

# **Image Compression Using FFT**

## **Objective:**

To understand the concept of image compression, to explore the Fast Fourier Transform (FFT) and its application in image compression and comparing the results of the proposed compression algorithm.

## **Abstract:**

Image compression plays a crucial role in various domains where efficient storage, transmission, and utilization of resources are paramount. This project explores the utilization of the Fast Fourier Transform (FFT) as a means of image compression. The objective is to develop an image compression algorithm using the FFT in MATLAB and evaluate its performance in terms of compression ratio. Through the implementation of the compression algorithm, a comparison is made. The results of this project contribute to a deeper understanding of the potential of the FFT for image compression.

## **Introduction:**

Image compression is the application of data compression on digital images. In effect, the objective is to reduce redundancy of the image data to be able to store or transmit data in an efficient form. The best image quality at a given bitrate (or compression rate) is the main goal of Image compression. Image compression can be lossy or lossless. Lossless compression is sometimes preferred for artificial images such as technical drawings, icons, or comics. This is because lossy compression methods, especially when used at low bit rates, introduce compression artifacts. Lossless compression methods may also be preferred for high value content, such as medical imagery or image scans made for archival purposes. Lossy methods are especially suitable for natural images such as photos in applications where minor

(sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate.

FFT2 is the two-dimensional Fourier transform of a matrix using a fast Fourier transform algorithm. If  $X$  is a two-dimensional array, then `fft2` takes the 2-D transform of each dimension higher than 2. The output  $Y$  is the same size as  $X$ . First it computes the fft of the Rows and then it will compute the fft for the columns.

In this Image Compression Algorithm first we are going to find the fft of the image and then we are eliminating the higher frequencies in the image and then we are finding the inverse fast fourier transform to reconstruct the Image.

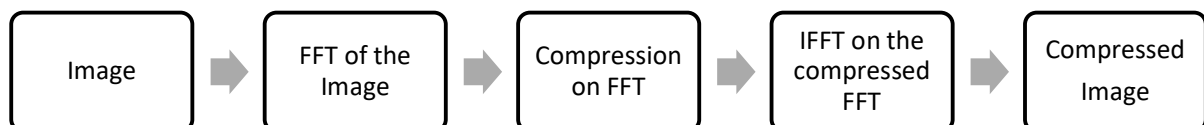
For eliminating the higher frequencies, we can have two methods.

1. Sorting the Fourier Coefficients and Eliminating the higher Values by using command `sort` in matlab.
2. Filtering the Fourier Transform using a Low pass filter.

#### Commands Used in MATLAB:

- `rgb2gray`: It converts the Coloured RGB image to a gray scale Image. It is the First Stage of Compression.
- `Fft2`: To find the fft of the Image.
- `Sort`: To sort the values.
- `Butter`: To design the filter.
- `Filter2`: To filter a two-dimensional Signal.
- `Ifft2`: To find the Inverse Fourier transform.

#### Algorithm:



## Code:

### 1.By using Sorting the Fourier Series Method:

```
A=imread("image.jpg");
B=rgb2gray(A);
Bt=fft2(B);
flow=log(abs(fftshift(Bt))+1);
Btsort = sort(abs(Bt(:)));
keep=0.90;
thresh = Btsort(floor((1-keep)*length(Btsort)));
ind = abs(Bt)>thresh;
Atlow = Bt.*ind;
Alow=uint8(iff2(Atlow));
keep=0.50;
thresh = Btsort(floor((1-keep)*length(Btsort)));
ind = abs(Bt)>thresh;
Atlow = Bt.*ind;
Alow2=uint8(iff2(Atlow));
keep=0.01;
thresh = Btsort(floor((1-keep)*length(Btsort)));
ind = abs(Bt)>thresh;
Atlow = Bt.*ind;
Alow3=uint8(iff2(Atlow));
keep=0.002;
thresh = Btsort(floor((1-keep)*length(Btsort)));
ind = abs(Bt)>thresh;
Atlow = Bt.*ind;
Alow4=uint8(iff2(Atlow));
figure(1);imshow(A);title("Original Image");
figure(2);imshow(B);title("Gray Scale Image");
figure(3);imshow(flow,[]);title("FFT of the Image");
figure(4);imshow(Alow) imwrite(Alow,"imc1.jpg");
title("Image Reconstructed with 90% of FFT Values");
figure(5);imshow(Alow2) imwrite(Alow2,"imc2.jpg");
title("Image Reconstructed with 50% of FFT Values");
figure(6);
imshow(Alow3)
imwrite(Alow3,"imc3.jpg");
title("Image Reconstructed with 1% of FFT Values");
figure(7);imshow(Alow4)
imwrite(Alow3,"imc4.jpg");
title("Image Reconstructed with 0.2% of FFT Values");
```

