OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG EIT FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIONSTECHNIK

Digital Information Processing

Implementation of High-pass and Low-pass filter for the reduction of noise of an image with Matlab

Abu Sayed
Matrikel Nr.: 224013

1. Contents of the Experiment

The goal of this lab is to get familiar with the bases of digital image processing. You will learn how images are similar to other signals and how can they be stored and processed using the same techniques as for other signals.

2. Preparatory Exercises – To Be Done Before the Lab

Exercise 2.1: Image as a Signal

In what way can images be seen as signals?

Answer:
An image is a signal and it is defined as a two-dimensional function, F(x,y), where x and y are spatial coordinates and the amplitude of F at any pair of coordinates (x,y) is called the intensity of that image at that point [1]. When x,y, and amplitude values of F are finite, we call it a digital image. In other words, an image can be defined by a two-dimensional array specifically arranged in rows and columns.

Figure 1: Image with two dimensional arrays of numbers ranging between 0 and 255.

Digital Image is composed of a finite number of elements, each of which elements have a particular value at a particular location. These elements are referred to as picture elements, image elements, and pixels. A Pixel is most widely used to denote the elements of a Digital Image.

In order to create a digital image, we have to convert signals into a digital form. This involves sampling and quantization. The result of sampling and quantization results in a two-dimensional array or matrix of numbers which are nothing but a digital image [1].

Digital Image representation in MATLAB:

$$f = \begin{bmatrix} f(1,1) & f(2,1) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & \cdots & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{bmatrix}$$

In MATLAB the start index is from 1 instead of 0 [2]. Therefore, f(1,1) = f(0,0).
In MATLAB, matrices are stored in a variable i.e X,x,input_image , and so on.

Exercise 2.2: Image Representation

Find out how images can be stored digitally, especially with respect to color channels. What Possibilities are there?

Answer:
Monitor display consists of a numerous number of triplets, for instance, red, green, and blue dots. Based on the manufacturers the shape of these dots may vary, like: square. The triplets of red, green, and blue are called pixels.

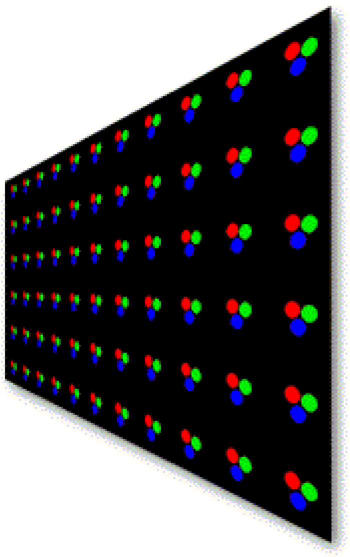


Figure 2: Image and Channels [3]

Every pixel must have a specific number to tell the computer about the color combination. Like: for a triplet as 255, 0, 0 means to run the R element to 255 brightness, the G element to 0 brightness, and also the B element to 0 brightness. This output will be seen by humans as a shade of red color. Digital image processing uses digitization to turn an image into a set of pixels. For displaying an image, the computer gives information to monitor to represent a fixed color for individual pixels. Images are represented as a

Height, Width, and Channels. For RGB images channels will be 3 and for greyscale images, channels will be 1. Furthermore, it is possible to increase the number of channels. Like: for a graphics editing software using four channels: 3 for RGB and alpha or ‘a’ channel. This ‘a’ channel is used to make the image more transparent. Another use of increasing the number of channels in satellite sensors, channels can add to highlight the height of pixels [3].

Exercise 2.3: Color and Greyscale

What possibilities are there to convert images to greyscale? Explain the algorithms. What are the differences between them? Effect according to your own taste.

Answer:
The grey-scale image uses only one channel that represents only gray. The main purpose of this image is to provide less information as it requires only one intensity value for each pixel [5]. Since much of today’s hardware can support 8-bit images, hence Greyscale image has become very common [5]. Furthermore, Greyscale is sufficient for different tasks, hence complicated the process can be avoided [5].

There are three algorithms for changing color images to Greyscale and the explanation and differences between them are given below,

Lightness method [4]: This technique takes the average of the most prominent and least prominent colors, for example, red, blue, and green.

Lightness value: $(\max(RED, GREEN, BLUE) + \min(RED, GREEN, BLUE))/2$

Drawback: It tends to reduce the contrast of original image.

Average method [4]: It is a simple one among others. It takes the average value of the three colors value.

Average value= $(RED + GREEN + BLUE)/3$

The main benefit of the average method, it is easy to implement and optimize. But the biggest drawback picture becomes is the converted darker. Because the average method assumed that each color has the same contribution in the image that is not really in reality.

Weighted/Luminosity method [4]: The luminosity method considered as a more advanced version of the average method and this method takes into consideration the wavelength of the colors. Like green is the most sensitive to the eye but it has less wavelength than red color. So, it’s better to increase the contribution of green color and decrease the red color, and putting the blue color wavelength in between.

Weighted value= $(0.21RED + 0.72GREEN + 0.07BLUE)$

According to this equation, Red has 21% contribution, Green has 71% contribution, and Blue is 7% contribution.

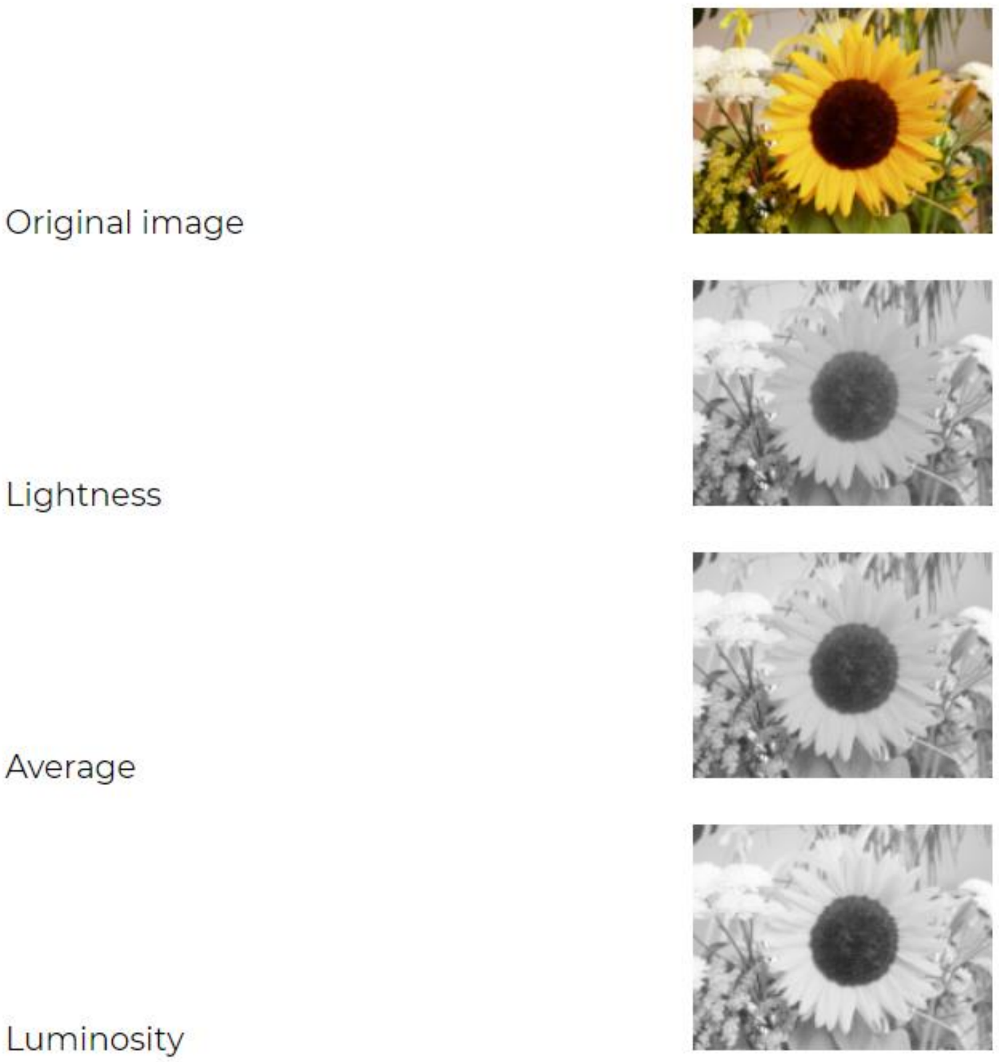


Figure 3: Greyscale view of a Sunflower using different method [5].

If we take a look on the figure 4, we can observe that the lightness method have a tendency to decline the contrast of the image. On the other hand, the luminosity method is looking best compare to other method output.

