Cairo University

Faculty of Engineering

Dept. of Electronics and Electrical Communications

Second Year

Signals Project

Report

Student Name	Section	B.N.
كريم عاطف نجيب سركيس	3	9220596
عمرو وائل ليثي محمد عثمان	3	9220566

Task 1(Image Processing)

Brief of the work

-First, we extracted the RBG components from the original and plotted each of the three as an image, then in edge filtering, we decided to use the Sobel kernel on the grayscale image, by convoluting it vertically and horizontally and getting the total of both.

-In sharpening we are more interested in the colored image, so we convolute the chosen kernel (which make the contrasts and edges noticeable) with each of the RGB components then concatenating them together for the final sharpened image.

-In the averaging section, we do the same as above but the kernel used is a matrix of ones multiplied by its average, which is used to hide the details of the image

-in the motion blurring, we are only blurring horizontally, so the kernel used is a row of ones with varying sizes depending on the amount of blurring we need, convoluting it with each component then concatenating them together for the final image.

-to restore the original image from the motion blurred, we transform both the image and the kernel to frequency domain, and we adjust the size of the kernel to fit the image, then we add a small constant to the matrix to avoid division by 0, after that we divide the image by the kernel, then transform the image back to time domain.

Outputs

a) RGB Components of the given image

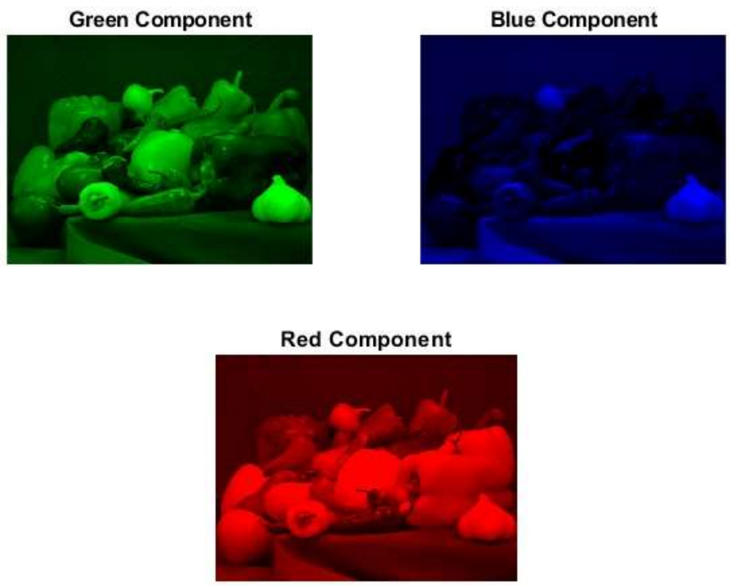


Fig (1): RGB

b.1) Edge detection

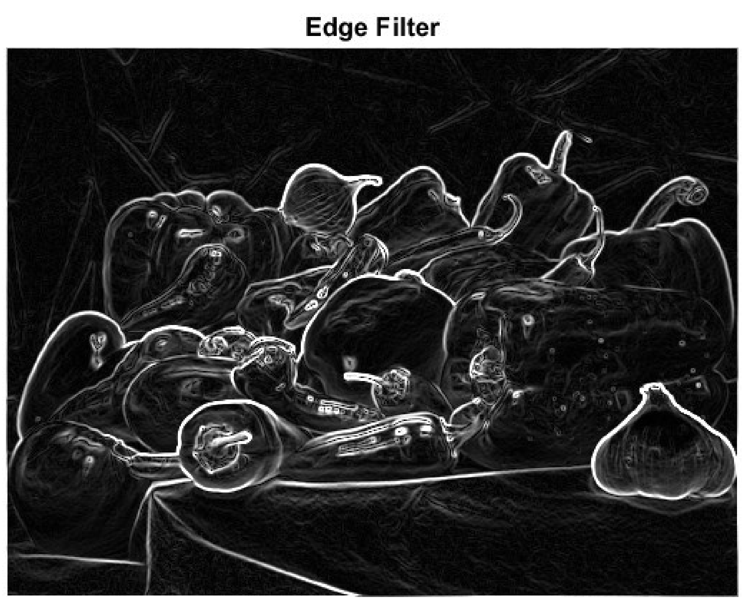


Fig (2): Edge

- **Regarding this, we chose the Sobel kernel which is mostly common for edge detection, this kernel shows the high frequency parts in the image that correspond to the edges needed by finding the gradient magnitude at each image.**

b.2) Sharpened (Enhanced) Image

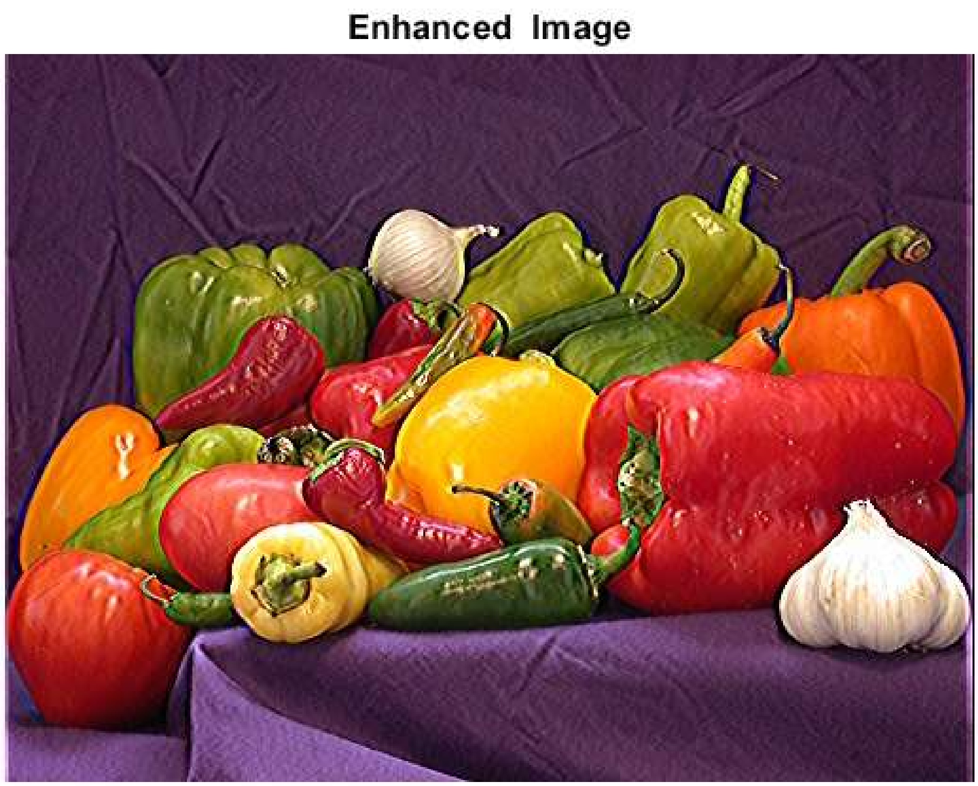


Fig (3): Enhanced Image.

