SC - 205 - Discrete Mathematics

Maths in Multimedia Video Compression

7th July 2022

Assigned By ***Prof. Manish Gupta*** and ***Prof. Manoj Raut***
202101464 - Shubham Patel **202101477 - Vinit Mehta** **202101185 - Jay Sanghani** **202101182 - Dhyey Ladani** **202101077 - Parth Parmar**

1 Introduction

In this project we did research on Video compression algorithm using codec and there are many algorithms around but MPEG-4 format is used widely and it also supports most Video and Audio formats. The container MPEG-4 can contain video audio and sub-title files with some meta data in it.

2 Motivation for this project

The video taken by camera $RawVideo$ can be around 60GB per hour and it is very hard to manage and very tough to transfer it requires very high bandwidth to transmit and to process it also require very high processing power. to resolve this we compress the video files with the help of codecs the audio file and all other required contents to a container. MPEG-4 is the widely used container and it supports most of the audio video formats.

3 About MPEG-4

MPEG-4 stands for moving picture expert group-4. MPEG-4 is one type container. It provide good video quality at lower bit. It includes both audio and video data and supports multiple A/V codecs. Mostly it used for transmit video content over the web and for streaming videos on the internet in large scale of amount.

4 Working Process

4.1 Differntiating I-B-P Frame

In a video there is a term called frame rate which is number of frames per second example for in 60 frames per second video there are 60 different frames going in a second so there is possibility that there are two frames which are same so this makes no sense to save both frames.so we use some compression methods to overcome this.one of the most famous method is I-B-P frame differentiation and then compression of respective frame using compression algorithms.The I-frames are “key-frames”, which have no reference to other frames and their compression is not that much.A P-frame (Predicted picture) holds only the changes in the image from the previous frame.The B-frame (Bidirectional predicted picture) saves even more space by using differences between the current frame and both the preceding and following frames to specify its content. P and B frames are also called Inter frames. The order in which the I, P and B frames are arranged is called the Group of pictures.in normal scenario in the classification of I-B-P frames is IBBPBBPBBPBBIBBPBB...the compression ratio is 1:27 in this frame distribution.

5 Compression Of I Frame

5.1 Colour Space Converter

If in a certain case the colour channel is RGB (Red, Green, Blue) then after passing it through a colour space converter it gets converted to another colour channel e.g. - YCbCr for better image representation/vision and it would be easier to solve further tasks using the new colour channel. Similarly, any colour channel gets converted to another with the help of this converter.

5.2 DCT

In DCT transformation, we get the input as the matrix of NxN where N is the pixels of the image.
The output of the DCT transformation is also a matrix of NxN denoted by V which is given by

$$\mathbf{V} = \mathbf{C}\ \mathbf{U}\ \mathbf{C}^T$$

where U is the matrix of the input image whose order is given by its pixel

V is the output of the DCT

$$V(k) = \alpha(k) \sum_{N-1}^{n=0} U(n) \cos\left(\frac{\pi(2n+1)K}{2N}\right)$$

$, 0 \leq k \leq N-1$

C matrix is given by the following formula:-

$$C = \alpha(k)\cos(\frac{\pi(2n+1)k}{2N}), 0 \leq n, k \leq N-1$$

Once, we get a matrix c we have to multiply it by our image matrix which was passed on as out input to DCT, U and would get an intermediate which we have to multiply by the transpose of C to get our final DCT matrix.

Example:-

If the image matrix U which is the input to DCT is 4x4 matrix given by

$$U = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

then first matrix C would be given by:

From the above formula of C,

$$C = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.65 & 0.27 & -0.27 & -0.65 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.27 & -0.65 & 0.65 & -0.27 \end{bmatrix}$$

Thus our final DCT output in the form of a matrix V would be:

V = C U C^T, and calculating it we get

$$V = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.65 & 0.27 & -0.27 & -0.65 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.27 & -0.65 & 0.65 & -0.27 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0.65 & 0.5 & 0.27 \\ 0.5 & 0.27 & -0.5 & -0.65 \\ 0.5 & -0.27 & -0.5 & 0.65 \\ 0.5 & -0.65 & 0.5 & -0.27 \end{bmatrix}$$

$$V = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

V is the final DCT matrix which will be passed on as an input to quantizer.

5.3 Quantisation

It involves lossy compression technique achieved by compressing a range value to a single quantum value. (Simply it converts the continuous signal to discrete signals)
In this process, there is a pre-defined quantisation matrix for each color channel.
Process:-

Divides the output $(8 \times 8 matrix)$ by quantisation matrix for appropriate color channel.Then it round it up. It contains maximum number of zeros.

DCT coefficient matrix:

$$\begin{bmatrix} -415 & -33 & -58 & 35 & 58 & -51 & -15 & -12 \\ 5 & -34 & 49 & 18 & 27 & 1 & -5 & 3 \\ -46 & 14 & 80 & -35 & -50 & 19 & 7 & -18 \\ -53 & 21 & 34 & -20 & 2 & 34 & 36 & 12 \\ 9 & -2 & 9 & -5 & -32 & -15 & 45 & 37 \\ -8 & 15 & -16 & 7 & -8 & 11 & 4 & 7 \\ 19 & -28 & -2 & -26 & -2 & 7 & -44 & -21 \\ 18 & 25 & -12 & -44 & 35 & 48 & -37 & -3 \end{bmatrix}$$

A common quantization matrix:-

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Divide and rounding to integer value:-

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -3 & 4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

As example first element,

round(-415/16) = round(-25.93) = -26

Similarly we can do that for other element.