3. Experiments – To Be Done During the Lab

The MATLAB file image_filering.m applies a low-pass filter and a high-pass filter on a selected image (the variable images contains the filenames, while image_index selects the image at the corresponding index).

Exercise 3.1: Image representation in Matlab

Find out how the image is represented in the variable *I*, especially with respect to images in greyscale and color. Which color space is used?

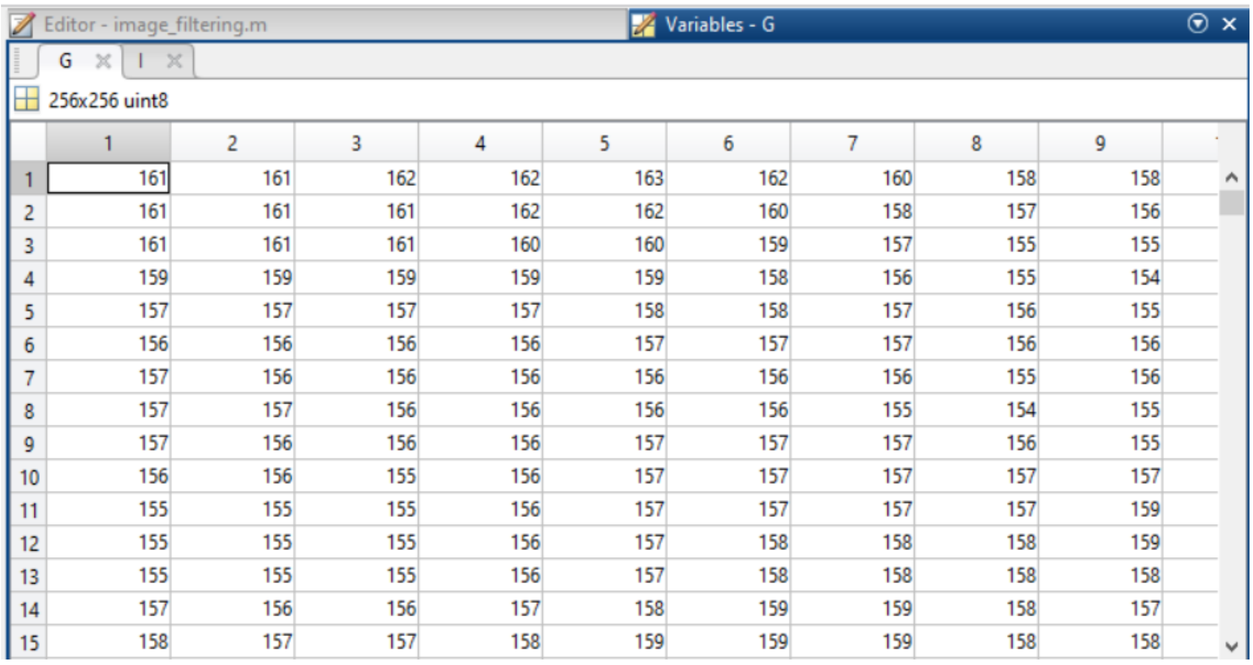
Answer:
The image is represented in the variable "I" using the "imread()" command in MATLAB. This "imread()" is basically a built-in function in MATLAB. It reads the image and returns the value as an array, for instance, array A. When we have an image on the greyscale format then imread() function will return an array A consist of two-dimensional data, such as M*N. Suppose we have a color image, at that time it will return a three-dimensional array A, for example, M*N*3.

Though we have various color spaces like CIE, RGB, CMYK, YUV, and so on. In our exercise, we are going to use RGB spacing.

MATLAB code:

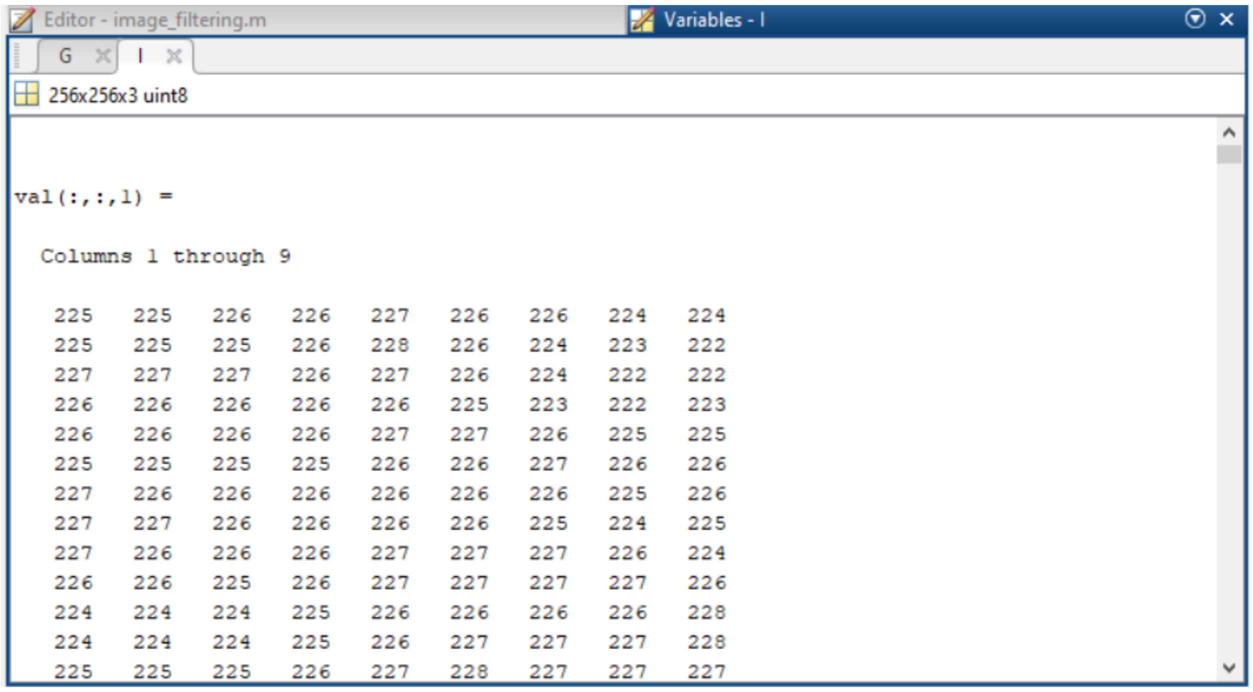
```
images=[{'cameraman.jpg'}, {'len_std.jpg'},{'ovgu.jpg'}];

image_index=2;
greyscale=1;

%% Image Filtering
% First we should read image
I=imread(images{image_index});

%%black white conversation
if (greyscale==1)
    %implement formula here
     G = rgb2gray(I)
end
```

At the beginning of the code we declare a list of images, then we read the image using imread() function and store the output array on variable I. After that, we converted the image from RGB to greyscale using rgb2gray() function.



	1	2	3	4	5	6	7	8	9
1	161	161	162	162	163	162	160	158	158
2	161	161	161	162	162	160	158	157	156
3	161	161	161	160	160	159	157	155	155
4	159	159	159	159	159	158	156	155	154
5	157	157	157	157	158	158	157	156	155
6	156	156	156	156	157	157	157	156	156
7	157	156	156	156	156	156	156	155	156
8	157	157	156	156	156	156	155	154	155
9	157	156	156	156	157	157	157	156	155
10	156	156	155	156	157	157	157	157	157
11	155	155	155	156	157	157	157	157	159
12	155	155	155	156	157	158	158	158	159
13	155	155	155	156	157	158	158	158	158
14	157	156	156	157	158	159	159	158	157
15	158	157	157	158	159	159	159	158	158

Figure 4: Array representation of Greyscale image using rgb2gray() function.



```
val(:,:,1) =

  Columns 1 through 9

   225   225   226   226   227   226   226   224   224
   225   225   225   226   228   226   224   223   222
   227   227   227   226   227   226   224   222   222
   226   226   226   226   226   225   223   222   223
   226   226   226   226   227   227   226   225   225
   225   225   225   225   226   226   227   226   226
   227   226   226   226   226   226   226   225   226
   227   227   226   226   226   226   225   224   225
   227   226   226   226   227   227   227   226   224
   226   226   225   226   227   227   227   227   226
   224   224   224   225   226   226   226   226   228
   224   224   224   225   226   227   227   227   228
   225   225   225   226   227   228   227   227   227
```

Figure 5: Array representation of color image using imread() function.

Interpretation:
From figure 4, we can see that the greyscale image is stored in a two-dimensional array such as 256 x 256 uint8 while from the figure 5, we can observe that the color image is stored in a three dimensional array for example 256 x 256 x 3 uint8. Here uint8 means data type is unsigned integer 8 bits.