- **The kernel we chose sharpens the image by making the edges more noticeable specially the more bright and darker parts, which in a way is close to the edge detection while keeping the original colors.**

b.3) Blurred Image

Blurred Image

Fig (4): Blurred Image.

- We chose a 3x3 matrix of ones and dividing by 9 so that each pixel is averaged around its closer ones (we could increase the size so that the blurring shows better)

b.4) Motion Blurred Image

Motion Blurred Image

Fig (5): Motion Blurred Image.

- We need to blur in one direction only which is horizontal, so we have a row of ones, we increased its size 1/16 so that the blurring is noticeable than the average one.

C) Deblurred Image

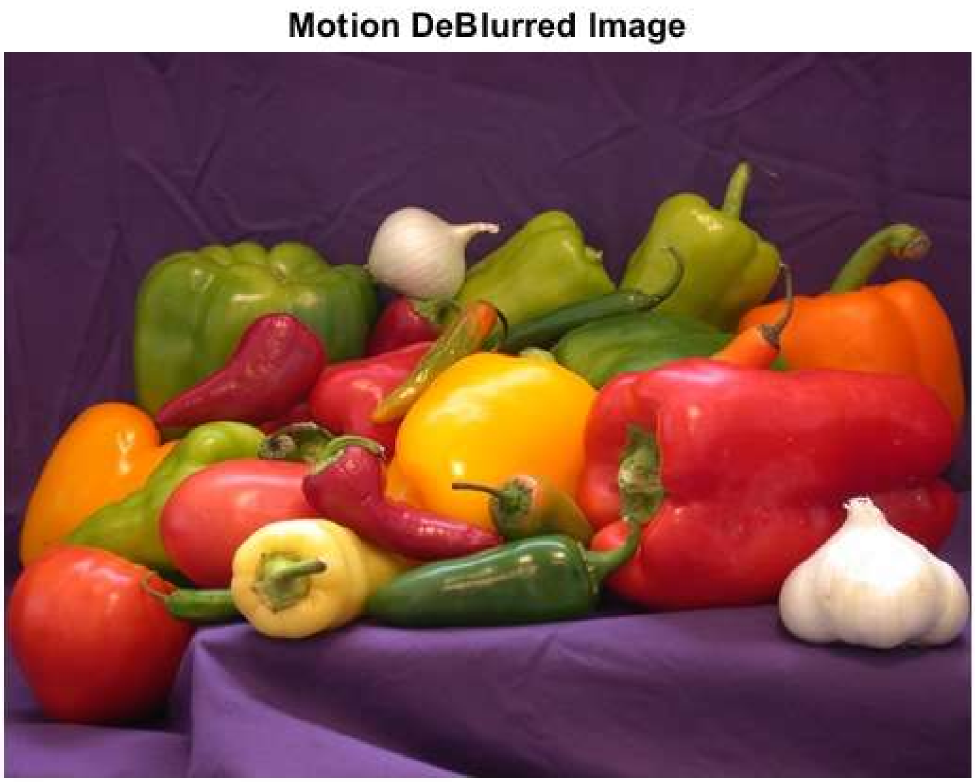


Fig (6): Deblurred Image.

- **We transform the Kernel and blurred image to frequency domain, and then we added a small constant to avoid dividing by zero, then divided them and transformed them back to time domain**

Task 2(Communication system simulation)

Brief of the work

- **First we recorded inside MATLAB two records our first goal was to filter both of them so we designed low pass filter for each one of them we defined the cut-off frequency by plotting each input and we chose to cut or filter the signal in a point that won't significantly affect the audio signals and also to get rid of background noise as much as we can then we plotted the filtered audios and listened to both input and filtered signal to ensure that a difference between them took place and didn't affect the input signals itself.**

- **Our second goal was to design a simulation for a simple communication system**

So first we got the modulated signals by shifting the signals by fc where fc is the carrier frequency and this shifting in frequency domain corresponds multiplying the signal by cos (2*pi*fc*t) in time domain.

Note: to modulate the signals correctly fc must be greater than fm (cut-off frequency or the maximum frequency of each signal) and also less than fs-fm where fs is the sampling frequency.

Then we added both the modulated signals and this was considered the transmitted (combined) signal.

Then to create the receiver we demodulated the transmitted signal into the original filtered signals and that by Multiplying the transmitted signal by cosine with fc1 specific for first audio then we passed this signal by the low pass filter designed for first signal to restore the first filtered signal and we did the same steps to restore the second filtered signal and with that we achieved our goal that was simulating a simple communication system

Answers of the required questions

1) Justify your choice of the values for the sampling frequency and bit depth.

As fs represents the numbers of samples in 1 sec this value considered one of the commonsampling frequencies used for ordinary audios.

