Signals Project Report
Computer Department - FECU

Muhammad Amr Hassan 9220728
Youssef Tarek Mehany 9220990

Presented To

Dr. Micheal Melek
Eng. Sayed Kamel

May 2024

Contents

List of Figures

Chapter 1

Used Libraries For Project

1.1 Numpy

Famous Library in python that deals with mathematical operations on vectors and scalars.

1.2 Matplotlib

Famous Library in python for plotting data.

1.3 Scipy

Famous Library in python that deals with operations on signals like DCT and IDCT.

1.4 CV2

Famous Library in python that deals with images and processing it. In this project it's used only in reading the image.

1.5 OS

Famous Library in python that deals with operations in opertaing system like creating directories and files.

```
1  ################################ All used Libraries ################################
2
3  import numpy as np
4  import matplotlib.pyplot as plt
5  import cv2
6  import sys
7  import os
8  from scipy.fft import dct,idct
9  from math import log10
10
11 ####################################################################################
```

Listing 1.1: Importing the used libraries in project.

Chapter 2

First Requirement

2.1 Reading Image File

From Reading The file, we can get the image data as a 3D array.The first dimension is the row of the pixel,The second one is the column of the pixel and the third is the RGB color channel. The value is intensity of the RGB color which varies from 0 to 255.

```
def getImageComponents(inputImage):
    os.makedirs("Image Components", exist_ok=True)
    colors = ["Reds","Greens","Blues"]
    for i in range(3):
        plt.figure(figsize=(14, 6))
        plt.imshow(inputImage[:,:,2-i],cmap=colors[i])
        plt.axis("off")
        plt.colorbar()
        plt.savefig(f"Image Components/a_{colors[i][:-1].lower()}.png")
inputImageArray = cv2.imread("image1.png")
getImageComponents(inputImageArray)
```

Listing 2.1: Getting Image Data.

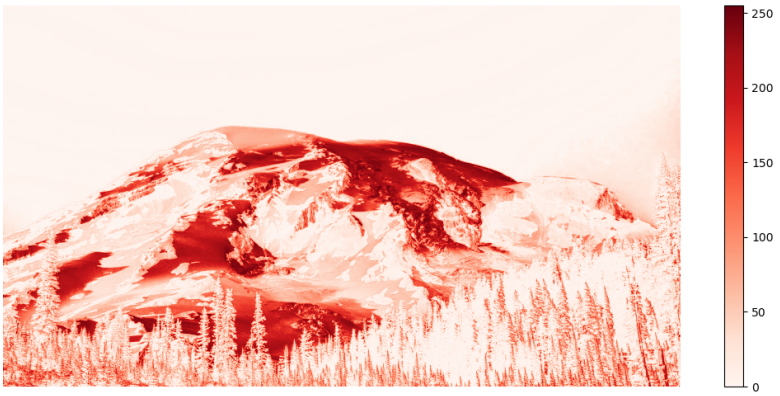

Figure 2.1: Red component of Image.

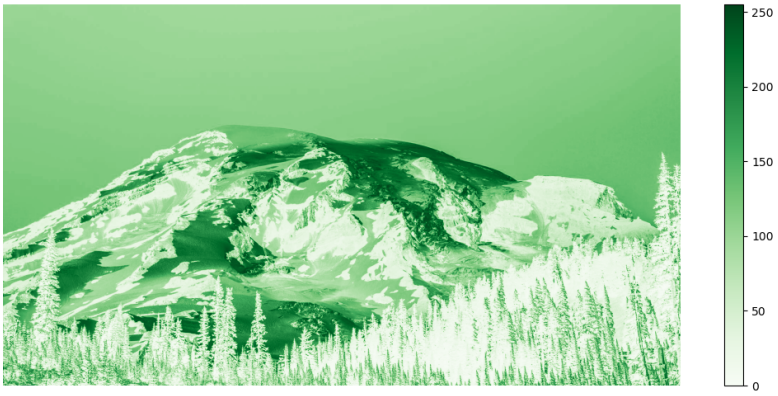

Figure 2.2: Green component of Image.

Figure 2.3: Blue component of Image.

Chapter 3

Second Requirement

3.1 Compressing Image Using Discrete Cosines Transform

As stated in the project document , We deal with each color channel of image matrix as group of blocks. Each block is 8x8 pixels.We apply to each block the DCT such that we get the lower frequencies that has the higher impact on eye and then we can neglect the other frequencies and just store the important ones.