Exercise 3.2: Low-pass filter and high-pass filter

Describe qualitatively the effect of lowpass/highpass filtering of the given images. Therefore, also apply different filter coefficients. Add further images (in *jpg* format).

Answer:
In the field of image processing, filtering is an important technique used to manipulate the image. An image can be filtered in many aspects like sharpening, smoothing, noise reduction and edge enhancements. A filter can be described by kernel which consists of a small array takes account to each pixel and its neighboring pixels. Therefore, we can say that different type of the algorithm generates different types of kernels and it filtered the image in various types. For instance, to blur the image, to change the brightness, etc.

Low-pass Filter (LPF): A lowpass filter is considered essential for most of the smoothing system. An image can be smoothed by declining the disparity between pixel values by averaging neighboring pixels. A LPF tends to hold the low frequency information of an image and at the same time it will reduce the high frequency components of an image [6]. In this exercise, Gaussian filter is used as a LPF.

Gaussian Filter[7]: It is defined by a weighted average of each pixel's neighboring pixel, with the average weighted more towards the value of the central pixel [7], This algorithm is useful for blur image and noise removing.

High-pass Filter (HPF): A HPF retain high frequency components and reduce low frequency components within an image [6], it is used for sharpening control of an image. In this exercise, obel filter is used as a HPF.

Sobel filter: This filter is used for horizontal and vertical edge emphasis. A Sobel filter convolves the image with a small, separable, integer valued filter that operates in horizontal or vertical direction. Hence, a Sobel filter is inexpensive to compute and implement. A 3x3 kernel is used to convolve with the image to calculate approximations (horizontal and vertical) for derivatives. A typical kernel for horizontal emphasis looks like,

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Since the middle row is zero, during convolution, rather than taking the current pixels, it takes the top and bottom pixels and calculates horizontal values. This way, more importance (weight) is given to the edge between the pixels and thus the edge is enhanced. The transpose of this matrix leads to vertical edge enhancement. Changing the values of middle row/column elements i.e 2, -2 to higher values leads to higher enhancement of images.

MATLAB code:

```
close all
clear all
clc
images=[{'cameraman.jpg'}, {'len_std.jpg'},{'ovgu.jpg'},{'tony_cross.jpg'},
{'Bridge.jpg'}];

image_index=2;
greyscale=0;

%% Image Filtering
% First we should read image
I=imread(images{image_index});
```

```
%%black white conversation
if (greyscale==1)
    %implement formula here
     G = rgb2gray(I);

end

I_noise= imnoise(I,'salt & pepper',0.02);
figure
subplot(1,2,1);imshow(I);title('Original Image')
subplot(1,2,2);imshow(I_noise);title('Original image with Salt&Pepper noise')
% act as low-pass filter on image
H = fspecial('gaussian',[3 3],2);
I_filt= imfilter(I,H);
I_filt_nis=imfilter(I_noise,H);
figure
subplot(1,2,1)
imshow(I_filt);title('Gaussian Filter on Original image')
subplot(1,2,2);imshow(I_filt_nis);title('Gaussian Filter on noisy original image')

% high-pass filter
H=fspecial('sobel');
H_1 = transpose(H);

I_filt= imfilter(I,H);
I_filt_nis= imfilter(I_noise,H);

I_filt_1= imfilter(I,H_1);
I_filt_nis_1= imfilter(I_noise,H_1);

figure
subplot(1,2,1)
imshow(I_filt);title('Horizontal Sobel Filter on Original image')
subplot(1,2,2);imshow(I_filt_nis);title('Horizontal Sobel Filter on noisy original
image')

figure
subplot(1,2,1)
imshow(I_filt_1);title('Vertical Sobel Filter on Original image')
subplot(1,2,2);imshow(I_filt_nis_1);title('Vertical Sobel Filter on noisy original
image')
```

At first we created a list of all images to which we will apply the filtering. In this exercise we add two extra images for example 'tony_cross.jpg' and 'Bridge.jpg'. From the matlab code, we can see that salt and pepper noise is added by using "imnoise " function. Then we apply Gaussian filter on original image and noisy original image. In addition, the "imfilter" function filters the multidimensional array with the multidimensional filter. Lastly we used sobel filter on original and noisy image.

As we know, smoothing and blurring image can be made by using low-pass filter and here we are using Gaussian filter as low-pass filter. In the output figure we will observe that, by changing the value of standard deviation and kernel size of the filter we can change the blur and smoothness percentage. On the other hand, Gaussian filter distributes the salt and pepper noise over wide region and the noise will be slightly visible in the output image of this filter. But the main advantage of this filter is that, the image boundaries will retain clearly.

Sobel filter is used for edge enhancement. By default a sobel filter will emphasize horizontal edge but doing transpose of the sobel filter we can observe vertical edge emphasize. Therefore, we can easily observe horizontal and vertical edges using sobel filter.

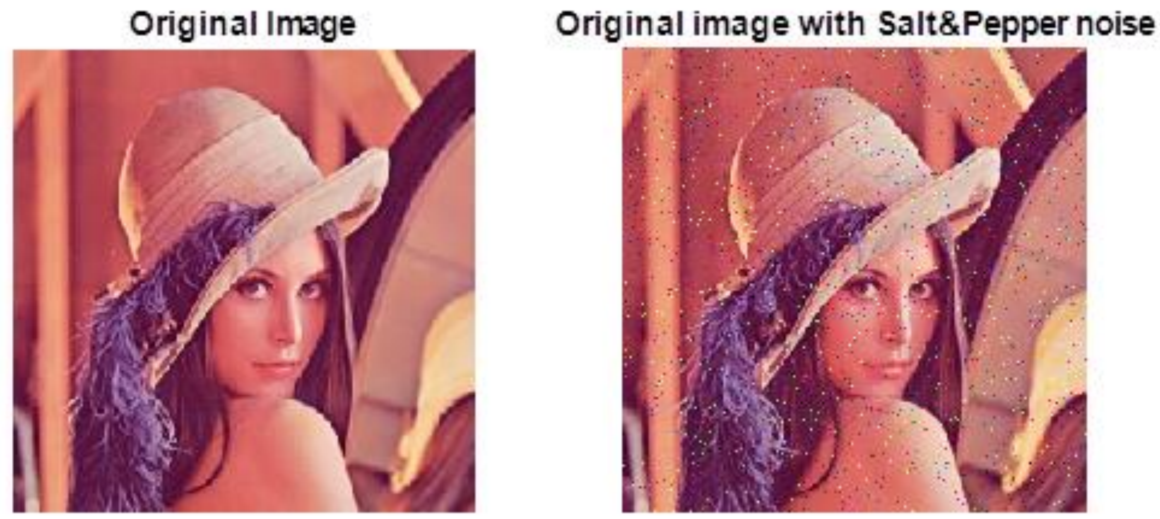


Figure 6: original len_std image and image with salt and pepper noise.

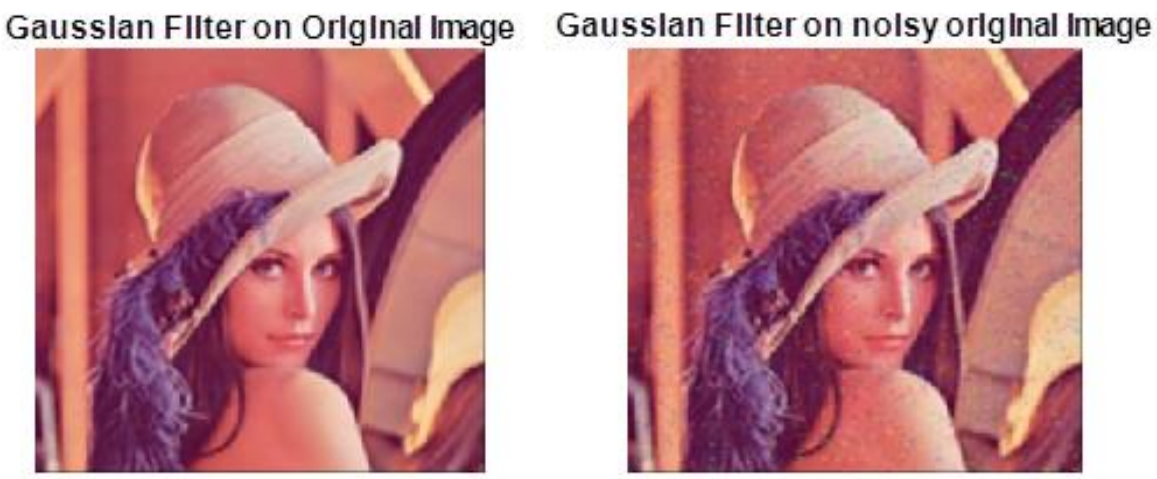

Gaussian Filter on Original Image
Gaussian Filter on noisy original image

Figure 7: Gaussian filter on original and noisy image with parameters h=3, σ =2.