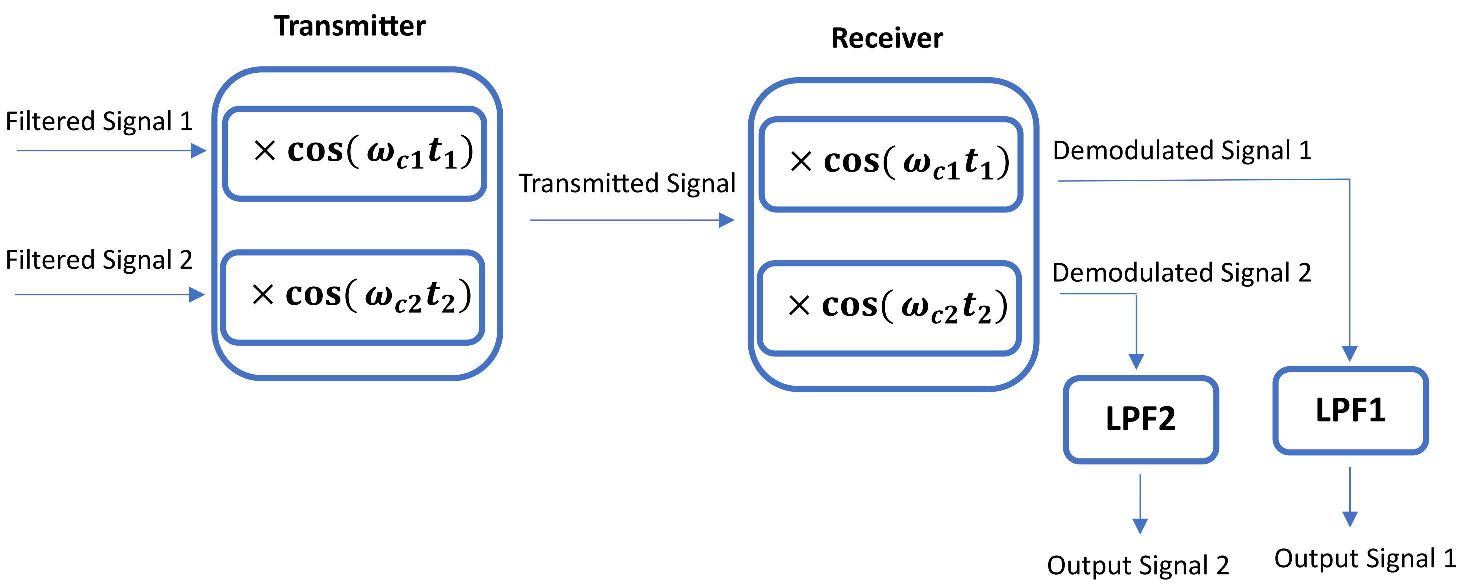
And the bit number represents each sample in the digital audio signal we can use either (8,16,24,32) bits but we considered the 16 bits to make balance between quality of signal and the size of the file.

2) Justify your choice of carrier frequencies.

The Values we used in the program was chosen according that fc should be greater than fm and less than fs-fm.

3) Draw a block diagram of both transmitter and receiver. Explain the operation of the receiver with equations both in time domain and in frequency domain.

a)



b) <u>Receiver Operation</u>

<u>Assume:</u> Transmitted signal = y(t), Demodulated signal = z(t), Output signal = x(t).

ω_{c1} (carrier frequency for signal 1), ω_{c2}(carrier frequency for signal 1)

ω_{p1} (max or cut-off frequency for LPF1), ω_{p2}(max or cut-off frequency for LPF2).

1) Time Domain

$$z_1(t) = y(t) \times \cos(\omega_{c1} t_1)\,, \quad z_1(t) \times \mathrm{sinc}(\omega_{p1} t_1) = x_1(t)$$

$$z_2(t) = y(t) \times \cos(\omega_{c2} t_2) \quad, z_2(t) \times \mathrm{sinc}(\omega_{p2} t_2) = x_2(t)$$

2) Frequency Domain

$$z(t) = y(t) \times \cos(\omega_c t) \quad \longleftrightarrow \quad Z(\omega) = \frac{1}{2}[Y(\omega - \omega_c) + Y(\omega + \omega_c)]$$

$$X(\omega) = Z(\omega) \times rect(\omega_p)$$

Outputs And Simulations

b)