## **2.By Filtering the FFT using a Lowpass Filter:**

```
A=imread("image.jpg");
B=rgb2gray(A);
Bt=fft2(B);
flow=log(abs(fftshift(Bt))+1);
[b1,a1]=butter(5,0.9);
y=(filter2(b1,a1,flow))+1;
thresh = y(floor((1-0.90)*length(y)));
ind = abs(Bt)>thresh; %Find small indices
Atlow = Bt.*ind; %Threshold small indices
Alow=uint8(iff2(Atlow)); % Compressed image
[b2,a2]=butter(15,0.9);
y1=(filter2(b2,a2,flow))+1;
thresh = y1(floor((1-0.90)*length(y1)));
ind = abs(Bt)>thresh; %Find small indices
Atlow = Bt.*ind; %Threshold small indices
Alow2=uint8(iff2(Atlow)); % Compressed image
[b3,a3]=butter(25,0.9);
y2=(filter2(b3,a3,flow))+1;
thresh = y2(floor((1-0.90)*length(y2)));
ind = abs(Bt)>thresh; %Find small indices
Atlow = Bt.*ind; %Threshold small indices
Alow3=uint8(iff2(Atlow)); % Compressed image
figure(1);imshow(A);title("Original Image");
figure(2);imshow(B);title("Gray Scale Image");
figure(3);
imshow(mat2gray(flow),[]);
title("FFT of the Image");
figure(4);
freqz(b1,a1);
title("Low Pass Filter with Order 5");
figure(5);
imshow(mat2gray(y),[]);
title("FFT after Filtering with order 5");
figure(6);
imshow(Alow) % Plot Reconstruction
title("Image After Filtering with a Filter of Order 5");
imwrite(Alow,"ic1.jpg");
figure(7);
freqz(b2,a2);
title("Low Pass Filter with Order 15");
figure(8);
```

```

imshow(mat2gray(y1),[]);
title("FFT after Filtering with a Filter of Order 15");
figure(9);
imshow(Alow2) % Plot Reconstruction
title("Image After Filtering with Filter of Order 15");
imwrite(Alow2,"ic2.jpg");
figure(10);
freqz(b3,a3);
title("Low Pass Filter with Order 25");
figure(11);
imshow(mat2gray(y2),[]);
title("FFT after Filtering with a filter of Order 25");
figure(12);imshow(Alow3) % Plot Reconstruction
title("Image After Filtering with Filter of Order 25");
imwrite(Alow3,"ic3.jpg");