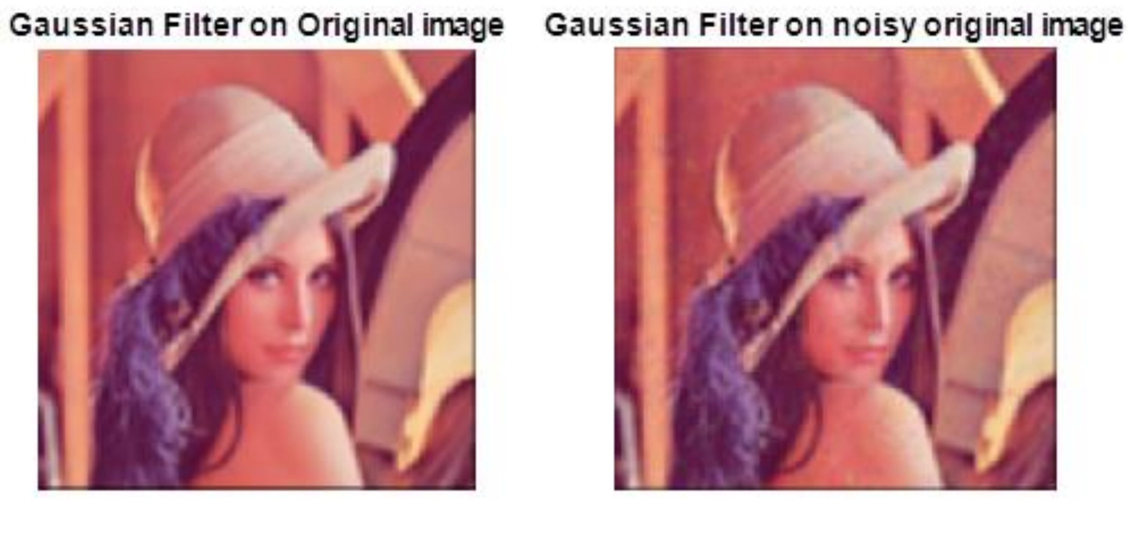

Gaussian Filter on Original image
Gaussian Filter on noisy original image

Figure 8: Gaussian filter on original and noisy image with parameters h=5, σ =20.

From the figure 7 we can see that after applying Gaussian filter to the original image looks like a smoothed and blurred. In the noisy image ideally filter should reduce the noise but Gaussian filter spreads the salt and pepper noise and we can see that noise is slightly noticeable. In the figure 8, we can observe that images looks like more blurred and noise is also less than figure 7, it's achieved by changing the Gaussian filter parameter.

The same procedures are going to perform for other images

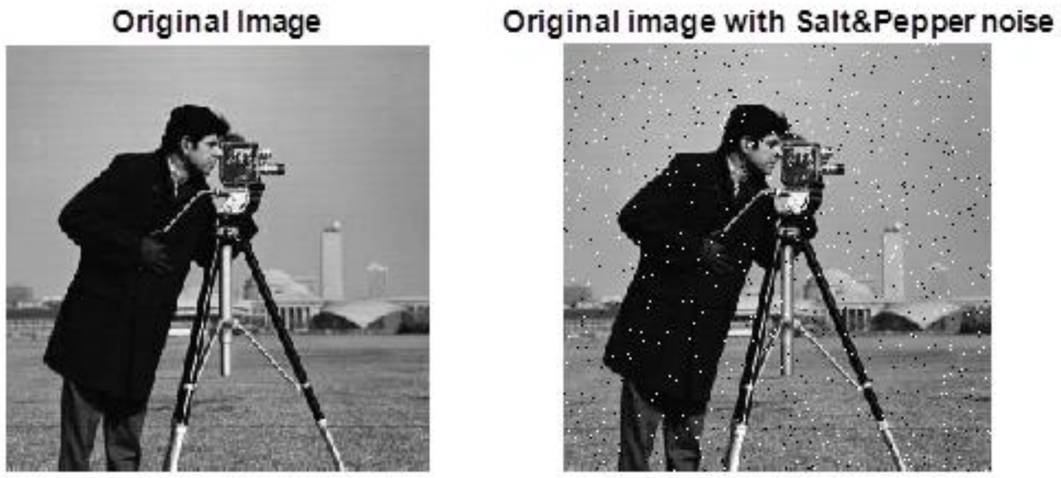


Figure 9: Original "cameraman.jpg" image and image with salt and pepper noise.

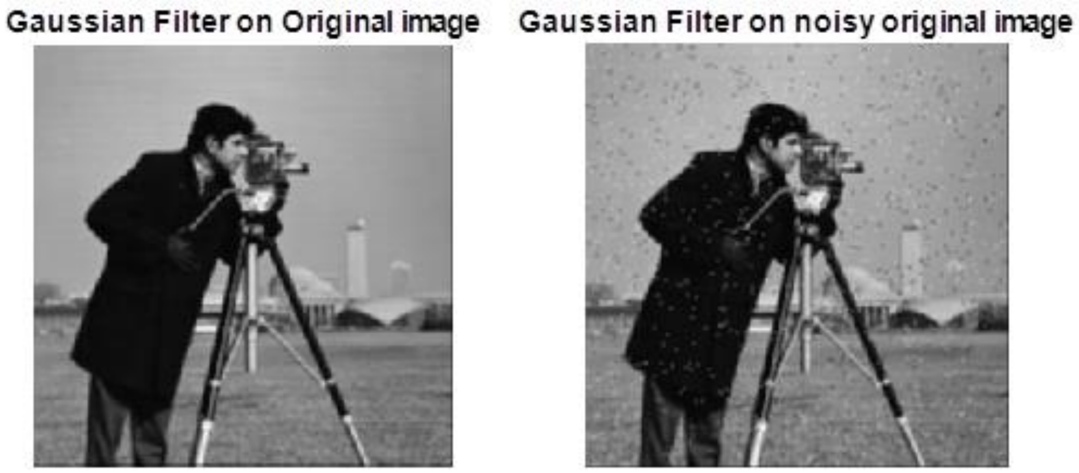

Gaussian Filter on Original image
Gaussian Filter on noisy original image

Figure 10: Gaussian filter on original and noisy image with parameters h=3, σ =2.

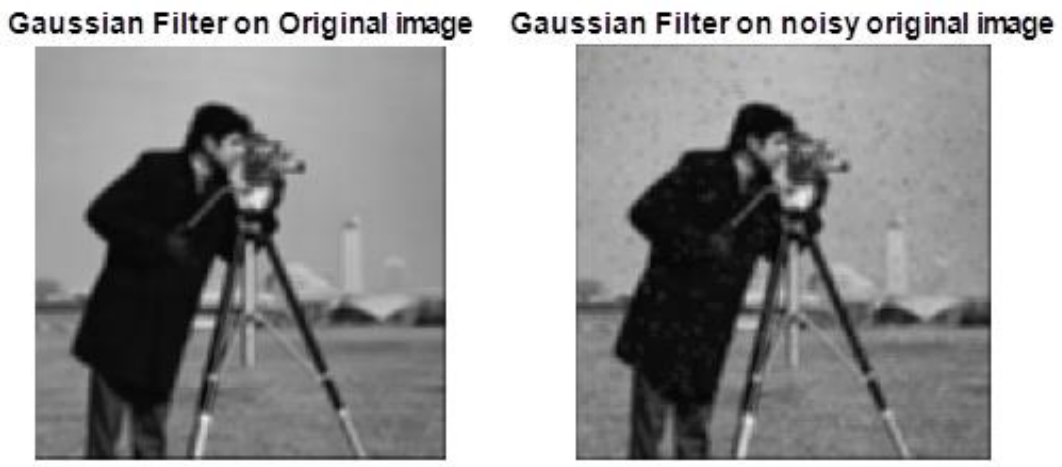

Gaussian Filter on Original image
Gaussian Filter on noisy original image

Figure 11: Gaussian filter on original and noisy image with parameters h=5, σ =20