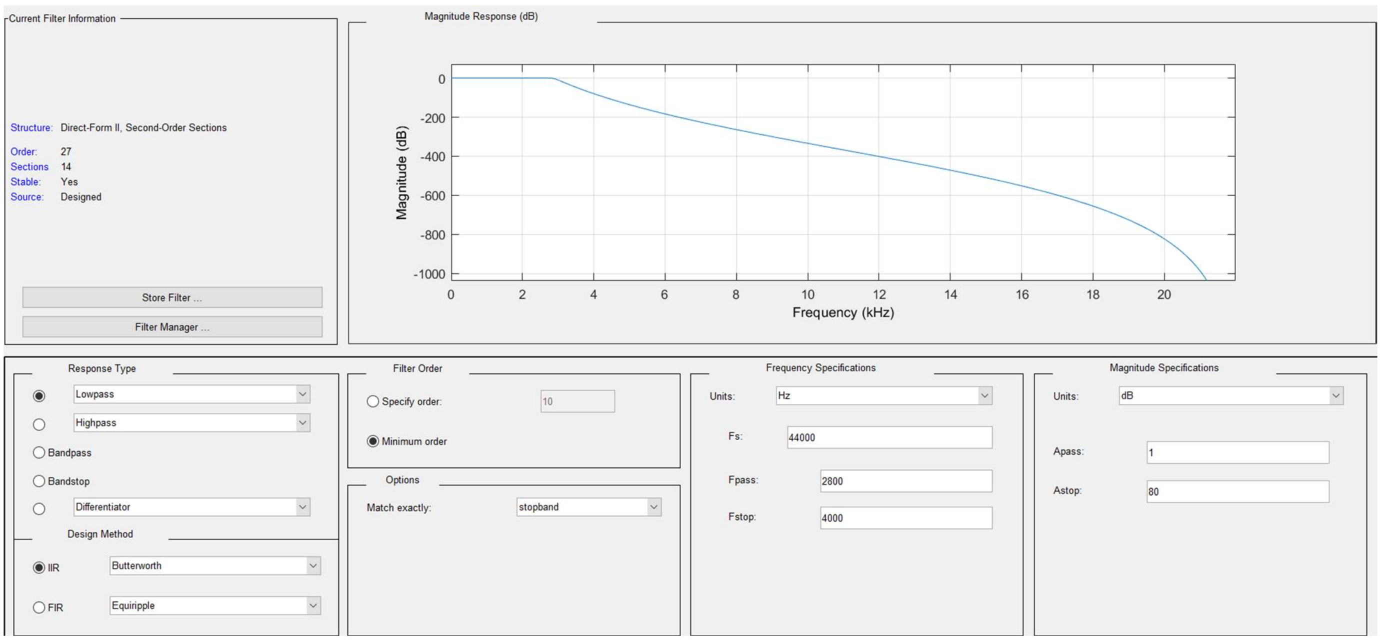


Fig (7): Frequency Response of LPF for input 1

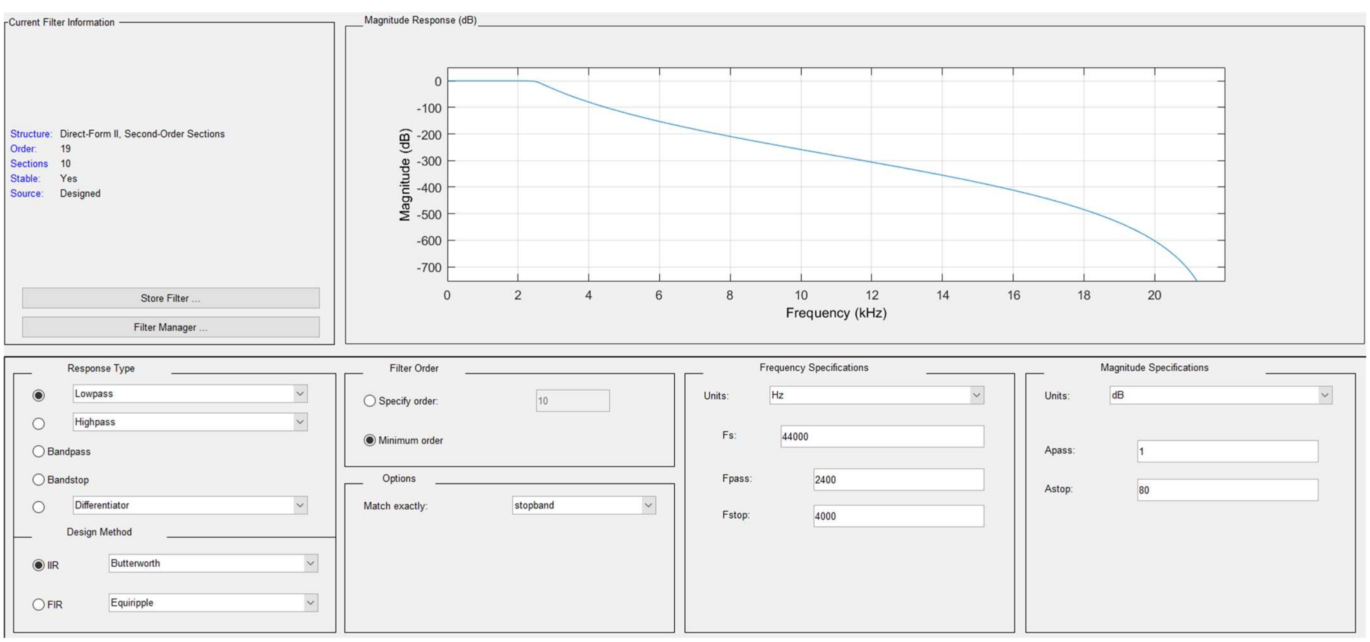


Fig (8): Frequency Response of LPF for input 2

c)

