.

TITLE OF PROJECT | IMAGE & AUDIO DIGITAL PROCESSING

COURSE | SIGNALS & SYSTEMS (ELC 2030)

SUPERVISOR | DR. MICHAEL MELEK

STUDENTs

1. ABDELRAHMAN MOHAMED SALAH
2. MOHAMED ESSAM ABDELAZEEM

IDs

1. 9220473
2. 9220720

Abstract

In this report, we present our work on the signals project supervised by Dr.Michael Melek which concentrated on some concepts in the Signals & Systems course, especially the Fourier Transform and discrete Fourier Transform represented in the Fast Fourier Transform.The concepts of convolution, signal transmission, and modulation were also strongly planted in the project's core.

We worked on this project in two directions, which were image Filtration& Restoration and communication system simulation. In the first direction, we aimed to do some specific filtration processes on a specific image and restore the image after a specific filter. In the other direction, we aimed to record our voice, apply a filter to it, and then transmit it to a higher frequency and get it back.

We can say that the skills gained from practicing Matlab through such a practical project, in addition to writing the report in latex, were of invaluable benefit to us in various ways.

1 Image Filtration and restoration

We did this part with the help of the following built-in functions:

- conv2 [1]
- fft2
- ifft2
- im2double[2]
- imshow[3]
- subplot [4]
- zeros
- ones
- rgb2gray

1.1 Getting the three color components of the image

1.1.1 Working idea

From the previously known information every image consists of three images of different colors (Red, Green, and Blue) concatenated at each other, We extracted each component alone and concatenated it with the other two black components to get full three images of different colors forming the original image.[5]

1.1.2 The processed images

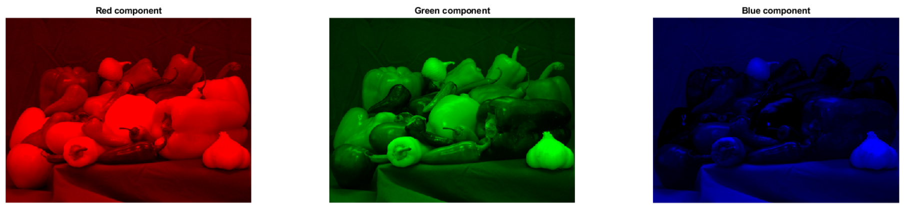

Figure 1: The three components of the color

1.2 Edge Detection

In edge detection, we decided to work by convolving the image with the Laplacian kernel, as the Laplacian is a measure of the spacial derivative of an image which can detect the great changes in the pixel value of an image and get the edges.[6]

1.2.1 How to apply the kernel?

We should conserve the realization of some conditions:

- The kernel is a square Matrix.
- The sum of kernel elements equals zero.
- We can apply the kernel on the gray image or each component of the image and concatenate them.

1.2.2 The Kernel used

$$\begin{pmatrix} 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & 16 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

1.2.3 The processed images

1.3 Sharpening

1.3.1 The working idea

The working idea of the kernel used to perform the sharing is so simple that you add the edges you got from the edge-detected image and add to the original so that you can get the edges sharpened.

