IMAGE AND VIDEO ANALYSIS

Assignment -1

U1983484

For this assignment, I have chosen to work on **Question-1** which is 'Image compression using DCT and Pyramid'. The implementation is done in **Python** using many libraries like SciPy, NumPy, matplotlib etc.

Question 1

I have implemented the DCT (Discrete Cosine Transform) function from first principles following the below-given formula:

$$T(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)g(x,y,u,v).$$

$$g_1(x,u) = \alpha(u)\cos\left[\frac{(2x+1)u\pi}{2N}\right]$$

As the image is in the spatial domain, we would like to transform the image into the frequency domain. That is when **DCT transformation** algorithm is used. It is also considered as an Image Enhancement technique.

I have started implementation with the function **'DCT'** which treats each row in an array of matrix R_{nxn} where each row r_i ($0 \leq i \leq n$) is considered as a distinct single array in one dimension. Each r_i acts as input in above-mentioned formula then in order to produce the output, rows r_1 to r_n are then concatenated.

In order to check if the input matrices of various sizes are equal to the gained output, I generated 10 matrices with random integer values (between 0 and 255), and applied **DCT** function on them, and reported my observation as follows:

```
test case: 1  ... isEqual: True
test case: 2  ... isEqual: True
test case: 3  ... isEqual: True
test case: 4  ... isEqual: True
test case: 5  ... isEqual: True
test case: 6  ... isEqual: True
test case: 7  ... isEqual: True
test case: 8  ... isEqual: True
test case: 9  ... isEqual: True
test case: 10 ... isEqual: True
```

Along with DCT, we have applied MSE (Mean Squared Error) on our output, which will calculate the difference between two matrices. Averaging the difference between two matrices element-wise. Average further is divided by the total number of elements. This average is also known as **'Residual Sum of Squares'**.

I have calculated MSE for matrices of different size. The output values are:

```
test case: 1 Mean Square Error : 2.2936352470726357e-26
test case: 2 Mean Square Error : 2.2039997869993074e-26
test case: 3 Mean Square Error : 3.8875075081739624e-26
test case: 4 Mean Square Error : 8.192517599251112e-26
test case: 5 Mean Square Error : 7.039302315290381e-26
test case: 6 Mean Square Error : 5.54910418433402e-26
test case: 7 Mean Square Error : 8.436569728865816e-26
test case: 8 Mean Square Error : 1.361067510901421e-25
test case: 9 Mean Square Error : 1.8315867943656266e-25
test case: 10 Mean Square Error : 1.3423867328576637e-25
```

The test case has been created in order to validate the accuracy of the code where the system generates a matrix of random size and the input and the output post applying and disapplying DCT are observed. The output values obtained are very much related to the input values from the original matrix.

Question 2

In this question, I have used the previously applied DCT transformation function to perform compression on an image. Then, the number of elements in the array (except zero elements) can be reported as the total number of bits. For our implementation, the total number of bits is as follows:

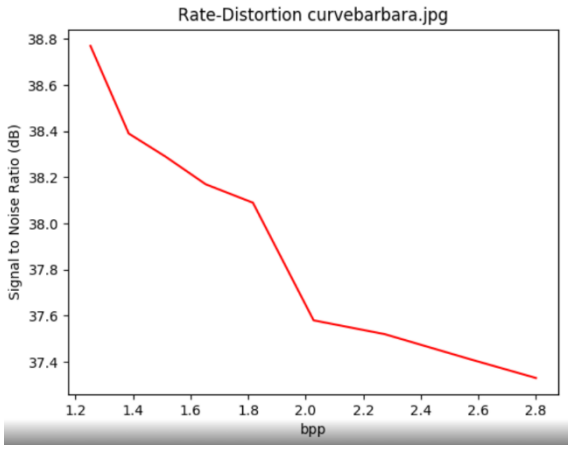
```
barbara.jpg TOTAL NUMBER OF bits: 262144
zelda.jpg TOTAL NUMBER OF bits: 262144
airplane.jpg TOTAL NUMBER OF bits: 262144
```

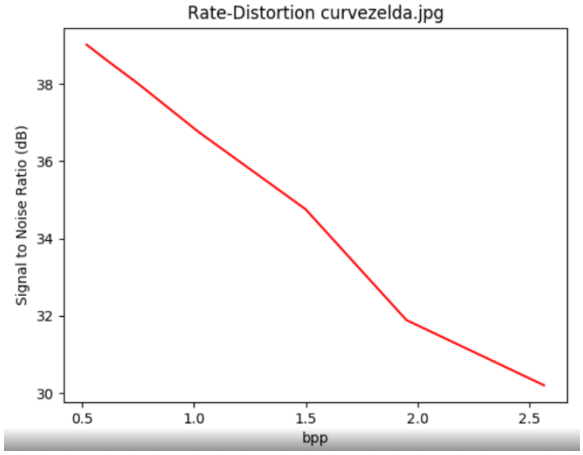
For the second part of the question, we have to calculate SNR value by dividing the variance value by previously calculated Residual sum of squares. our output obtained is:

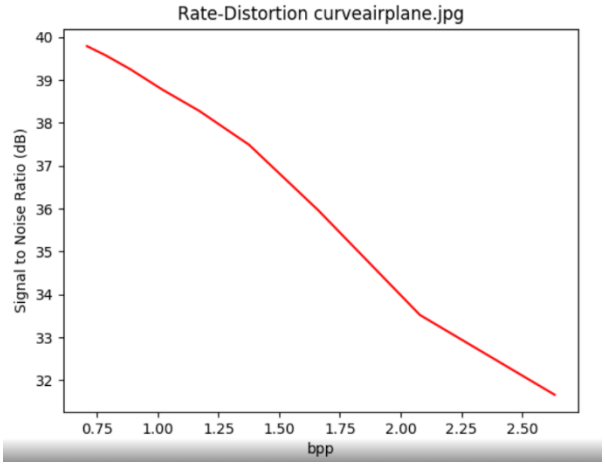
```
barbara.jpg Signal To Noise is : 27.11
zelda.jpg Signal To Noise is : 26.59
airplane.jpg Signal To Noise is : 26.51
```

Question 3

We must apply a provided 8 by 8 matric on our input image in order to quantise it. Meanwhile, we are asked to draw rate-distortion graphs in order to show the performance of the threshold. I applied quantisation function on three different images and used a linear graph to show the relation between SNR and bbps of each image i.e. Rate Distortion. It is good to mention it that I used “ImageSize / (length * height)” formula to calculate bbps. The output is below shown images:







Question 4

Implementation of Gaussian and Laplacian pyramid for images of sizes of powers of two (i.e. 2^n) has been carried out in the current question. It is implemented with the '**REDUCE'** and the '**EXPAND'** operation from first principles.

Gaussian Pyramid

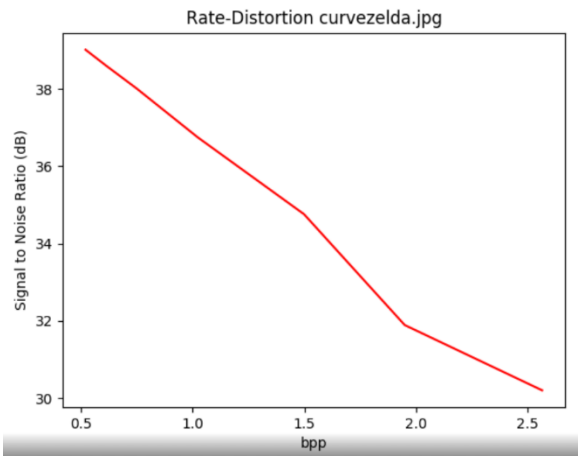
Laplacian Pyramid

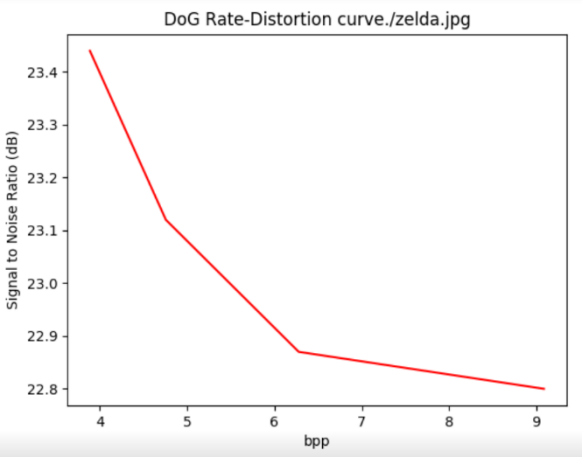
Question 5

In this question, I am using the previously implemented Laplacian pyramid to compress and quantise with thresholding of Laplacian coefficients. I am quantising the coefficients at different levels at the various number of bits as asked in the question, the output obtained is as below:

We can see the effect of quantisation in below image which can be clearly differentiated when compared with effects in Laplacian pyramid in the above question. The noise and the distortion level is increasing as we go lower.

Question 6





	Zelda	DCT Zelda	Laplacian Zelda
Size	**143.9**	**84.1**	**108**

DCT Zelda

Laplacian Zelda

A glance at the pictures provided shows that DCT Zelda has a better quality compared to Laplacian Zelda. Additionally, it is clear that the size of the image after applying DCT Zelda, is less than the image after applying Laplacian Zelda. It is because that Laplacian Zelda quantises the image in each level with the various bit (2, 3, 4 and 5). DCT Zelda, However, quantises the image just for once, and it removes frequencies which are more than a given threshold.

Project link on GitHub : https://github.com/mhn1991/VideoAndImageProject