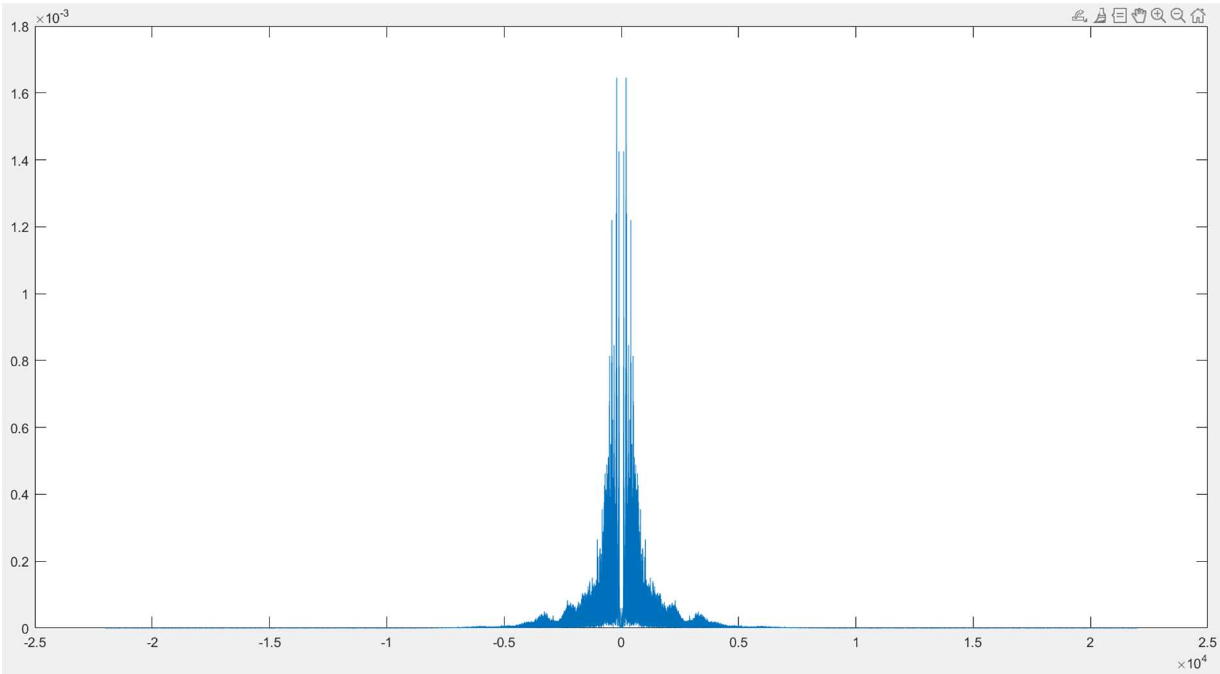


Fig (9): Input 1 Before Filtering

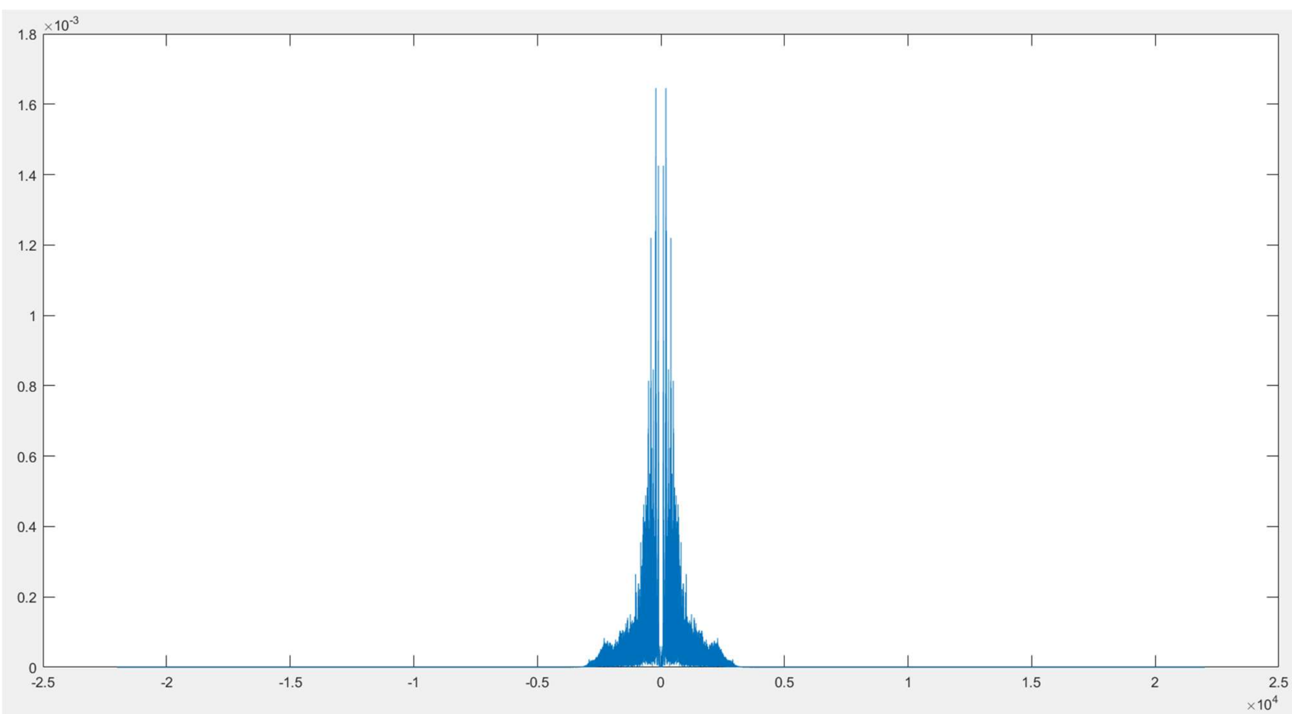


Fig (10): Input 1 After Filtering

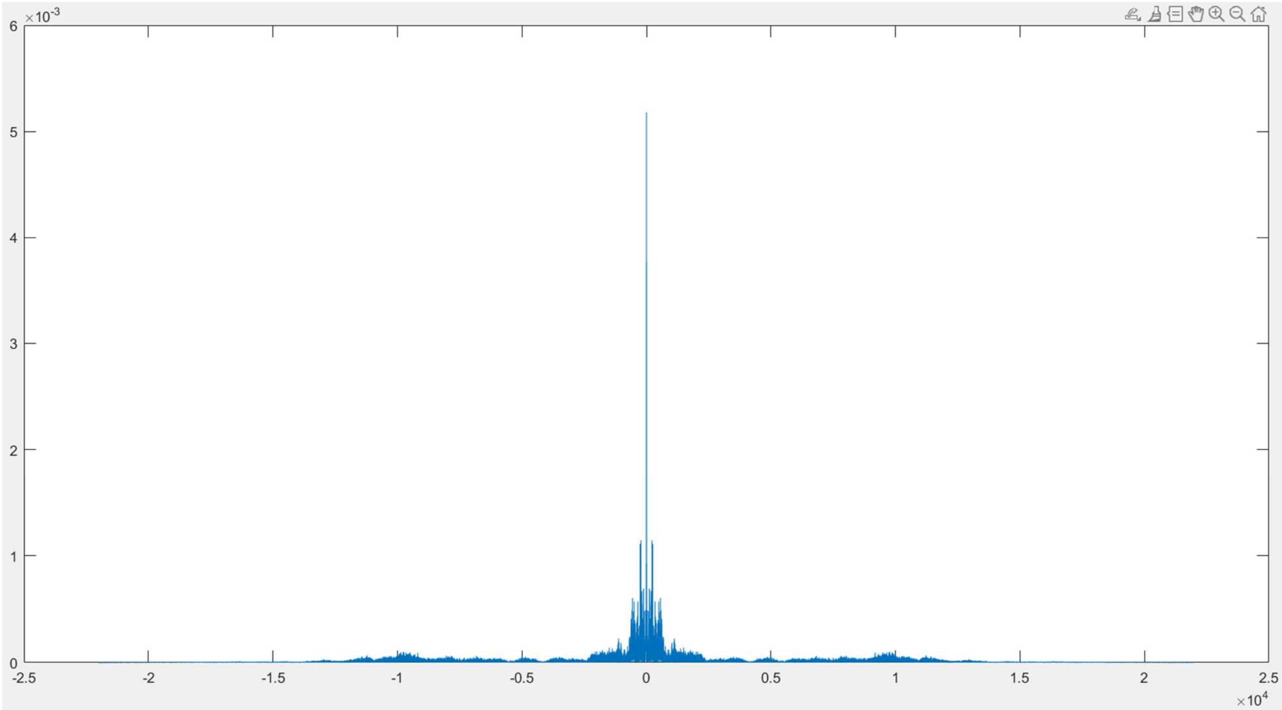