Discrete Cosines Transform Equation

$$X\left[p,q\right]=\alpha_p\alpha_q\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}x[m,n]\cos\frac{\pi\left(2m+1\right)p}{2M}\cos\frac{\pi\left(2n+1\right)q}{2N}$$

$$\alpha_p=\begin{cases}\frac{1}{\sqrt{M}} & p=0\\ \sqrt{\frac{2}{M}} & 1\leq p\leq M-1\end{cases}\qquad \alpha_q=\begin{cases}\frac{1}{\sqrt{N}} & q=0\\ \sqrt{\frac{2}{N}} & 1\leq q\leq N-1\end{cases}$$

```
def compressImage(inputImage,m):
    blocksShape = (inputImage.shape[0]//8,inputImage.shape[1]//8)
    compressedImage = np.zeros((m*blocksShape[0],m*blocksShape[1],3))
    for channel in range(3):
        for row in range(blocksShape[0]):
            for col in range(blocksShape[1]):
                processedBlock = inputImage[8*row:8*(row+1),8*col:8*(col+1),channel]
                dctBlock = DCT(processedBlock)
                dctRetainedBlock = dctBlock[:m,:m]
                compressedImage[m*row:m*(row+1),m*col:m*(col+1),channel] =
    dctRetainedBlock
    return compressedImage
```

Listing 3.1: Code For Compressing Image using DCT.

Chapter 4

Third Requirement

4.1 Comparing the sizes of all images

Image	Size (MB)
Original	5.93
Compressed with $m = 1$	0.19
Compressed with $m = 2$	0.74
Compressed with $m = 3$	1.67
Compressed with $m = 4$	2.97

Chapter 5

Fourth Requirement

5.1 Decompressing Image using IDCT

To get the original image we decompress the compressed image using IDCT but on the retained mxm block with ignoring the other values by setting them to zeros.

Inverse Discrete Cosines Transform

$$x[m,n] = \sum_{p=0}^{M-1}\sum_{q=0}^{N-1} \alpha_p \alpha_q X[p,q] \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}$$

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}} & p = 0 \\ \sqrt{\frac{2}{M}} & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} \frac{1}{\sqrt{N}} & q = 0 \\ \sqrt{\frac{2}{N}} & 1 \le q \le N-1 \end{cases}$$

```
def decompressImage(compressedImage,m):
    blocksShape = (compressedImage.shape[0]//m,compressedImage.shape[1]//m)
    decompressedImage = np.zeros((8*blocksShape[0],8*blocksShape[1],3))
    for channel in range(3):
        for row in range(blocksShape[0]):
            for col in range(blocksShape[1]):
                processedBlock = np.zeros((8,8))
                processedBlock[:m,:m] = compressedImage[m*row:m*(row+1),m*col:m*(col
    +1),channel]
                idctBlock = IDCT(processedBlock)
                decompressedImage[8*row:8*(row+1),8*col:8*(col+1),channel] =
    idctBlock
    return decompressedImage
```

Listing 5.1: Code For Decompressing The Compressed Image.

We will show below how increasing m can effectively change the quality of the image.