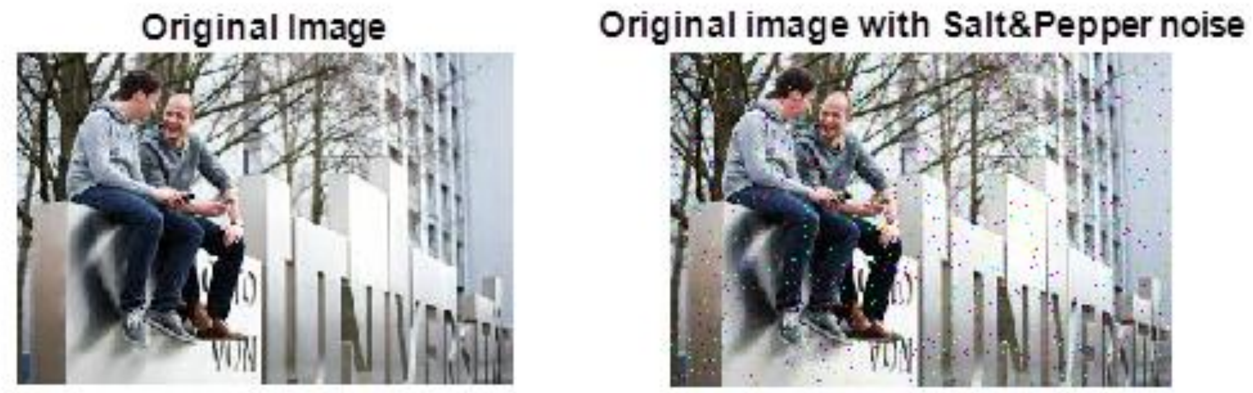


Figure 12: Original “ovgu.jpg” image and image with salt and pepper noise.

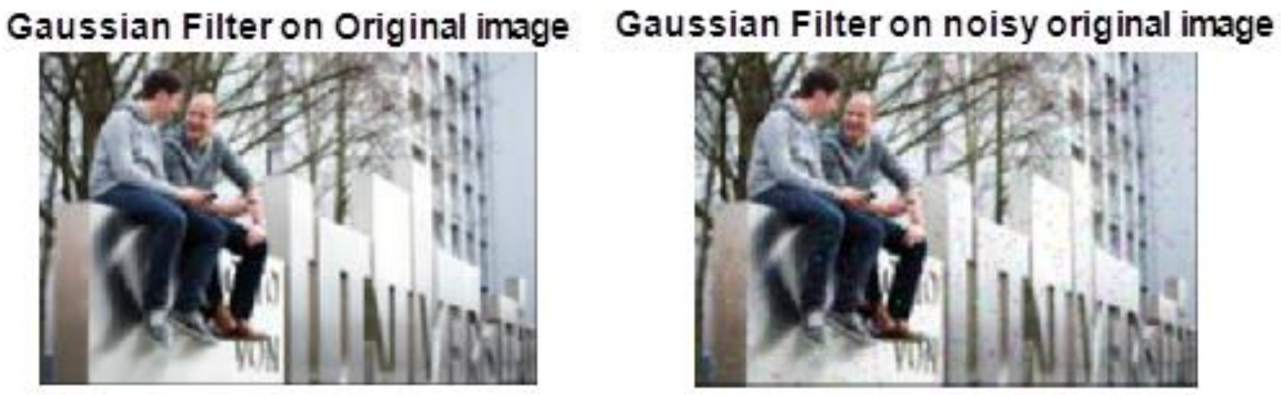

Gaussian Filter on Original image
Gaussian Filter on noisy original image

Figure 13: Gaussian filter on original and noisy image with parameters h=3, σ =2.

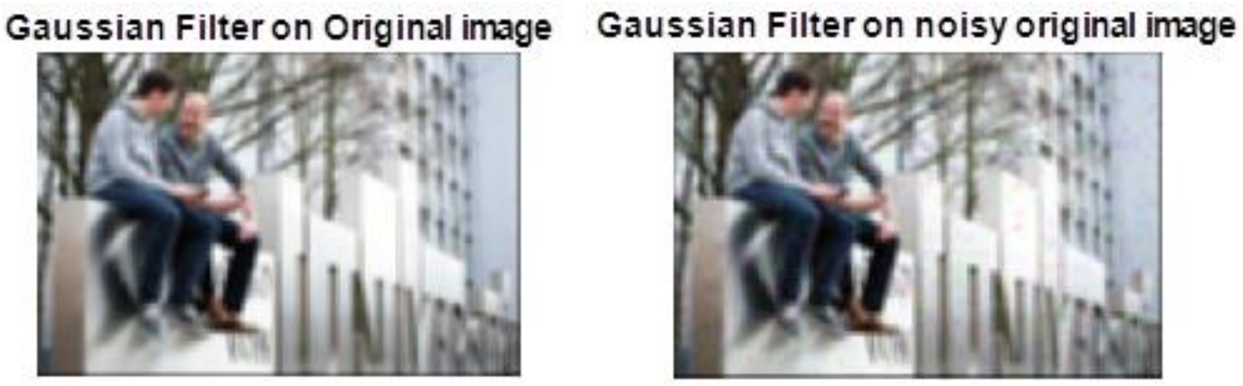

Gaussian Filter on Original image
Gaussian Filter on noisy original image

Figure 14: Gaussian filter on original and noisy image with parameters h=5, σ =20.

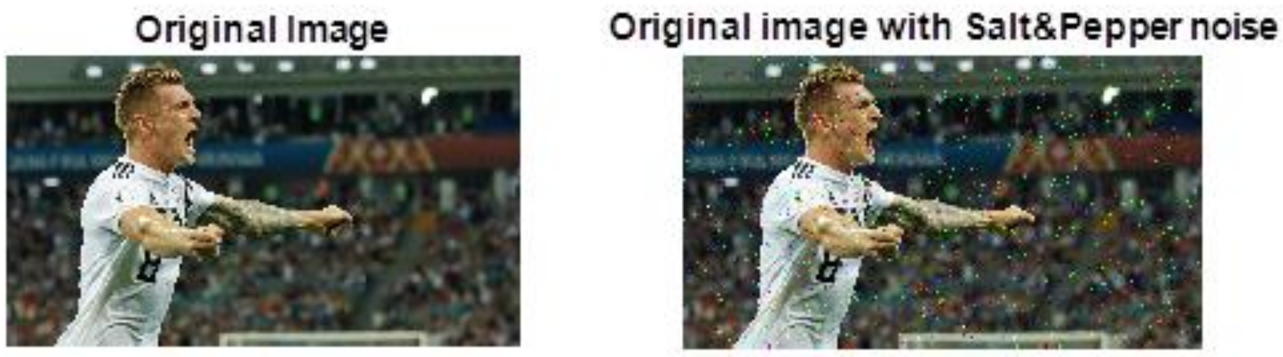


Figure 15: Original “tony_cross.jpg” image and image with salt and pepper noise

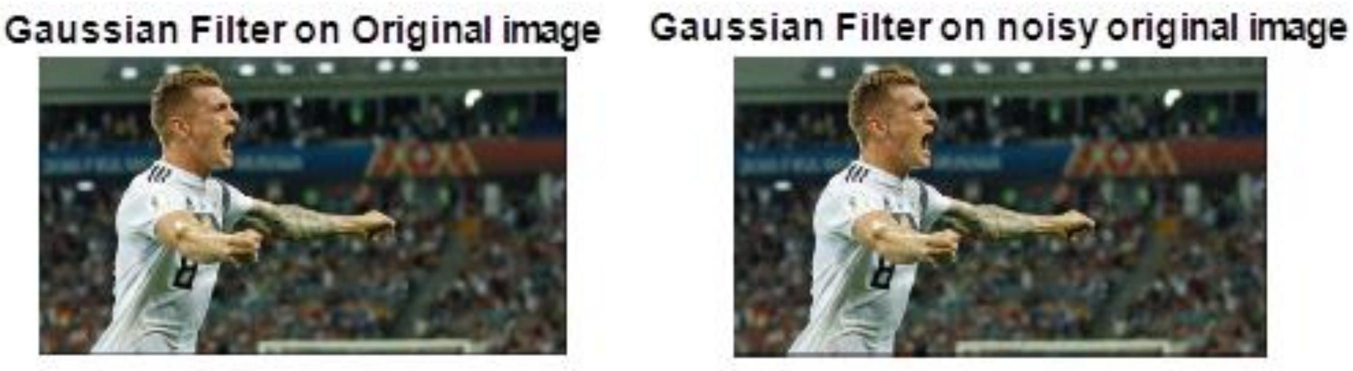

Gaussian Filter on Original image
Gaussian Filter on noisy original image

Figure 16: Gaussian filter on original and noisy image with parameters h=5, σ =20