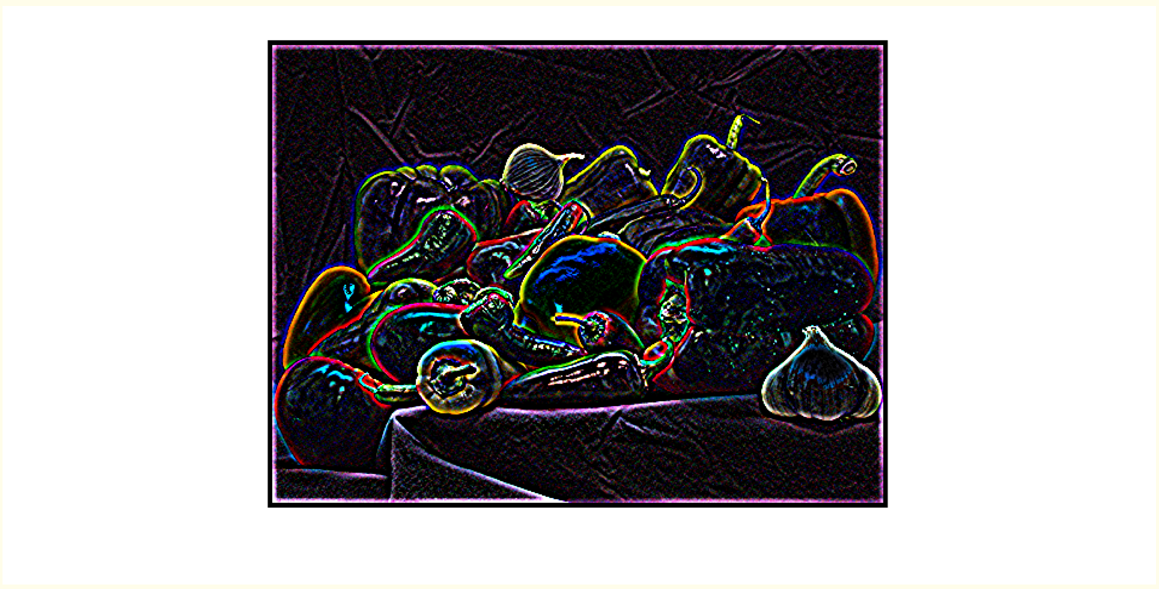

Figure 2: Edge Detection for colored images

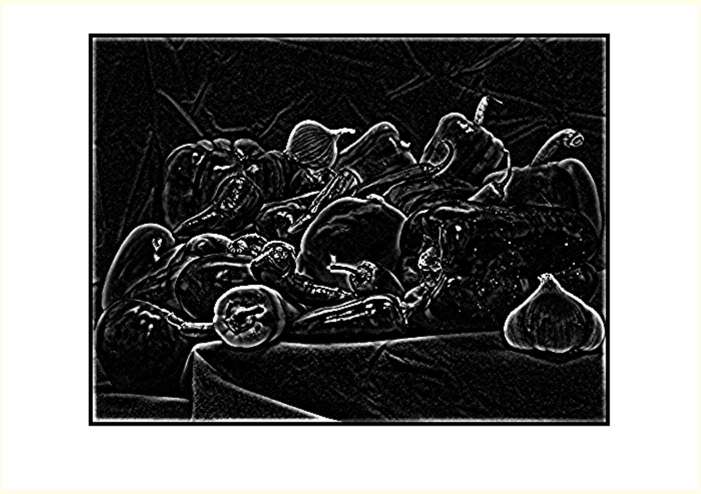

Figure 3: Edge Detection for gray images

1.3.2 The Kernel used

$$\begin{pmatrix} 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & 17 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

1.3.3 The processed images

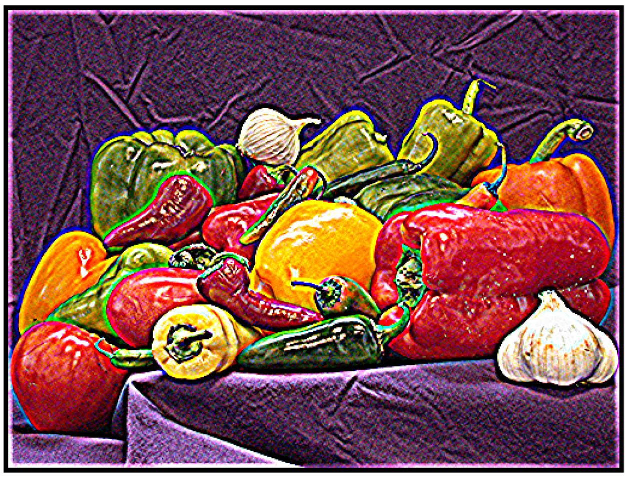

Figure 4: Sharpening colored images

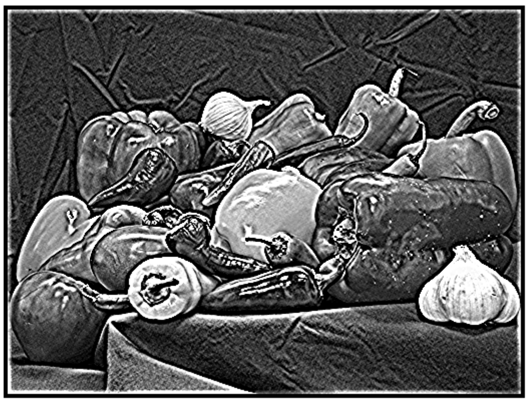

Figure 5: Sharpening gray images

1.4 Bluring

1.4.1 The working idea

Bluring basically means averaging, as when you want to blur a photo, you want to make it smoother, less detailed, and not sharpened. So, you just can make every pixel dependent on the value of the pixels around it.

We can make the dependency of the pixel on other pixels dependent also on the distance between the two pixels[7], but we will just be satisfied by the simple averaging kernel. The averaging kernel is characterized that we divide the whole kernel by the sum of its elements.