Fig (11): Input 2 Before Filtering

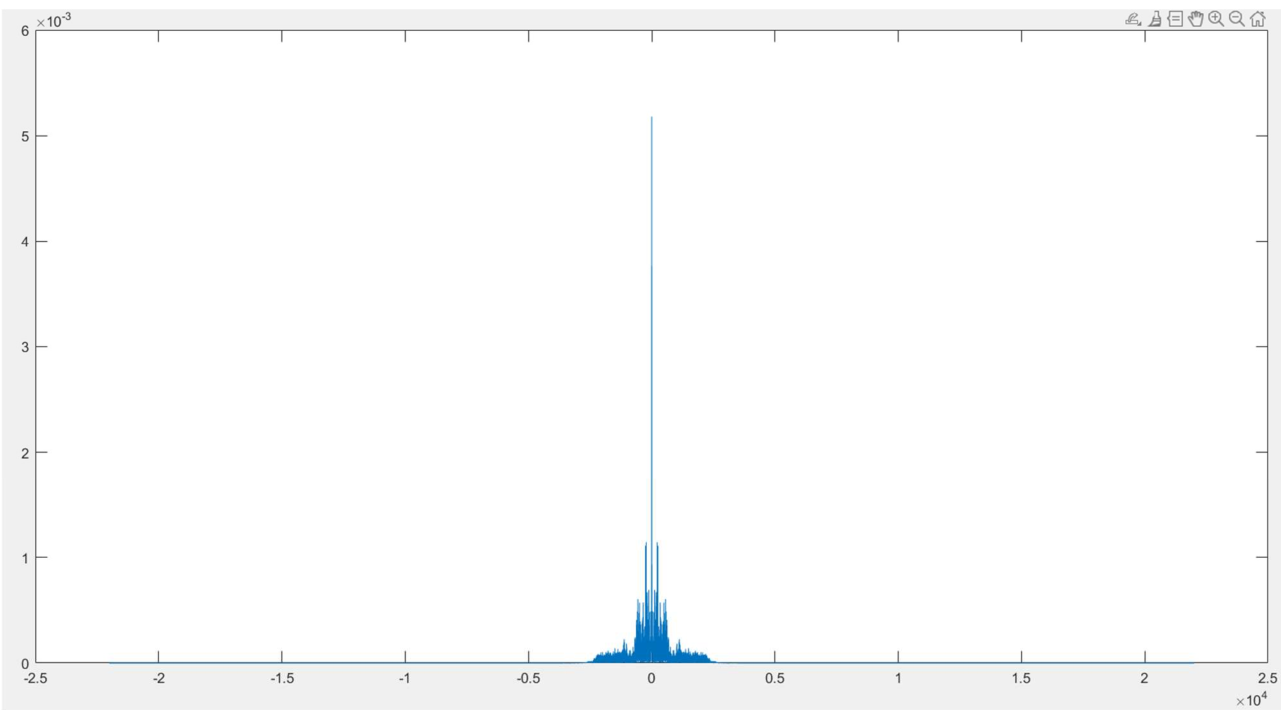


Fig (12): Input 2 After Filtering

d)

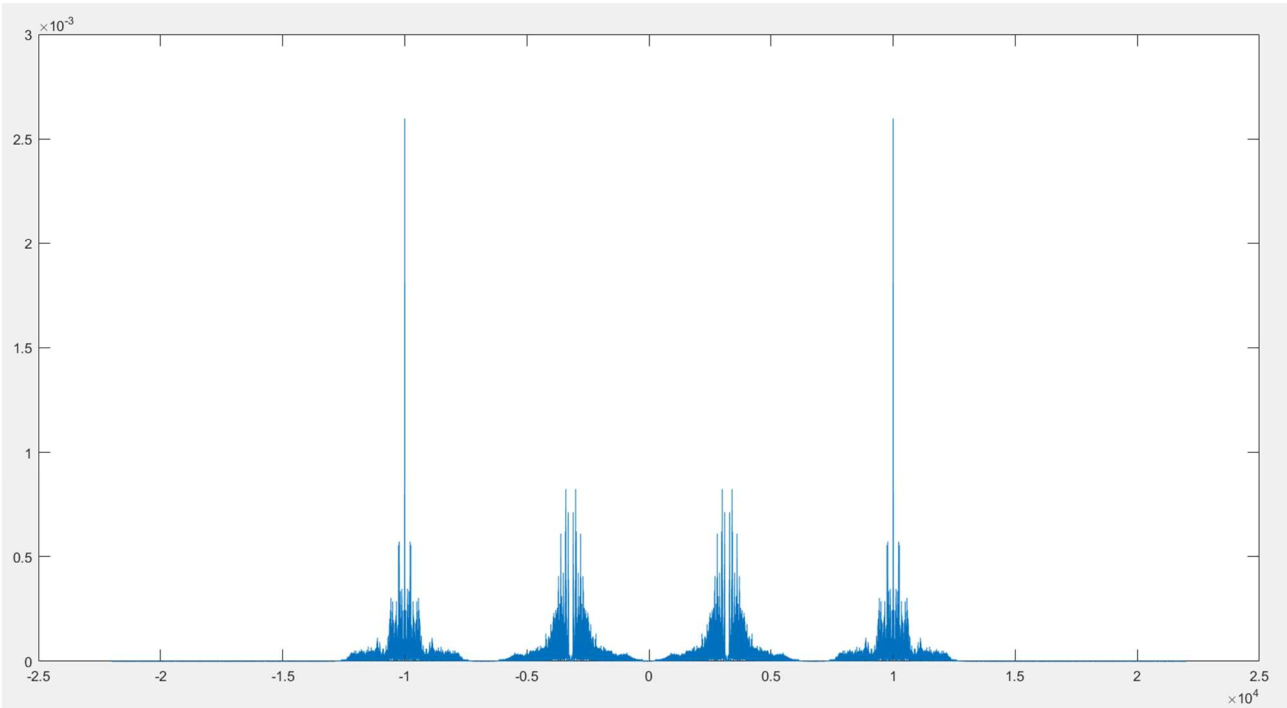


Fig (13): Transmitted Signal

e)