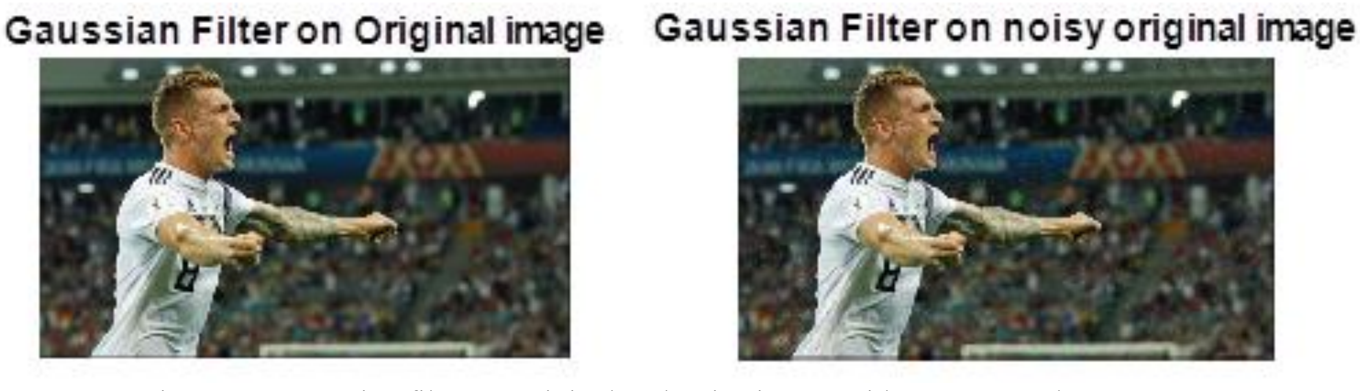

Gaussian Filter on Original image
Gaussian Filter on noisy original image

Figure 17: Gaussian filter on original and noisy image with parameters h=3, σ =2.

In the figure 16 and figure 17, we observe that after applying Gaussian filter there is no noise visible. The main reason is this image focused on "Tony Kross" body and the background is blurred as a result noisy part on image background easily erased and the main image object "Tony Kross" body looks like smooth and background going to be more blurred.

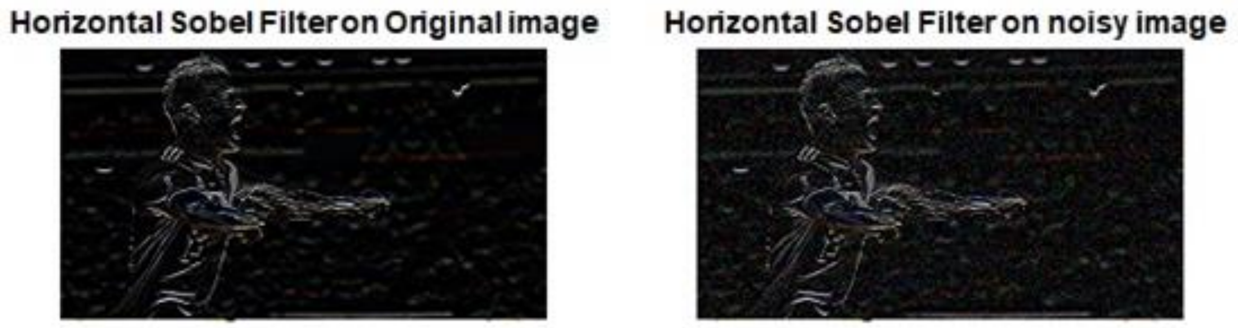

Horizontal Sobel Filter on Original image
Horizontal Sobel Filter on noisy image

Figure 18: Horizontal Sobel filter on original and noisy image

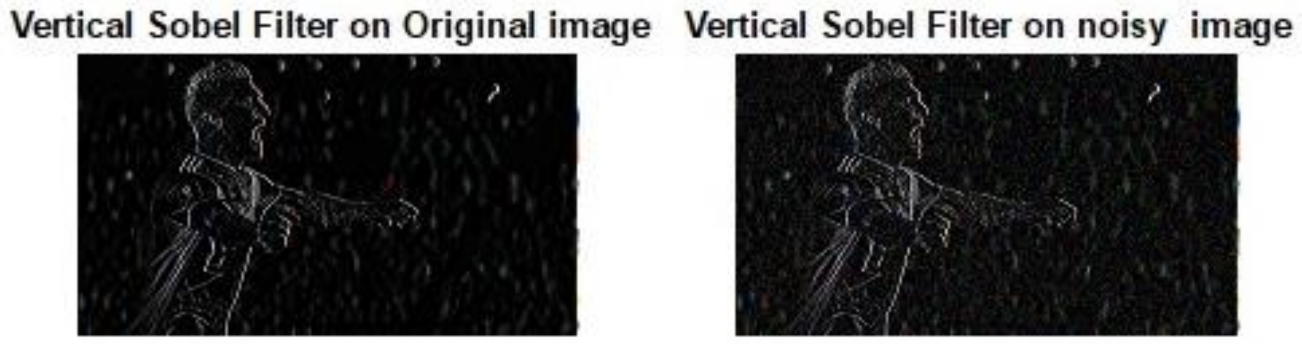

Vertical Sobel Filter on Original image
Vertical Sobel Filter on noisy image

Figure 19: Vertical Sobel filter on original and noisy image

From the figure 18 and the figure 19, we can observe that horizontal sobel filter highlights the horizontal edges and vertical filter enhances the vertical edges. Though, the sobel filter is not good for removing noise. Because sobel filter is a one kind of high pass filter and the main purpose of this filter to do edge enhancement, if we want to remove any noise then low pass filter is more appropriate

Exercise 3.3: Greyscale images

Extend the code with the functionality of converting color images into greyscale before filtering. Therefore, figure out a formula which maps color values into one greyscale value. Apply the formula on each pixel of the color image using two for loops.
Hint: to receive a correct image it is necessary to ensure all numbers being unsigned integers, hence use I=uint8(converted_image); as last command.

Answer:

Matlab code:

```
%% This Function will convert a color image into a grayscale image
function [gray] = RgbToGray(img)

R=img(:, :, 1)  % Extracting Red color element to R
G=img(:, :, 2)  % Extracting Green color element to G
B=img(:, :, 3)  % Extracting Blue colour component to  B

 [M, N, ~]=size(img); % Defining rows in M and column in N of RGB image
                % matrix
                     % size(img) function will return row, column and
% dimension of the RGB image

%% calculating grayscale values by forming a weighted sum of the R,G,and B elements for each pixel

for i=1:M
    for j=1:N
            % creating a new 2-d matrix 'gray' of size M*N of 'uint8'
            % data type to ensure all number being unsigned
            % The coefficients that are used to calculate grayscale                          % values are from
luminance (E’y) in Rec. ITU-R BT.601-7

            gray(i, j)=uint8((R(i, j)*0.2989)+(G(i, j)*0.5870)+(B(i,                                     j)*0.114));

           end
    end
end

>> img=imread('len_std.jpg');  % Read coloured input image
>> I=RgbToGray(img);  % Convert RGB image to gray scale image
>>imshow(I); % Show grayscale image
>> size(img)  % Show size of orginal RGB image

ans =

   256   256     3

>> size(I)% Show size of orginal RGB image

ans =

   256   256
```

Explanantion:

For orginal RGB image we need 3 bytes (24 bits) for each pixel. And when we convert an RGB image to grayscale image, we only need 1 byte (8 bit) to store a single pixel of the image. From the size of grayscale image, we can see that a grayscale image is represented by a two dimensional array of bytes. The size of the array considered to be the height and width of the image. On the other hand a RGB image is stored as an *m*-by-*n*-by-3 data array which defines red, green, and blue color elements for each and individual pixel.

Exercise 3.4: Moving average

Implement a filter which smoothes the image by averaging. For this define a parameter *n* which determines the size of a square block of (2n+1)x(2n+1) pixels. Shift this block pixel wise over the image and after each shift the center pixel of the block receives the value of the average over all pixel of the block. Apply an intelligent strategy for the edge regions where the block sticks out of this image area.

Answer:
Averaging is the procedure by using which the current pixel and its neighboring pixels are summed and their common is applied to the modern-day pixel. In this kind of filtering, all the pixels are given equal weightage. This may want to be disadvantageous in a way that if a particular pixel has greater intensity than its neighbors, with the aid of making use of averaging, the depth of the pixel is very a lot reduced.