1.4.2 The Kernel used

$$\frac{1}{100} * \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

1.4.3 The processed images

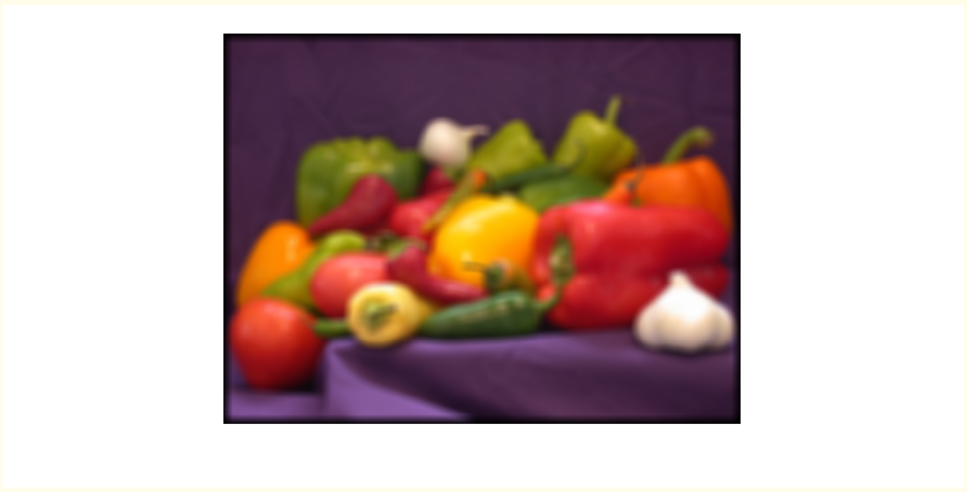

Figure 6: Bluring colored images

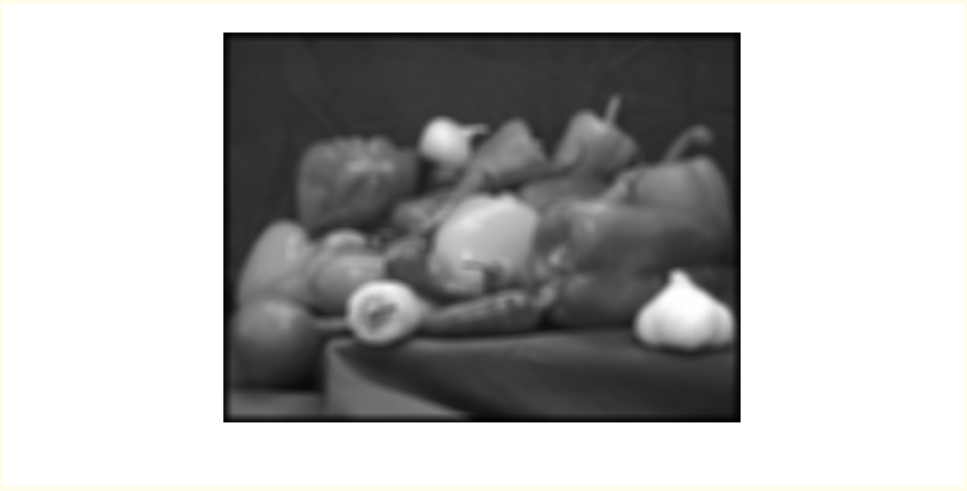

Figure 7: Bluring gray images

1.5 Motion Bluring

1.5.1 The working idea

By intuition, we can say that motion blurring has the idea of normal averaging, but we make every pixel depend on the pixels in the same direction of motion instead of all directions. The kernel can only be a vector of all ones divided by their number and directed in the direction we want the motion in.

1.5.2 The Kernel used

$$\frac{1}{50} * \text{ vector of length (50) containing only ones}$$

1.5.3 The processed images

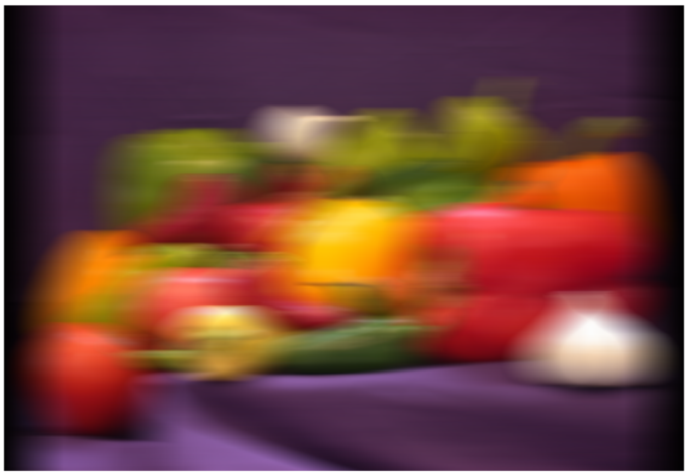

Figure 8: Motion bluring colored images

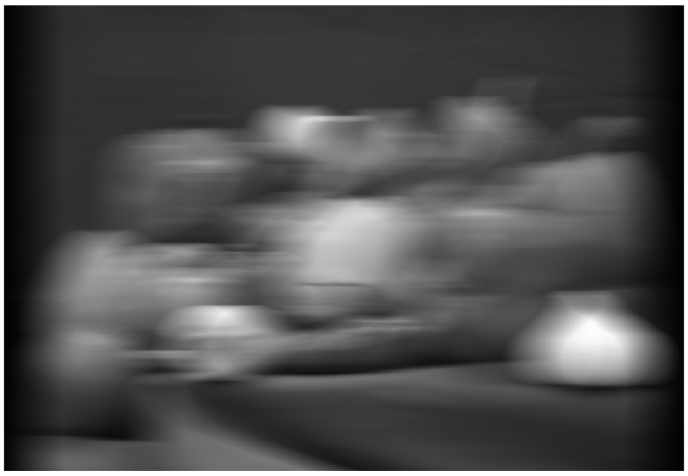

Figure 9: Motion bluring gray images

1.6 Restoring the image from motion blurring

1.6.1 Working idea

The idea of restoring the original image is based on the Fourier Transform concept to reflect the impact of convolution, we can't do this in the time domain easily so, we have to convert to the frequency domain as convolution turns into multiplication in frequency domain, so we can reflect the impact of convolution by using division.

1.6.2 the restoring process

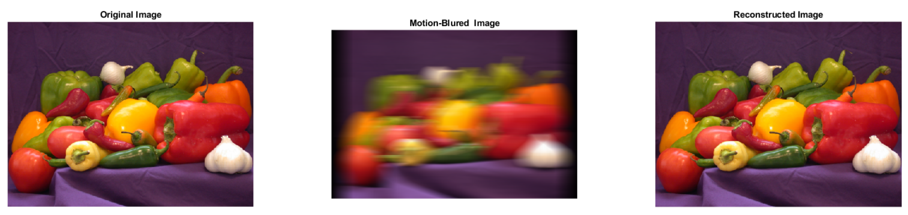

Figure 10: Reconstructing colored images

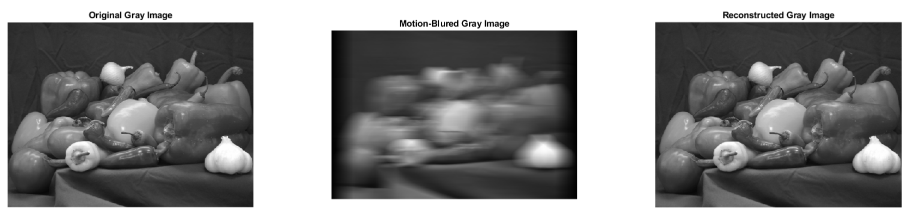

Figure 11: Reconstructing gray images

2 Communication System Simulation

2.1 Recording the voice

In order for the voice to be of good quality, the sampling rate must be more than double the frequency of the sound that humans can hear (which ranges from 20 Hz to 20 kHz). So, we chose a sampling rate of 44100 Hz, which is a widely used number worldwide.

We can use a higher sampling rate to get more quality sound, such as 96 kHz or 192 kHz, which are widely used in higher-quality communication systems.