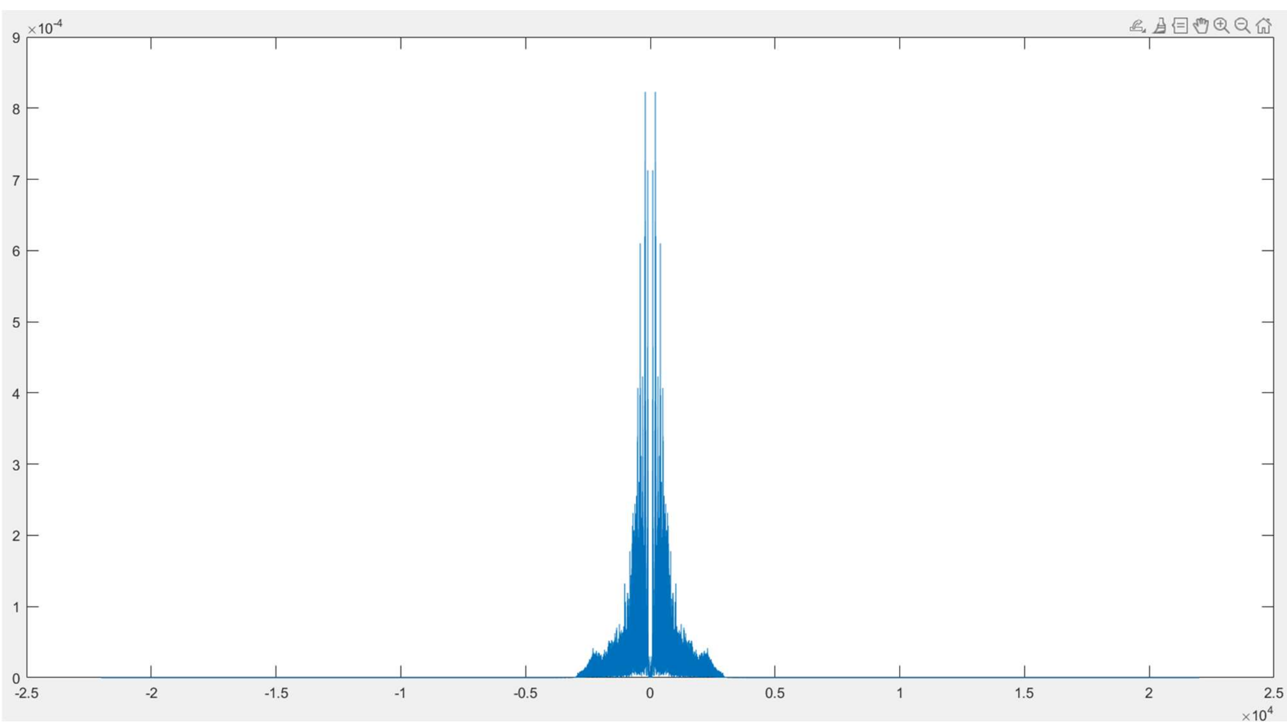


Fig (14): Demodulated Signal 1 "Output1"

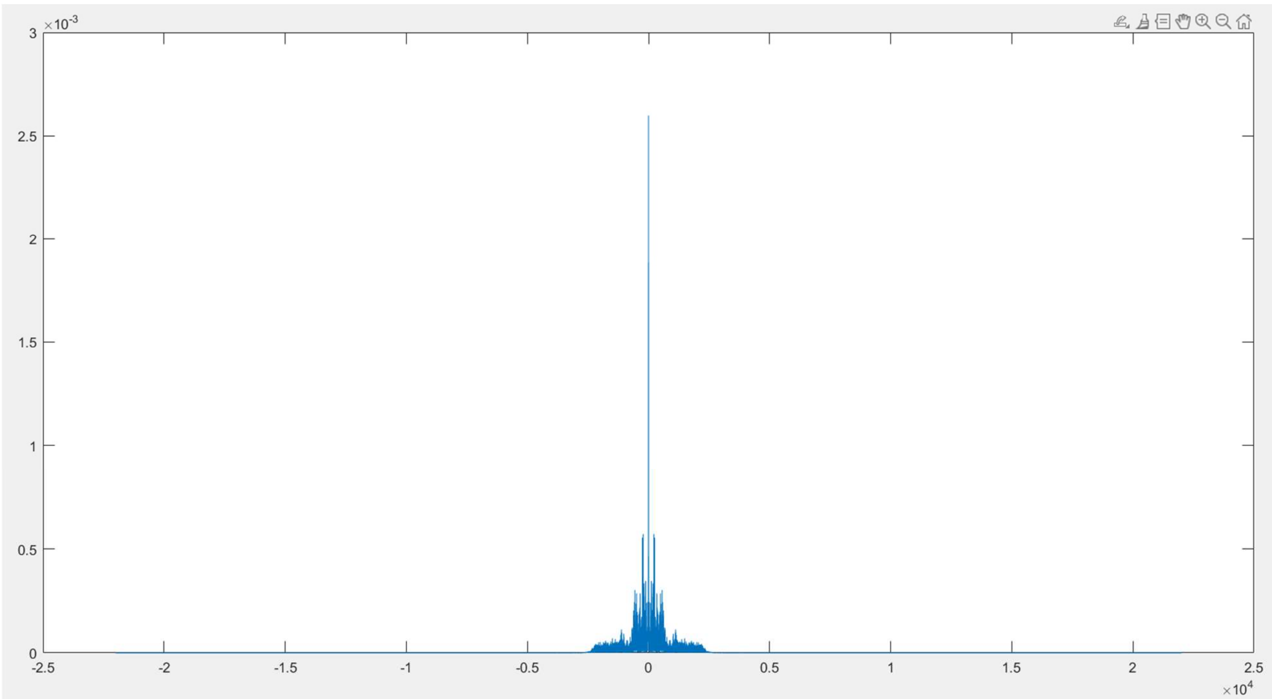


Fig (15): Demodulated Signal 2
“Output2”