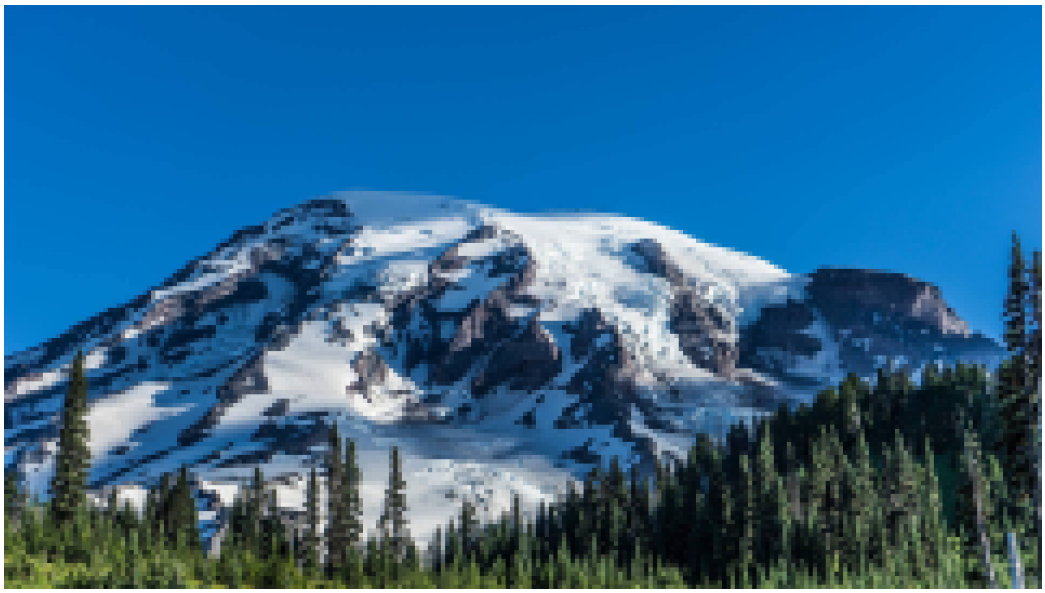

Figure 5.1: Decompressed image using $m = 1$.

Figure 5.2: Decompressed image using $m = 2$.

Figure 5.3: Decompressed image using $m = 3$.

Figure 5.4: Decompressed image using $m = 4$.

Chapter 6

Fifth Requirement

6.1 Computing PSNR

The quality of the decompressed image is measured using the Peak Signal-to-Noise Ratio (PSNR), which is defined by

$$\text{PSNR} = 10 \log_{10} \left(\frac{\text{peak}^2}{\text{MSE}} \right)$$

Where peak is the max value of pixel data type and MSE is defined by

$$\text{MSE} = \frac{1}{3MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{c=0}^{2} (x[m,n,c] - \hat{x}[m,n,c])^2$$

Where $\hat{x}$ is decompressed image matrix and x is the original image matrix.

m	PSNR (dB)
1	31.51
2	32.08
3	32.57
4	33.21

```
def calculatePSNR(inputImage,decompressedImage):
    SE = (inputImage-decompressedImage)**2
    MSE = np.mean(SE)
    peak = 255
    PSNR = 10*log10(peak**2/MSE)
    return PSNR
```

Listing 6.1: Code For Calculating PSNR.

Chapter 7

Sixth Requirement

7.1 Analysing the relation between PSNR and Quality

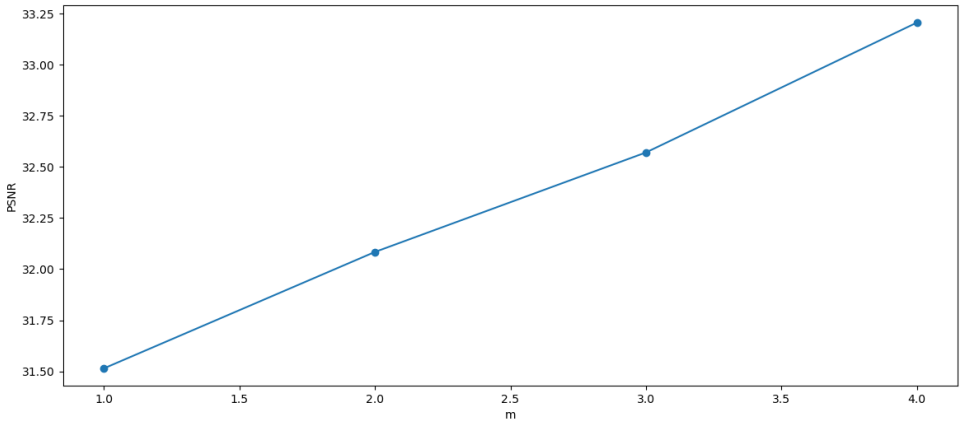

Figure 7.1: The Relation between m and PSNR.

It's obvious that by increasing m we take more effective frequencies,thus we enhance its quality more and more and increasing the PSNR also. Thus the PSNR is directly propotional with m.