In low-quality voice, 8-bit is widely used but to make the voice of an acceptable quality range, we used 16-bit depth. For higher quality, we use 24-bit or 32-bit depth.[8]

2.2 Filtering the voice

We used the Butterworth filter because it has no ripples[9]. At first, we used $F_{pass} = 8000Hz$ and $F_{stop} = 8500Hz$, but it interfered with the communication part as MATLAB has a limited range in the carrier, so we dropped the value a little to make sure the whole communication system works well, and also decreasing the frequency doesn't affect the voice so much as the frequency of the human most power exists in the range of 300 to 3400 Hz. So, we used $F_{pass} = 6000Hz$ and $F_{stop} = 6500Hz$.[10]

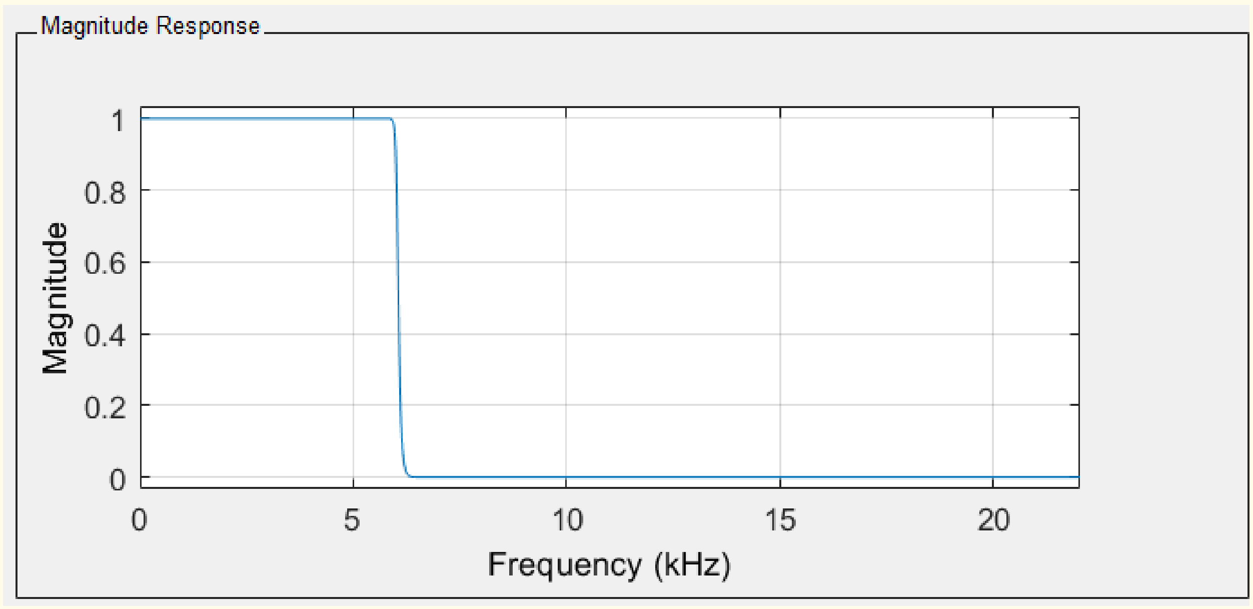

Figure 12: Magnitude vs Frequency

2.3 Plotting the spectrum of the signals

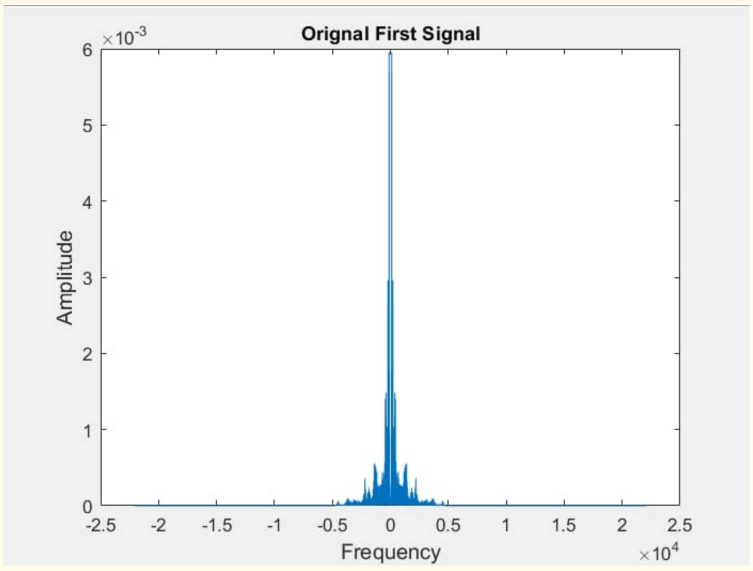

Figure 13: original first signal

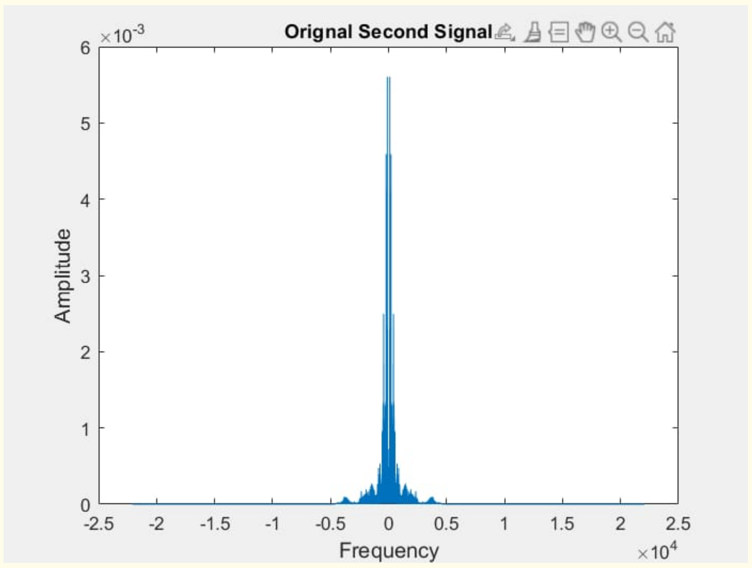