Appendix

Code (Task 1)

```
%%% Submitted By : Amr Wael Leithy , Kareem Atef %%%%


%%% RGB components %%%
img = imread('peppers.png');
red = img(:,:,1);
green = img(:,:,2);
blue = img(:,:,3);
a = zeros(size(img, 1), size(img, 2));
red_component = cat(3, red, a, a);
green_component = cat(3, a, green, a);
blue_component = cat(3, a, a, blue);
figure, imshow(img), title('Original image')
subplot(2,1,2), imshow(red_component), title('Red Component')
subplot(2,2,1),imshow(green_component), title('Green Component')
subplot(2,2,2), imshow(blue_component), title('Blue Component')

%%%% Edge filter %%%%
img_gray = rgb2gray(img);
E = [-1 0 1; -2 0 2; -1 0 1];
vertical = conv2(img_gray, E.', 'same');
horizontal = conv2(img_gray, E, 'same');
edged_img = sqrt(vertical.^2 + horizontal.^2);
figure, imshow(uint8(edged_img)), title('Edge Filter')

%%% Image Sharpening %%%
S = [ 0-1 0, -1 5 -1 , 0 -1 0];
red_enh = conv2(red,S,'same');
green_enh = conv2(green,S,'same');
blue_enh = conv2(blue,S,'same');
img_enhanced= cat(3,red_enh,green_enh,blue_enh);
figure, imshow(uint8(img_enhanced)), title('Enhanced  Image')

%%% Image Blurring %%%
B = (1/9)*[ 1 1 1, 1 1 1, 1 1 1];
red_blur = conv2(red,B,'same');
green_blur = conv2(green,B,'same');
blue_blur = conv2(blue,B,'same');
img_blurred= cat(3,red_blur,green_blur,blue_blur);
figure, imshow(uint8(img_blurred)), title('Blurred Image')

%%% Image Motion Blurring %%%
MB = 1/16 * [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ];
%%% i increased the ones so that the blurring shows %%
red_Mblur = conv2(red,MB);
green_Mblur = conv2(green,MB);
```

```
blue_Mblur = conv2(blue,MB);
img_Mblurred= cat(3,red_Mblur,green_Mblur,blue_Mblur);
figure, imshow(uint8(img_Mblurred)), title('Motion Blurred Image')

%%% Image deblurring %%%
img_fft = fft2(img_Mblurred);
MB_fft = fft2(MB,384,527);
constant = 0.0001;
MB_fft = MB_fft + constant; %%% add to avoid division by 0
img_fft = img_fft ./ MB_fft;
img_Dblurred = ifft2(img_fft);
img_Dblurred = img_Dblurred(1:384,1:512,:);
figure, imshow(uint8(img_Dblurred)), title('Motion DeBlurred Image')
```

Code (Task 2)

```
%%% Submitted By : Amr Wael Leithy , Kareem Atef %%%%


%% write audio 1 %%

%% as fs represents the numbers of samples in 1 sec
%%this value considered one of the common sampling frequencies used for ordinary
audios

fs1 = 44000;

%% as the bit number represents each sample in the digital audio signal
%% we can use either (8,16,24,32) bits but we considered the 16 bits to make balance
between quality of signal and the size of the file

bit_number = 16;

Ch = 1;
period = 10;
audio = audiorecorder(fs1,bit_number,Ch);
disp('Recording');
recordblocking(audio,period);
disp('Recorded');
input1=getaudiodata(audio);
audiowrite('input1.wav',input1,fs1);


%% write audio 2 %%
fs2 = 44000;
audio = audiorecorder(fs2,bit_number,Ch);
disp('Recording');
recordblocking(audio,period);
disp('Recorded');
input2=getaudiodata(audio);
audiowrite('input2.wav',input2,fs2);
```

```
%%Ploting input 1,2 before filtering%%

[input1,fs1] = audioread('input1.wav');
N=length(input1);
f=(-N/2:N/2-1)*fs1/N;
X1=fft(input1,N);
plot(f,abs(fftshift(X1)/N));
figure;
[input2,fs2] = audioread('input2.wav');
N=length(input2);
f=(-N/2:N/2-1)*fs2/N;
X2=fft(input2,N);
plot(f,abs(fftshift(X2)/N));

%% filtering input 1,2 hear the difference and plot them %%

filtered_signal1 = filter(LPF1,input1);
filtered_signal2 = filter(LPF2,input2);

%%hear the diffrences
soundsc(input1, fs1);
pause(10);
soundsc(filtered_signal1, fs1);
pause(10);
soundsc(input2, fs2);
pause(10);
soundsc(filtered_signal2, fs2);
pause(10);

%%plot them
N=length(filtered_signal1);
f=(-N/2:N/2-1)*fs2/N;
X1=fft(filtered_signal1,N);
plot(f,abs(fftshift(X1)/N));
figure;
N=length(filtered_signal2);
f=(-N/2:N/2-1)*fs2/N;
X2=fft(filtered_signal2,N);
plot(f,abs(fftshift(X2)/N));
audiowrite('filtered_input1.wav',filtered_signal1,fs1);
audiowrite('filtered_input2.wav',filtered_signal2,fs2);


%% Transmitted Signal %%

[filtered_input1,fs1] = audioread('filtered_input1.wav');
[filtered_input2,fs2] = audioread('filtered_input2.wav');

minLength = min(length(filtered_input1), length(filtered_input2));

t1 = 0:1/fs1:(minLength-1)/fs1;
t2 = 0:1/fs2:(minLength-1)/fs2;
```

```
fm1=2800;
fm2=2400;

%% fc should be greater than fm and less than fs-fm
fc1=3200;

%% we chose here slightly bigger fc to not let the two signals interfere after
modulation
fc2=10000;

modulatedSignal1 = filtered_input1 .* cos(2*pi*fc1*t1).';
modulatedSignal2 = filtered_input2 .* cos(2*pi*fc2*t2).';

combinedSignal = modulatedSignal1 + modulatedSignal2;
N=length(combinedSignal);
f=(-N/2:N/2-1)*fs2/N;
X1=fft(combinedSignal,N);
plot(f,abs(fftshift(X1)/N));
figure;

%% Demodulated Signals%%
demodulatedSignal1 = combinedSignal .* cos(2*pi*fc1*t1).';
demodulatedSignal2 = combinedSignal .* cos(2*pi*fc2*t2).';

output1=filter(LPF1,demodulatedSignal1);
output2=filter(LPF2,demodulatedSignal2);
N=length(output1);
f=(-N/2:N/2-1)*fs2/N;
X1=fft(output1,N);
plot(f,abs(fftshift(X1)/N));
figure;
N=length(output2);
f=(-N/2:N/2-1)*fs2/N;
X1=fft(output2,N);
plot(f,abs(fftshift(X1)/N));
audiowrite('output1.wav',output1,fs1);
audiowrite('output2.wav',output2,fs2);
```

References

1) https://www.youtube.com/watch?v=C05tMF2kvpM&t=236s
3) https://www.mathworks.com/matlabcentral/answers/index
2) https://blog.demofox.org/2022/02/26/image-sharpening-convolution-kernels/
4) Lecture 10.