Appendix A

Project's Code

```python
import numpy as np
import matplotlib.pyplot as plt
import cv2
import sys
import os
from scipy.fft import dct,idct
from math import log10

def getImageComponents(inputImage):
    os.makedirs("Image Components", exist_ok=True)
    colors = ["Reds","Greens","Blues"]
    for i in range(3):
        plt.figure(figsize=(14, 6))
        plt.imshow(inputImage[:,:,2-i],cmap=colors[i])
        plt.axis("off")
        plt.colorbar()
        plt.savefig(f"Image Components/a_{colors[i][:-1].lower()}.png")
def DCT(processedBlock):
    return dct(dct(processedBlock.T, norm='ortho').T, norm='ortho')
def IDCT(processedBlock):
    return idct(idct(processedBlock.T, norm='ortho').T, norm='ortho')
def compressImage(inputImage,m):
    blocksShape = (inputImage.shape[0]//8,inputImage.shape[1]//8)
    compressedImage = np.zeros((m*blocksShape[0],m*blocksShape[1],3))
    for channel in range(3):
        for row in range(blocksShape[0]):
            for col in range(blocksShape[1]):
                processedBlock = inputImage[8*row:8*(row+1),8*col:8*(col+1),channel]
                dctBlock = DCT(processedBlock)
                dctRetainedBlock = dctBlock[:m,:m]
                compressedImage[m*row:m*(row+1),m*col:m*(col+1),channel] =
    dctRetainedBlock
    return compressedImage
def decompressImage(compressedImage,m):
    blocksShape = (compressedImage.shape[0]//m,compressedImage.shape[1]//m)
    decompressedImage = np.zeros((8*blocksShape[0],8*blocksShape[1],3))
    for channel in range(3):
        for row in range(blocksShape[0]):
            for col in range(blocksShape[1]):
                processedBlock = np.zeros((8,8))
                processedBlock[:m,:m] = compressedImage[m*row:m*(row+1),m*col:m*(col
    +1),channel]
                idctBlock = IDCT(processedBlock)
                decompressedImage[8*row:8*(row+1),8*col:8*(col+1),channel] =
    idctBlock
    return decompressedImage
def calculatePSNR(inputImage,decompressedImage):
    SE = (inputImage-decompressedImage)**2
    MSE = np.mean(SE)
    peak = 255
    PSNR = 10*log10(peak**2/MSE)
    return PSNR
def writeImageInfo(file,sizeInBytes,inputImage,decompressedImage,m):
    file.write(f"\n                    -------------------------For m = {m
    }---------------------------------------\n\n")
    file.write(f"\t\t\t\t\t\tCompressed Image Size : {sizeInBytes/(1024)**2:.2f} MB\n
```

```
    ")
    PSNR = calculatePSNR(inputImage,decompressedImage)
    file.write(f"\t\t\t\t\t\tPSNR : {PSNR:.2f} dB\n\n")
    file.write("
    ------------------------------------------------------------\n\n")
    return PSNR
def plotPSNRS(allMs,PSNRS):
    plt.figure(figsize=(14, 6))
    plt.plot(allMs,PSNRS,marker='o')
    plt.xlabel("m")
    plt.ylabel("PSNR")
    plt.savefig("PSNR_plot.png")
def Project():
    with open('sizes.txt', 'w') as file:
        # Write to the file
        inputImageArray = cv2.imread("image1.png")
        getImageComponents(inputImageArray)
        os.makedirs("Decompressed Images", exist_ok=True)
        os.makedirs("Compressed Images", exist_ok=True)
        file.write(f"Original Image Size : {sys.getsizeof(inputImageArray)/(1024**2)
    :.2f} MB\n")
        PSNRS = np.zeros(4)
        for m in range(1,5):
            compressedImage = compressImage(inputImageArray,m)
            np.save(f"Compressed Images/compressedImage_m{m}",compressedImage.astype(
    np.uint8))
            decompressedImage = decompressImage(compressedImage,m)
            PSNR = writeImageInfo(file,sys.getsizeof(compressedImage.astype(np.
    float16)),inputImageArray,decompressedImage.astype(np.uint8),m)
            PSNRS[m-1]=PSNR
            cv2.imwrite(f"Decompressed Images/decompressedImage_m{m}.png",
    decompressedImage)
        allMs = np.linspace(1,4,4,dtype=np.uint8)
        plotPSNRS(allMs,PSNRS)
Project()
```

Listing A.1: Code For Compressing Image using DCT.

Appendix B

References

1. Matlab DCT explaination
2. Wikipedia DCT explaination