Figure 14: original second signal

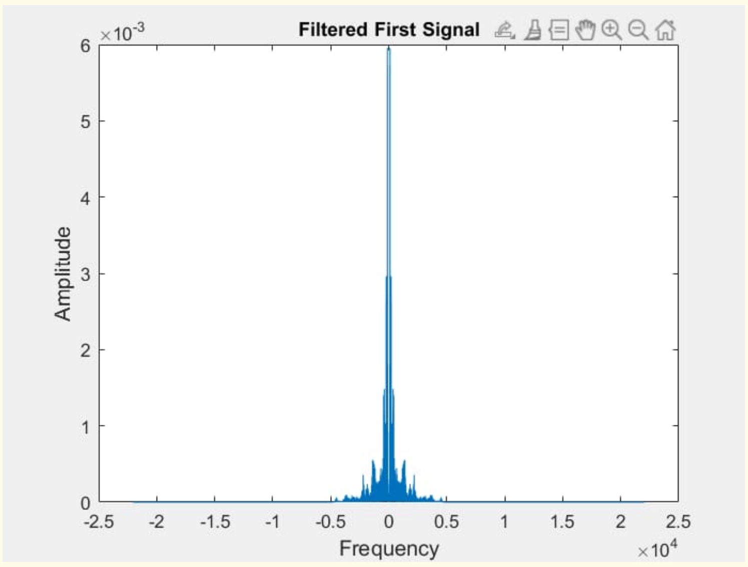

Figure 15: Filtered first signal

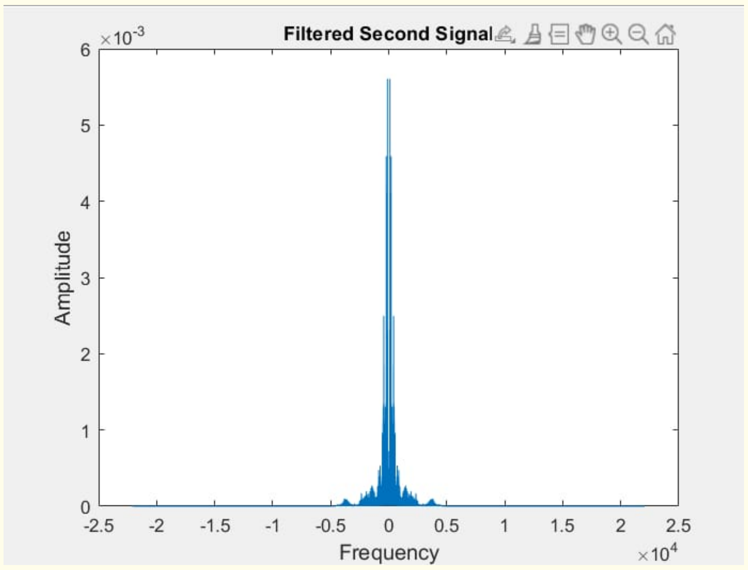

Figure 16: Filtered second signal

2.4 Modulation

In carrier, we used the frequency values $F_{c1} = 5500Hz$ and $F_{c2} = 16000Hz$ to make the two frequencies away from each other to not interfere and could filter them by non-ideal filter easily[11]. Also, MATLAB can't put the signal on the higher carrier so we don't have a lot of frequency space here[12]

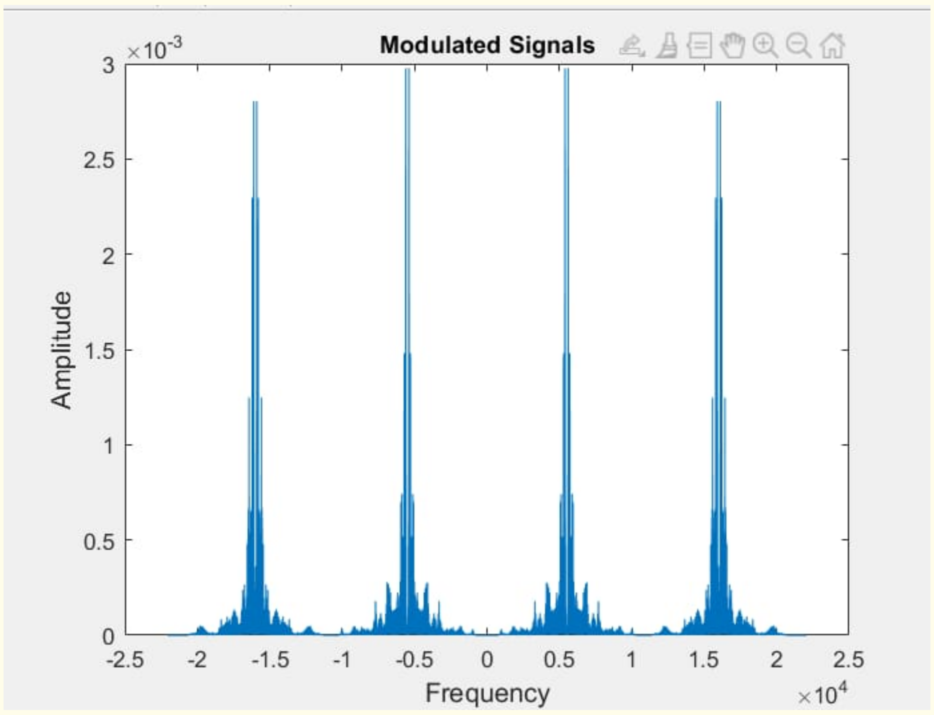

Figure 17: Modulated signals

2.5 Demodulation

In the receiver stage, first we need to separate the two signals by applying a bandpass filter to separate the two signals (demultiplexing).the first bandpass frequencies valuesare $F_{stop1} = 100Hz, F_{pass1} = 1100Hz, F_{pass2} = 10000Hz, F_{stop2} = 15000Hz$ and the second one values were $F_{stop1} = 10200Hz, F_{pass1} = 11400Hz$,$F_{pass2} = 20800Hz, F_{stop2} = 22000Hz$