MATLAB Code:

```
close all
clear all
clc


%List of images used
images=[{'cameraman.jpg'}, {'len_std.jpg'}];
image_index=2; %Pointer to choose image
greyscale=0; %Flag to set/reset grayscale


%% Image Filtering
I=imread(images{image_index}); %Command to read image

%%Black white conversation
if (greyscale==1)
G=rgb2gray(I);
figure(1)
subplot(1,2,1);imshow(I);title('Original Image')
subplot(1,2,2);imshow(G);title('Gray scale Image')
end

 %Logic to implement smoothing filter by averaging
imageSize=size(I); % Command to get the size of the image in row and column
n=3; %Size of a square block of pixels
I_smooth = I;
for i=n+1:imageSize(1)-(n+1) %Defining row width
for j=n+1:imageSize(2)-(n+1) %Defining column length
sum = 0;
for t=-n+1:n+1 %Block's row size
for r=-n+1:n+1 %Block's column size
sum = sum + int16(I(i+t,j+t)); %Summing the pixel values
end
end
I_avg=sum/((2*n+1)^2); %Average of the summed values
I_smooth(i,j) = I_avg; %Average of the values assigned to the centre pixel
end
end
figure(1)
subplot(1,2,1);imshow(I);title('Original Image')
subplot(1,2,2);imshow(I_smooth);title('Smoothed Image')
```

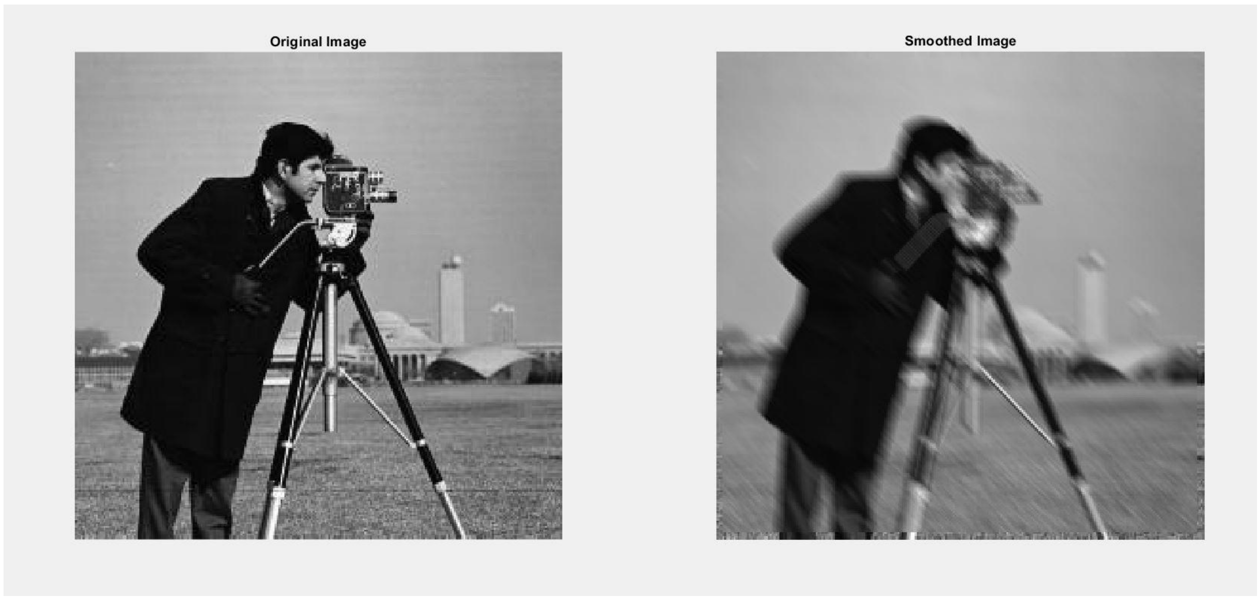


Figure 20: Original cameraman image and smoothened image using averaging.

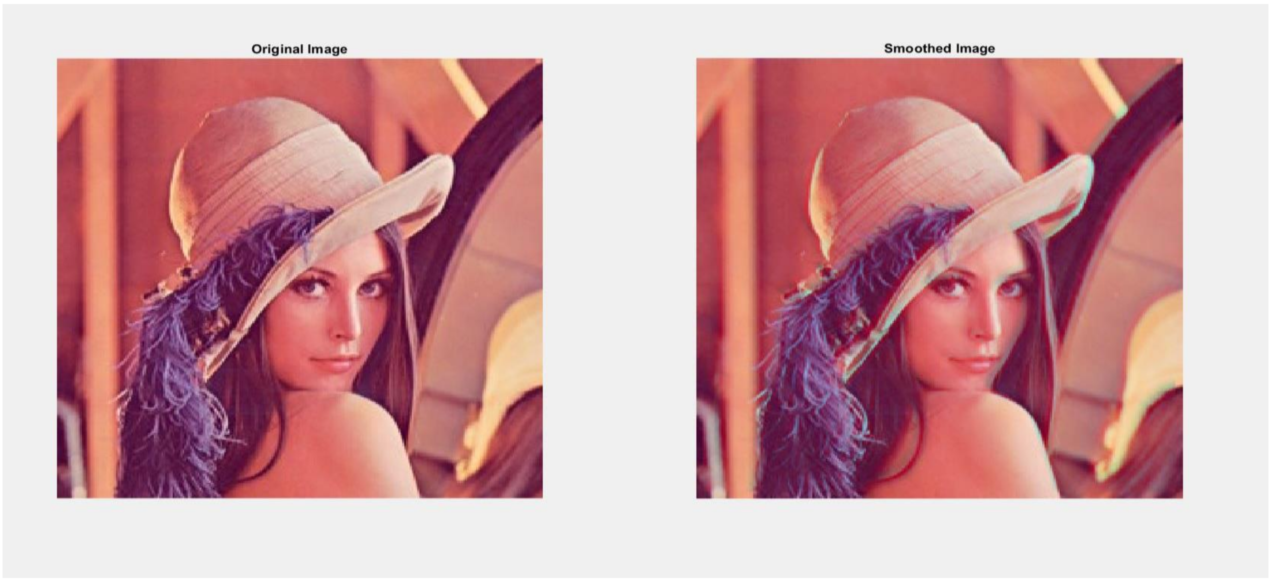


Figure 21: Original len_std image and smoothened image using averaging

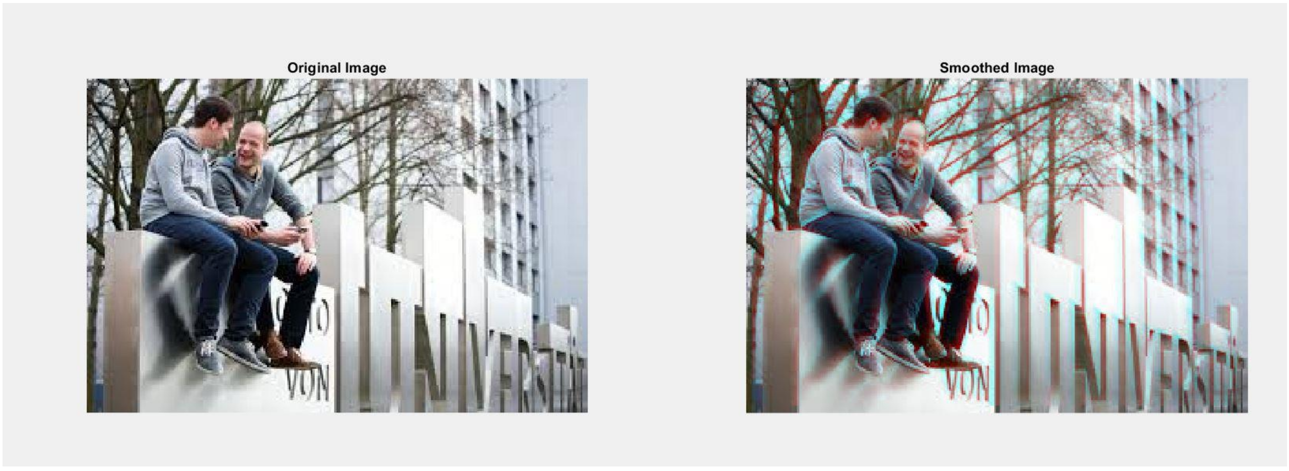


Figure 22: Original OvGU image and smoothened image using averaging

A picture is chosen from the list of images. Smoothening of the picture is carried by means of averaging method. Here, a block of dimension n = 3 (3x3 array of pixels) is taken. As the pointer moves on the block, the common of all pixels is calculated and utilized to the center pixel. Thus, the received image has the features of all its neighboring pixels which outcomes in smoothening of the image. In averaging, all pixels (including the modern pixel) are given equal weightage.

Figure 20 and 21 produces a certainly smoothened photo. But, when the same components are utilized on a grayscale image, it produces a blur as shown. This is because, in a colored image, all the neighboring color intensities are averaged and the resultant is a lowered color depth of the equal color. But in a grayscale image, there is both the dark part and the light area which when averaged produces reduced (by half) intensities of both the sections. These consequences in picture blur. Hence, for grayscale images, a Gaussian filter produces higher results. Figure 22 suggests the smoothening impact on ovgu.jpg image. Here, the effect of smoothening is not seen as much as the preceding pictures as almost the whole photograph has the same color depth.