```

### **Output:**

#### **1.By using Sorting the Fourier Series Method:**

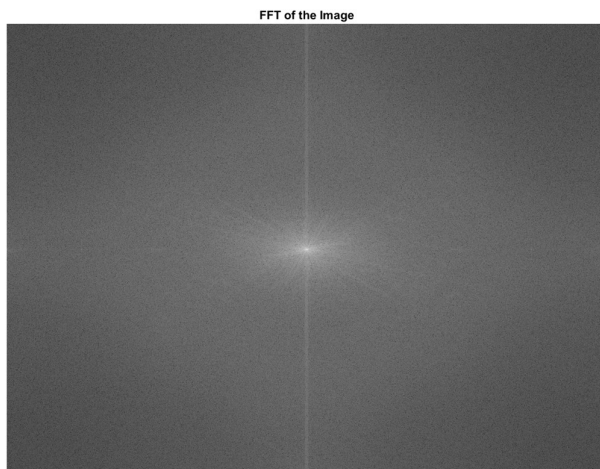


Image Reconstructed with 90% of FFT Values



Image Reconstructed with 50% of FFT Values








Image Reconstructed with 1% of FFT Values



Image Reconstructed with 0.2% of FFT Values



### Sizes of the Images:

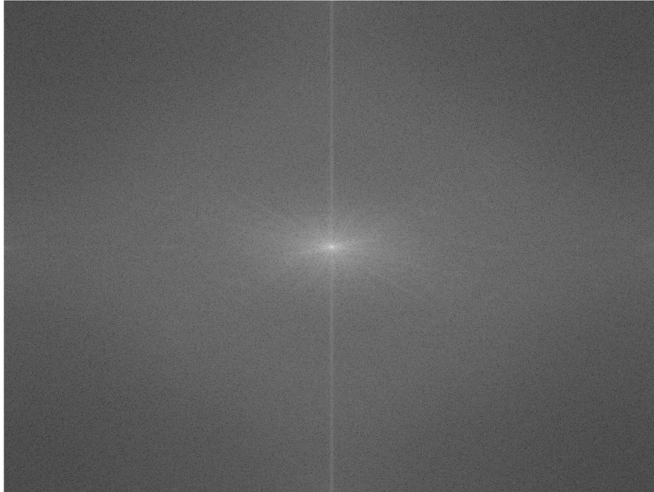
 image.jpg	14-05-2023 09:33	JPG File	1,275 KB
 imc1.jpg	02-06-2023 21:45	JPG File	905 KB
 imc2.jpg	02-06-2023 21:45	JPG File	852 KB
 imc3.jpg	02-06-2023 21:45	JPG File	356 KB
 imc4.jpg	02-06-2023 21:45	JPG File	298 KB