second, we multiply the signal with the carrier signal,then apply a low pass filter with gain of 2 to restore original audios(demodulation). the signal operates in different domains:

1. time domain:

$$r(t) = y(t) * c(t)$$

   Where, r(t): received signal, y(t): the signal came from bandpass filter, c(t): the carrier

2. Frequency domain:

$$R(\omega) = \frac{Y(\omega - \omega_o) + Y(\omega + \omega_o)}{2}$$

   - R(ω): received signal
   - Y(ω): the signal came from bandpass filter

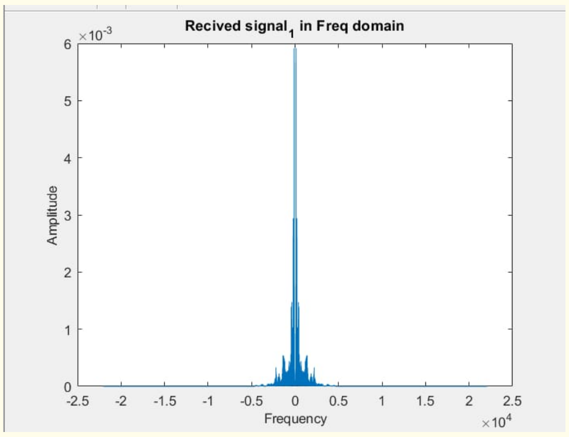

Figure 18: Received first signal in frequency domain

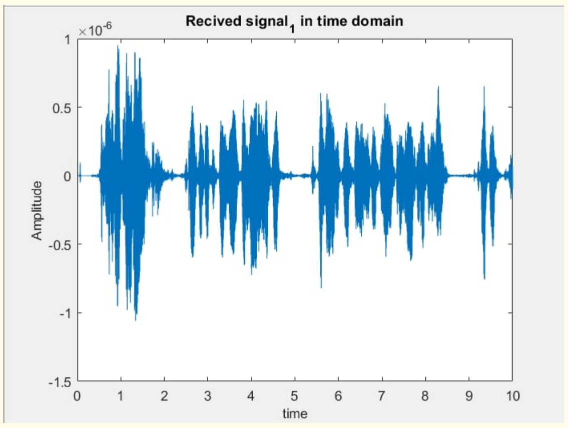

Figure 19: Received first signal in time domain

A References

A.1 Bibliography

[1] 2-d convolution - matlab conv2. https://www.mathworks.com/help/matlab/ref/conv2.html, 2024. Accessed: 2024-01-02.

[2] Convert image to double precision - matlab im2double. https://www.mathworks.com/help/matlab/ref/im2double.html, 2024. Accessed: 2024-01-02.

[3] Display image - matlab imshow. https://www.mathworks.com/help/images/ref/imshow.html?searchHighlight=imshow&s_tid=srchtitle_support_results_1_imshow, 2024. Accessed: 2024-01-02.

[4] Create axes in tiled positions - matlab subplot. https://www.mathworks.com/help/matlab/ref/subplot.html?searchHighlight=subplot&s_tid=srchtitle_support_results_1_subplot, 2024. Accessed: 2024-01-02.

[5] r g b components of an image - matlab answers - matlab central. https://www.mathworks.com/matlabcentral/answers/90908-r-g-b-components-of-an-image, 2024. Accessed: 2024-01-02.

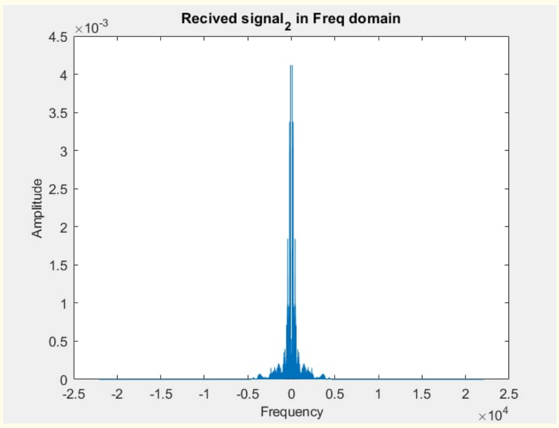

Figure 20: Received second signal in frequency domain

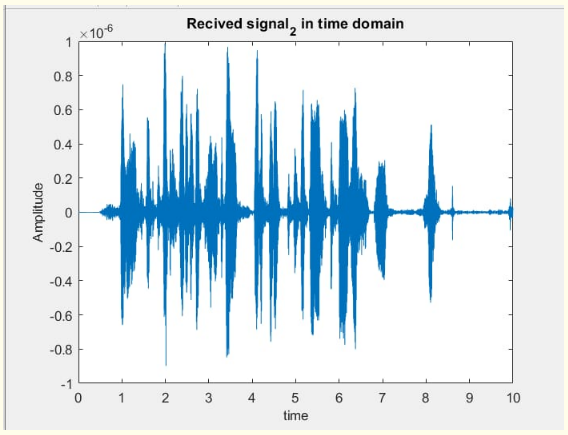

Figure 21: Received second signal in time domain

[6] Spatial filters - laplacian/laplacian of gaussian. https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm, 2024. Accessed: 2024-01-02.