Exercise 3.5: Further filters

Implement a further filter/effect according to your own taste.

Answer:

Motion blur is the streaking of shifting objects in a film (collection of sequence of frames) or in a photograph. This blur is due to the movement of an object when it is being recorded. For example, clicking a photo of a bus on road when it is moving, a picture of a person jumping etc.

In Motion blur, the complete image is blurred in a particular route and angle according to requirement. If the requirement is jumping, then consequently the angle and size of movement are set such that, all the objects in the photo appear blurred, showing the human eye viewpoint of the objects during movement. Motion blur can additionally be used for machine learning. In cases of hit-and-run vehicles, the license plate of the car in the recorded picture would be too blurred for a human eye. In such cases, one has to process the image manually and parent out the license variety for in addition tracking. But, say that a robot of some sort is concerned in the tracking, and then such action blurred pictures can be fed to the robot, for it to comprehend what manner has to be carried out next to track the vehicle.

MATLAB Code:

```
close all
clear all
clc
%List of images used
images=[{'cameraman.jpg'}, {'len_std.jpg'}];
image_index=1; %Pointer to choose image

%% Image Filtering
I=imread(images{image_index}); %Command to read image
H = fspecial('motion',20,0); %Motion filter creation, fspecial('motion', len,theta);
%len - length of motion, theta - angle of motion
I_filt= imfilter(I,H); %Adding filter to original image
figure(1)
subplot(1,2,1)
imshow(I);title('Original Image') %Plotting original image
subplot(1,2,2);imshow(I_filt);title('Motion Filter on original image') %plotting filtered original image
```

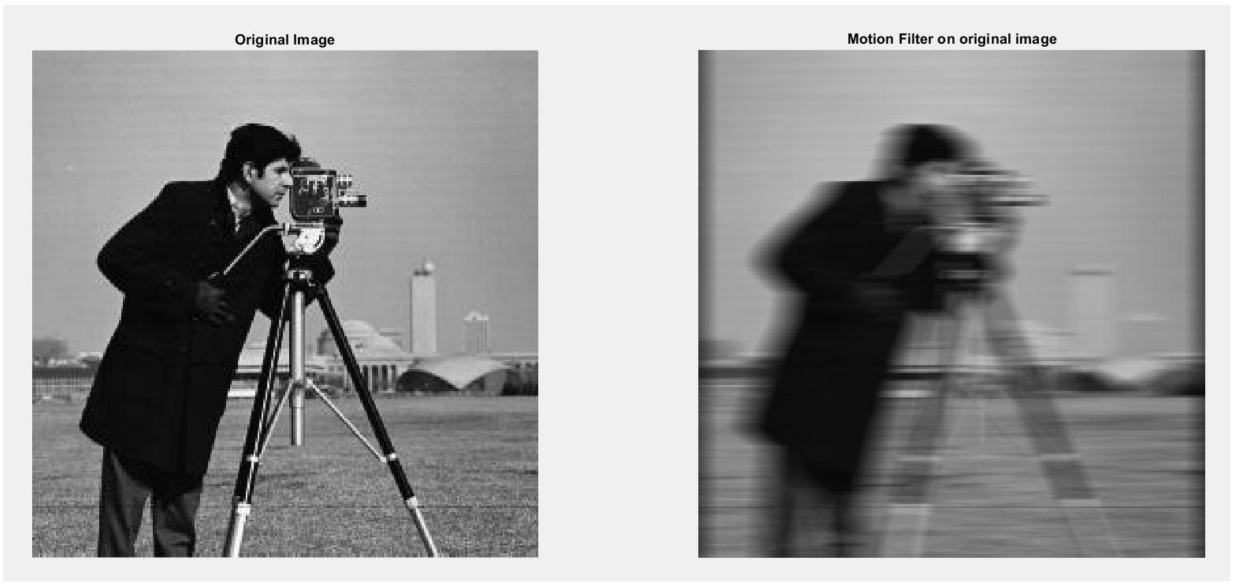


Figure 23: Original Cameraman image and motion blurred image with len = 20, theta = 0.

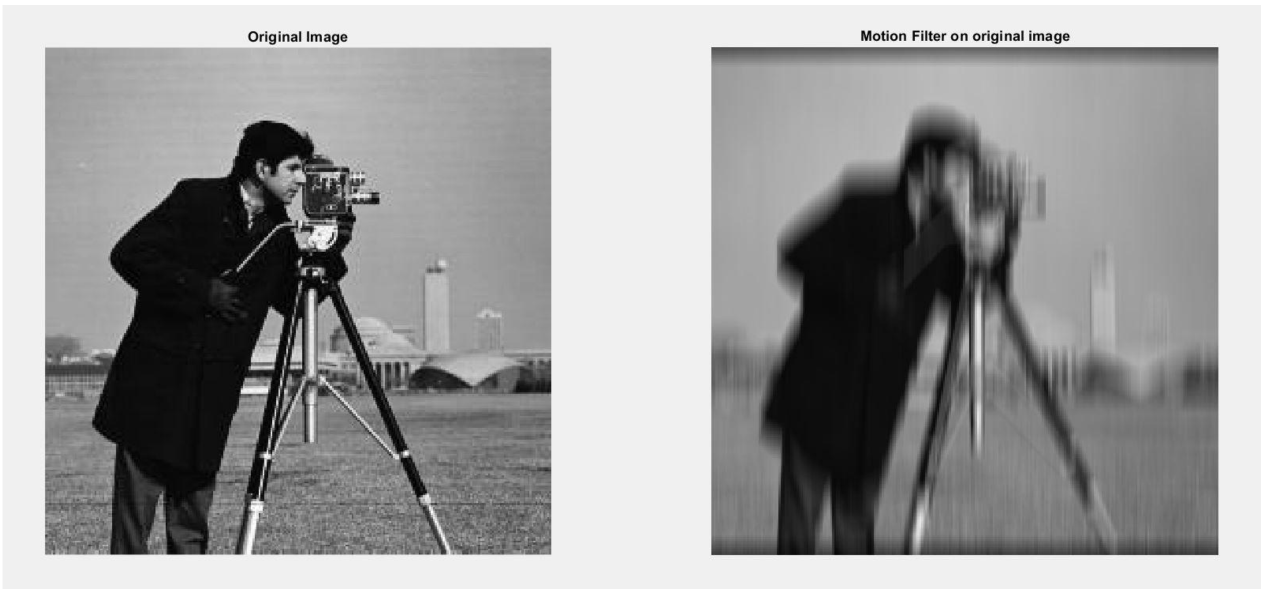


Figure 24: Original Cameraman image and motion blurred image with len = 20, theta = 90

Figure 23 shows motion blurred image in horizontal direction. The cameraman in the image seems to be moving in the horizontal direction. In Figure 24, shows motion blurred image in vertical direction. The cameraman in the image seems to be moving in the vertical direction.

4. References

[1] Nishant_KumarCheck out this Author's contributed articles., Nishant_Kumar, and Check out this Author's contributed articles., "Digital Image Processing Basics," *GeeksforGeeks*, 06-Feb-2018. [Online]. Available: https://www.geeksforgeeks.org/digital-image-processing-basics/. [Accessed: 16-Jun-2020].

[2] "Digital Image Processing Introduction," *Tutorialspoint*. [Online]. Available: https://www.tutorialspoint.com/dip/image_processing_introduction.htm. [Accessed: 16-Jun-2020].

[3] *Images and Channels*. [Online]. Available: http://www.georeference.org/doc/images_and_channels.htm. [Accessed: 14-Jun-2020].

[4]"Grayscale Images," *Glossary - Grayscale Images*. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gryimage.htm. [Accessed: 15-Jun-2020].

[5] "Three algorithms for converting color to grayscale," *John D. Cook*, 24-Aug-2009. [Online]. Available: https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/. [Accessed: 14-Jun-2020].

[6] *Filtering an Image*, 16-Jun-2005. [Online]. Available: https://northstar-www.dartmouth.edu/doc/idl/html_6.2/Filtering_an_Imagehvr.html#wp1022750. [Accessed: 02-Jul-2020].

[7] Geek Bit of Everything "Gaussian filter implementation in Matlab for smoothing images (Image Processing Tutorials)" *YouTube*, Dec. 4, 2077 [Video file].
Available: https://www.youtube.com/watch?v=LZRiMS0hcX4&fbclid=IwAR2o22LmM_X0f4-BEY6s_K0-Md7eLucPLxnBHapcqO8BokXNinc5Lu7cKp8 [Accessed: 2-July-2020].