeExtractionNoNoiseImage1_WithScaling = ApplyHighPassFilter(GrayImage, MaskEdgeExtraction1, true);
EdgeExtractionNoisyImage1_WithScaling = ApplyHighPassFilter(NoisyGrayImage, MaskEdgeExtraction1, true);
%==================================================================
imwrite(EdgeExtractionNoNoiseImage1_WithScaling, fullfile(pwd,'results','Point3','EdgeExtractionNoNoiseWithScaling.jpg'), 'jpg');
imwrite(EdgeExtractionNoisyImage1_WithScaling, fullfile(pwd,'results','Point3','EdgeExtractionNoisyWithScaling.jpg'), 'jpg');

%==================================================================
%Edge Extraction using a Mask of values -1 -1 -1; -1 8 -1; -1 -1 -1
%(highpass filter) without scaling
%------------------------------------------------------------------
MaskEdgeExtraction2 = [-1 -1 -1;
                        -1 8 -1;
                        -1 -1 -1];
%Parameters :
%1) the GrayScale Image that we want to extract edges from using a HighPass
%filter
```

```
%2) the Mask
%3) Boolean Value indicating whether we want to scale values to 0 255 after
%applying the filter
EdgeExtractionNoNoiseImage2_WithoutScaling = ApplyHighPassFilter(GrayImage, MaskEdgeExtraction2, false);
EdgeExtractionNoisyImage2_WithoutScaling = ApplyHighPassFilter(NoisyGrayImage, MaskEdgeExtraction2, false);
%==========================================
imwrite(EdgeExtractionNoNoiseImage2_WithoutScaling, fullfile(pwd,'results','Point3','EdgeExtractionNoNoiseWithoutScaling.jpg'), 'jpg');
imwrite(EdgeExtractionNoisyImage2_WithoutScaling, fullfile(pwd,'results','Point3','EdgeExtractionNoisyWithoutScaling.jpg'), 'jpg');
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>%

%%


%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<POINT 4>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>%
%%
%==========================================
%Sharpen GrayScale Image with a mask [ -1 -1 -1; -1 9 -1; -1 -1 -1]
%Note: we use the same mask that we used for edge extraction [ -1 -1 -1; -1 8 -1; -1 -1 -1]
%we just tell the sharping function to add the value of f(x,y) to the new image,
%thus 8 will become 9
%------------------------------------------
%parameters :
%1) GrayScale Image that we want to sharpen
%2) Mask
%3) boolean to tell the function to add the f(x,y) (value of original
%image) or not, if it is false, mask must have weights that sum up to a number
%other than zero (1 if the middle is + and -1 if the middle is -)
SharpenedGrayImage1 = Sharpen(GrayImage, MaskEdgeExtraction2, true);
```

```
% S1 = size(Image6, 1)-2;

ColoredImageMatrix = zeros(size(Image6, 1)-2, size(Image6, 2)-2, size(Image6, 3), 'uint8');
%for the colored image sharpen each channel seperately
ColoredImageMatrix(:,:,1) = Sharpen(Image6(:,:,1), MaskEdgeExtraction2, true);
ColoredImageMatrix(:,:,2) = Sharpen(Image6(:,:,2), MaskEdgeExtraction2, true);
ColoredImageMatrix(:,:,3) = Sharpen(Image6(:,:,3), MaskEdgeExtraction2, true);
SharpenedColorImage1 = im2uint8(ColoredImageMatrix);
%==================================================
imwrite(SharpenedGrayImage1, fullfile(pwd,'results','Point4','SharpenedGrayImageConvolution.jpg'), 'jpg');
imwrite(SharpenedColorImage1, fullfile(pwd,'results','Point4','SharpenedColoredImageConvolution.jpg'), 'jpg');

%==================================================
%Sharpen GrayScale Image using unsharp masking
%--------------------------------------------------
Blurring_Mask = [1 1 1;
                 1 1 1;
                 1 1 1];

BlurredImageToProduceUnSharpMask = SpatialFilter2_BlurImage(GrayImage, Blurring_Mask);
Unsharp_Mask = GrayImage - BlurredImageToProduceUnSharpMask;
SharpenedGrayImage2 = GrayImage + Unsharp_Mask;
%now do it for colored image
ColoredImageMatrix2 = Image6;
BlurredColoredChannels = Image6;
Unsharp_Mask_Colored = Image6;
BlurredColoredChannels(:,:,1) = SpatialFilter2_BlurImage(Image6(:,:,1), Blurring_Mask);
BlurredColoredChannels(:,:,2) = SpatialFilter2_BlurImage(Image6(:,:,2), Blurring_Mask);
BlurredColoredChannels(:,:,3) = SpatialFilter2_BlurImage(Image6(:,:,3), Blurring_Mask);
Unsharp_Mask_Colored(:,:,1) = Image6(:,:,1) - BlurredColoredChannels(:,:,1);
Unsharp_Mask_Colored(:,:,2) = Image6(:,:,2) - BlurredColoredChannels(:,:,2);
```

```
Unsharp_Mask_Colored(:,:,3) = Image6(:,:,3) - BlurredColoredChannels(:,:,3);
SharpenedColorImage2 = Image6 + Unsharp_Mask_Colored;
%==========================================
imwrite(SharpenedGrayImage2, fullfile(pwd,'results','Point4','SharpenedGrayImageUnsharpMasking.jpg'), 'jpg');
imwrite(SharpenedColorImage2, fullfile(pwd,'results','Point4','SharpenedColoredImageUnsharpMasking.jpg'), 'jpg');

%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<%
%%



%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<POINT 5>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>%
%%
%==========================================
%sharpen an image using two masks then sharpen it again using a mask made
%using both of the original masks
%------------------------------------------
Mask1 = [0 -1 0;
         -1 5 -1;
         0 -1 0];
Mask2 = [-1 0 -1;
          0 5 0;
         -1 0 -1];

Mask3 = [0  1  0  1 0;
         1 -5 -3 -5 1;
         0 -3 25 -3 0;
         1 -5 -3 -5 1;
         0  1  0  1 0];

SequenceImage_Step1 = Sharpen(GrayImage, Mask1, false);
SequenceImage_Step2 = Sharpen(SequenceImage_Step1, Mask2, false);
SequnceImage = SequenceImage_Step2;
OneMaskImage = Sharpen(GrayImage, Mask3, false);
%==========================================
```

```
imwrite(SequnceImage, fullfile(pwd,'results','Point5','
    Applying2SharpenMasksInSequence.jpg'), 'jpg');
imwrite(OneMaskImage, fullfile(pwd,'results','Point5','
    Applying2SharpenMasksUsing1Mask.jpg'), 'jpg');
%
    <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>%

%%
```

References

[1] Ehsan Khoramshahi. *Image Enhancement in Spatial Domain.*

[2] P. K. Sinha. *Gray-Level Transformation.* Tech. rep. Image Acquisition and Pre-processing for Machine Vision Systems, 2012.

[3] *Concept of Blurring.* Tech. rep. 2016. URL: `https://www.tutorialspoint.com/dip/concept_of_blurring.htm`.

[4] Cyanogen Imaging^{TM} MaxIm DL. *High-Pass Filtering.* Diffraction Limited.