[7] Blurring an image — apple developer documentation. https://developer.apple.com/documentation/accelerate/blurring_an_image, 2024. Accessed: 2024-01-02.

[8] (98) how to record audio in matlab (with complete code) - youtube. https://www.youtube.com/watch?v=C05tMF2kvpM, 2024. Accessed: 2024-01-02.

[9] 1-d digital filter - matlab filter. https://www.mathworks.com/help/releases/R2021a/matlab/ref/filter.html, 2024. Accessed: 2024-01-02.

[10] (98) design fir in matlab — fir iir filters in matlab - youtube. https://www.youtube.com/watch?v=Jf9NvCU6siU, 2024. Accessed: 2024-01-02.

[11] Transpose vector or matrix - matlab transpose .'. https://www.mathworks.com/help/releases/R2021a/matlab/ref/transpose.html, 2024. Accessed: 2024-01-02.

[12] (98) amplitude modulation - matlab tutorial (amplitude modulation in matlab with code) 2016 - youtube. https://www.youtube.com/watch?v=dEG_hv5E8VQ, 2024. Accessed: 2024-01-02.

B Matlab code for image processing

```
peppers_img=imread('peppers.png');

%----------------------------------------------------------------------

%Extracting the three color components

red_component=peppers_img(:,:,1);%red component of the image
green_component=peppers_img(:,:,2);%green component of the image
blue_component=peppers_img(:,:,3);%red component of the imageSeS

%--------------------------------------------------------------------

%Creating a black image

Black= zeros(size(peppers_img,1),size(peppers_img,2));

%--------------------------------------------------------------------

%Creating colored images of the image

red_image =cat(3,red_component,Black,Black);
green_image =cat(3,Black,green_component,Black);
blue_image =cat(3,Black,Black,blue_component);

%--------------------------------------------------------------------

%Displaying color components

subplot(1,3,1)
imshow (red_image)
fontsize= 18;
title('Red component');
subplot(1,3,2)
imshow (green_image)
fontsize= 18;
title('Green component');

subplot(1,3,3)
imshow (blue_image)
fontsize= 18;
title('Blue component');

%----------------------------------------------------------

%getting a gray image out of the original one
grey_image=rgb2gray(peppers_img);
```

```matlab

%---------------------------------------------------------

%Converting image components into double
 Red_double=im2double(red_component);
 Blue_double=im2double(blue_component);
 Green_double=im2double(green_component);
 Gray_double= im2double(grey_image);

 %--------------------------------------------------------

%Edge Detection
Edge_Detection_kernel=[
    0   0   0    0    -1  0   0   0  0
    0   0   0    0,  -1  ,0   0   0  0 ;
    0   0   0    0   -1   0   0   0  0;
    0   0   0    0   -1   0   0   0  0;
   -1  -1  -1   -1   16  -1  -1  -1 -1;
    0   0   0    0   -1   0   0   0  0 ;
    0   0   0    0   -1   0   0   0  0 ;
    0   0   0    0   -1   0   0   0  0;
    0   0   0    0   -1   0   0   0  0 ];


%-------------------------

%Edge detection for a gray image

Gray_edge_detection = conv2( Gray_double,Edge_Detection_kernel);
figure;
imshow(Gray_edge_detection);

%-------------------------
%Edge detection for the original image
figure;
Red_edge_detectioned_image=conv2(Red_double,Edge_Detection_kernel);
Blue_edge_detectioned_image=conv2(Blue_double,Edge_Detection_kernel);
Green_edge_detectioned_image=conv2(Green_double,Edge_Detection_kernel);

Img_processed =cat(3,Red_edge_detectioned_image,Green_edge_detectioned_image,
    Blue_edge_detectioned_image);
imshow(Img_processed);


%-------------------------------------------------------


%Sharpining image

```

```
Sharpening_Kernel =[
    0   0   0    0   -1   0   0   0  0 ;
    0   0   0    0,  -1  ,0   0   0  0 ;
    0   0   0    0   -1   0   0   0  0 ;
    0   0   0    0   -1   0   0   0  0 ;
   -1  -1  -1   -1   17  -1  -1  -1 -1 ;
    0   0   0    0   -1   0   0   0  0 ;
    0   0   0    0   -1   0   0   0  0 ;
    0   0   0    0   -1   0   0   0  0 ;
    0   0   0    0   -1   0   0   0  0 ];


%----------------------------

%%Sharpening the gray image

Gray_sharpened_image = conv2( Gray_double,Sharpening_Kernel);
figure;
imshow(Gray_sharpened_image);

%------------------------------

%%Sharping the original image

figure;
Red_sharpened_image=conv2(Red_double,Sharpening_Kernel);
Blue_sharpened_image=conv2(Blue_double,Sharpening_Kernel);
Green_sharpened_image=conv2(Green_double,Sharpening_Kernel);

Sharpened_image =cat(3,Red_sharpened_image,Green_sharpened_image,Blue_sharpened_image);
imshow(Sharpened_image);


%--------------------------------------------------------------


% Bluring image

Gz=(1/100 )*ones(10,10);

%------------------------

%Bluring the gray image
Blured_gray_image = conv2( Gray_double,Gz);
figure;
imshow(Blured_gray_image);

%-------------------------