### **2.By Filtering the FFT using a Lowpass Filter:**

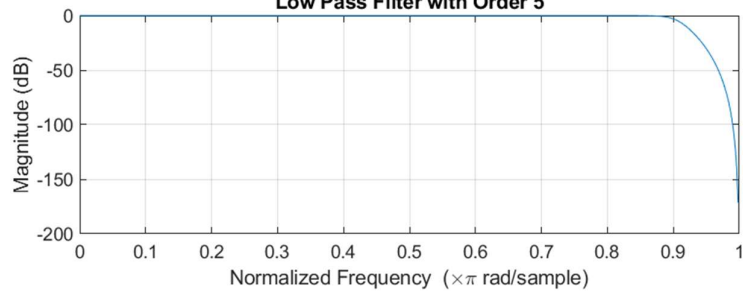
Gray Scale Image



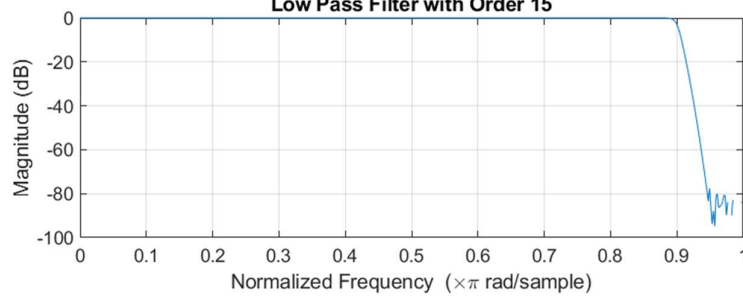
FFT of the Image



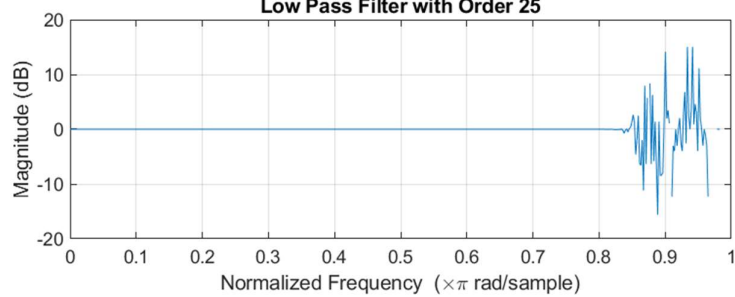
Low Pass Filter with Order 5



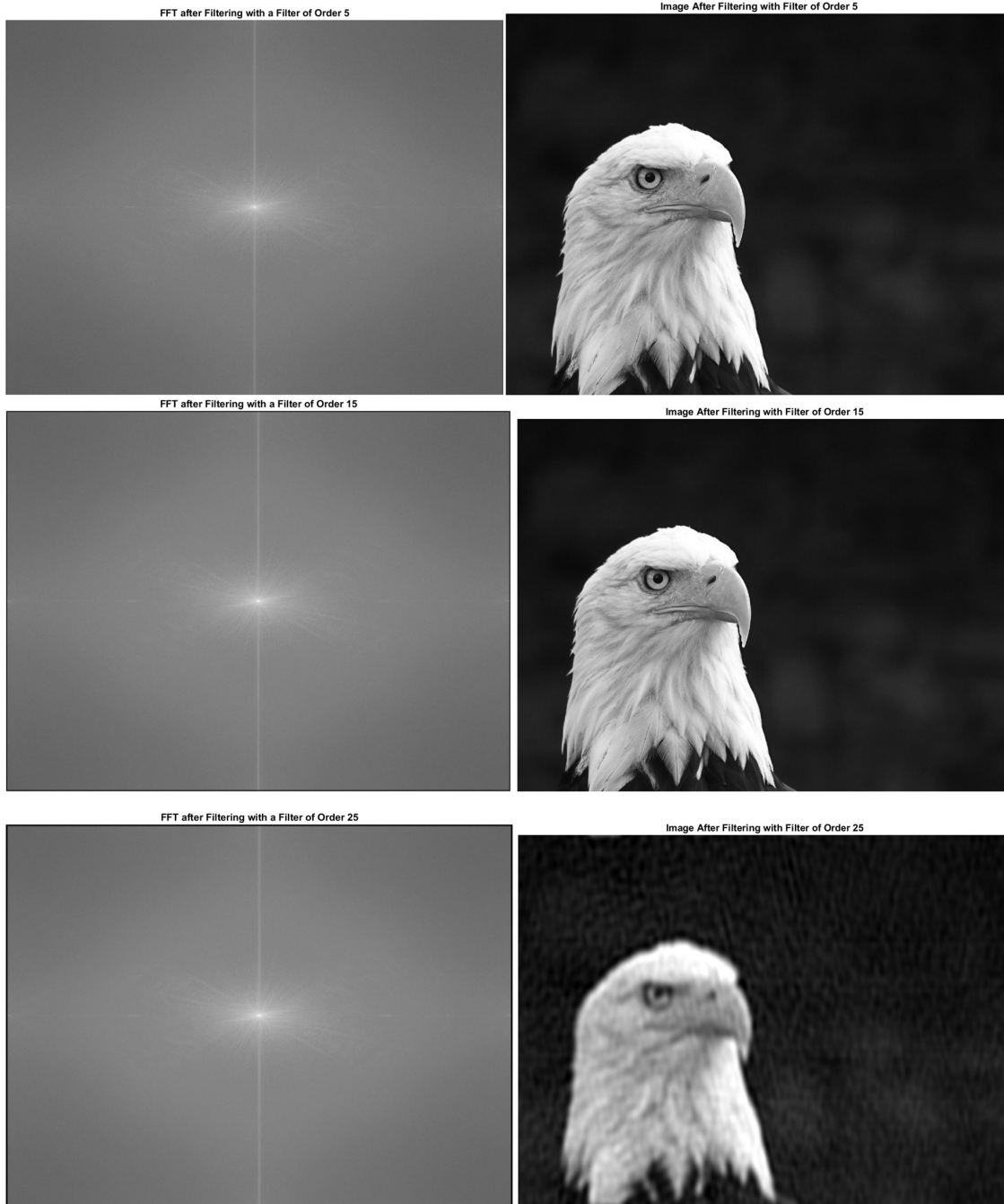
Low Pass Filter with Order 15







Low Pass Filter with Order 25







### Sizes of the Images:

 ic1.jpg	02-06-2023 21:17	JPG File	905 KB
 ic2.jpg	02-06-2023 21:49	JPG File	635 KB
 ic3.jpg	02-06-2023 21:17	JPG File	253 KB
 image.jpg	14-05-2023 09:33	JPG File	1,275 KB

### **Applications of Image Compression:**

The applications in which image compression techniques are used:

- 1) E-Health Systems
- 2) Telemedicine
- 3) Video Conferencing

### **Advantages:**

1. Reduced Storage Space
2. Faster Transmission
3. Bandwidth Conservation
4. Cost Savings

### **Disadvantages:**

1. Loss of Image Quality
2. Loss of data
3. Not suitable for sharp edged images

### **Result:**

The code for Image Compression using FFT has been Implemented and the outputs has been shown for different compression rates for both methods.

### **Conclusion:**

The results of this project demonstrate that the FFT-based compression algorithm effectively reduces storage requirements and enables faster transmission of images. The compression ratios achieved indicate the efficiency of the algorithm in compressing images. There are more efficient algorithms for Image Compression like Wavelet's Transform and Discrete-Cosine Transform.