```

```
%Bluring the original image

figure;

Blured_red_image=conv2(Red_double,Gz);
Blured_blue_image=conv2(Blue_double,Gz);
Blured_green_image=conv2(Green_double,Gz);

Blured_image =cat(3,Blured_red_image,Blured_green_image,Blured_blue_image);
imshow(Blured_image);


%------------------------------------------------------

%Motion bluring

Motion_bluring_kernel=(1/50)*ones(1,50)

%---------------------------

%Motion blur of gray image

Motion_blured_gray_image = conv2( Gray_double,Motion_bluring_kernel);
figure;
imshow(Motion_blured_gray_image);

%----------------------------

%Motion blur of the original image


figure;

Motion_blured_red_image=conv2(Red_double,Motion_bluring_kernel);
Motion_blured_blue_image=conv2(Blue_double,Motion_bluring_kernel);
Motion_blured_green_image=conv2(Green_double,Motion_bluring_kernel);

Motion_blured_image =cat(3,Motion_blured_red_image,Motion_blured_green_image,
    Motion_blured_blue_image);
imshow(Motion_blured_image);

%--------------------------------------------------------------

%Adjusting the size of the motion_bluring kernel to suit the divisioin in
%frequency domain

The_widened_motion_bluring_kernel=zeros(size(Motion_blured_image,1),size(Motion_blured_image,2));
The_widened_motion_bluring_kernel(1,1:50)=Motion_bluring_kernel;

```

```matlab

%-------------------------------------------------------------------

%Reconstructing the image from the motion-blured one in frequency domain

%Converting into Fourier domain

Frequency_domain_motion_blured_kernel=fft2(The_widened_motion_bluring_kernel);
Frequency_domain_motion_blured_red_image=fft2(Motion_blured_red_image);
Frequency_domain_motion_blured_blue_image=fft2(Motion_blured_blue_image);
Frequency_domain_motion_blured_green_image=fft2(Motion_blured_green_image);
Frequency_domain_motion_blured_gray_image=fft2(Motion_blured_gray_image);

%---------------------------------------


%Getting the original image in fourier domain


reconstructed_red_in_frequendy_domain=Frequency_domain_motion_blured_red_image./
    Frequency_domain_motion_blured_kernel;
reconstructed_blue_in_frequency_domain=Frequency_domain_motion_blured_blue_image./
    Frequency_domain_motion_blured_kernel;
reconstructed_green_in_frequency_domain=Frequency_domain_motion_blured_green_image./
    Frequency_domain_motion_blured_kernel;
reconstructed_gray_in_frequency_domain=Frequency_domain_motion_blured_gray_image./
    Frequency_domain_motion_blured_kernel;

%-----------------------------------------

%Getting inverse Fourier Transform of the image

Reconstructed_red=ifft2(reconstructed_red_in_frequendy_domain);
Reconstructed_green=ifft2(reconstructed_green_in_frequency_domain);
Reconstructed_blue=ifft2(reconstructed_blue_in_frequency_domain);
Reconstructed_gray=ifft2(reconstructed_gray_in_frequency_domain);

Reconstructed_gray=Reconstructed_gray(1:384,1:512,:);
%------------------------------------------

%The reconstructed image

Reconstructed_image=cat(3,Reconstructed_red,Reconstructed_green,Reconstructed_blue);

Reconstructed_image=Reconstructed_image(1:384,1:512,:);

%--------------------------------------

%displaying the original image besides the motion_blured and reconstructed
```

```
%one

figure;
subplot(1,3,1);
imshow (peppers_img);
fontsize= 18;
title('Original Image');

subplot(1,3,2);
imshow (Motion_blured_image);
fontsize= 18;
title('Motion-Blured  Image');

subplot(1,3,3);
imshow (Reconstructed_image);
fontsize= 18;
title('Reconstructed Image');

figure;
subplot(1,3,1);
imshow (grey_image);
fontsize= 18;
title('Original Gray Image');

subplot(1,3,2);
imshow (Motion_blured_gray_image);
fontsize= 18;
title('Motion-Blured Gray Image');

subplot(1,3,3);
imshow (Reconstructed_gray);
fontsize= 18;
title('Reconstructed Gray Image');


%-----------------------------------------------------------------------%

%--------------------Thank  you doctor Michael----------------------------%
```

C Matlab code for Simulating Communication Systems

```matlab
%% check no error
clc;
clear ;
close all;
%to make sure that no previous data interupt and no error happended and
%clear the command window
%% variable setting
ch=1;%Number of channels--2 options--1 (mono) or 2 (stereo)
Fs=44100;%Sampling frequency in hertz
datatype='int16';
Nseconds=10;
nbits=16;%8,16,or 24

% as the doctor say in the first lecture sampling time for telefon is 8khz
% and in music in CDs is 44.1 khz
% if you want a high quality resultion audio sampling frequency would be
% 96khz or 196khz

% the numbers of bit used in quantzion are 8 bits for low quality voices
% like in telefon and 16 bit for CDs used in music as shown in the first
% lecture of the term and also there are 24 and 32 bits which used in high
% quality resulatuion and professional

%% recording audio and saving it
recorder=audiorecorder(Fs,nbits,ch);
disp('Start speaking..')
%Record audio to audiorecorder object,hold control until recording completes
recordblocking(recorder,Nseconds);
disp('End of Recording.');
%Store recorded audio signal in numeric array
x=getaudiodata(recorder,datatype); % datatype could be int16 or uint8 or int8 or
%Write audio file
audiowrite('input1.wav',x,Fs);

%% Ex
% first voice parameteres fs 44100 hz and 16 bit for quantization with no
% noise

% second voice as the previous but external noise is 25000 hz sound


% after trying some prameters i choose this to make the sound clear and not
% with high space
```

D Names and codes in Arabic

ID	Section	الأسم
9220720	3	محمد عصام عبد العظيم ابراهيم
9220473	2	عبدالرحمن محمد صلاح الدين أبوهندي