6 Zig-Zag Scan

After getting 8×8 matrix from quantization, to pass this matrix to entropy encoding stage, we've to convert it into a 1D array. will give 1D array of integers.The largest term in the section is known as DC coefficient while all other terms are known as AC coefficients.

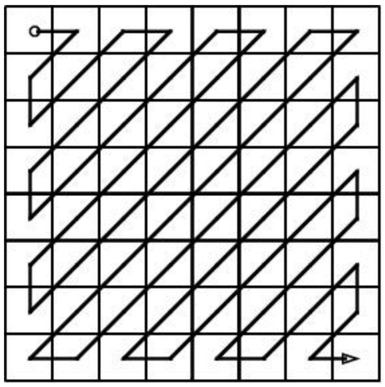


Figure 1: Zig-Zag Scan

7 Entropy Encoding and Decoding

It is a type of lossless coding to compile data type.

This technique involves replacing data elements with coded representation of data elements which is passed on as an input to entropy decoder block.
There are two types of entropy encoding:

1) Huffmann encoding
2) Arithmetic encoding

Arithmetic encoding is better than Huffmann encoding as its compression ratio is larger than Huffmann encoding.

7.1 Encoding

7.1.1 Huffman Encoding

It is a method of compression of data which compresses the data which much efficient loss of any data.

Example:-
Consider the message 'COMMITTEE' Number of symbols = 9
Unique symbols = 6

Alphabets	Frequency	Length code
C	1	000
E	2	001
I	1	010
M	2	011
O	1	100
T	2	101

Code word: 000100011011010101101001001
Size: 27 bits
Using Huffmann compression we can do

Alphabet	Probability
E	$\frac{2}{9}$
M	$\frac{2}{9}$
T	$\frac{2}{9}$
C	$\frac{1}{9}$
I	$\frac{1}{9}$
O	$\frac{1}{9}$

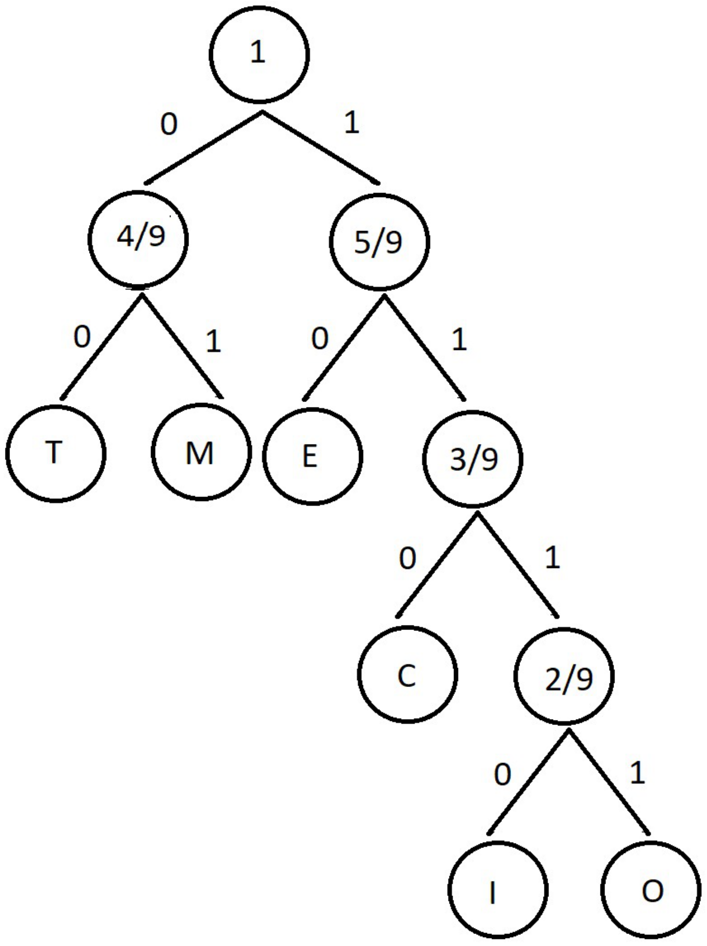


Figure 2: Encoded Hufmann Tree

Encoded Hufmann tree

Code word for 'COMMITTEE' becomes 00100011010000011110101
Size=23 bits Hence, size is reduced(23 is less than 27)
Efficiency computation:

$$\eta = \frac{H(s)}{L_{avg}}$$

H(s) =-$\sum_{i=0}^{N-1} P_i \log_2 P_i$

$$= (\tfrac{2}{9}\log_2 \tfrac{2}{9}) + 3(\tfrac{1}{9}\log_2 \tfrac{1}{9})$$

$$L_{avg} = \sum_{k=1}^{N} P_k I_k$$

$$= 2 \times \tfrac{2}{9} + 2 \times \tfrac{2}{9} + 2 \times \tfrac{2}{9} + 3 \times \tfrac{1}{9} + 4 \times \tfrac{1}{9} + 4 \times \tfrac{1}{9}$$

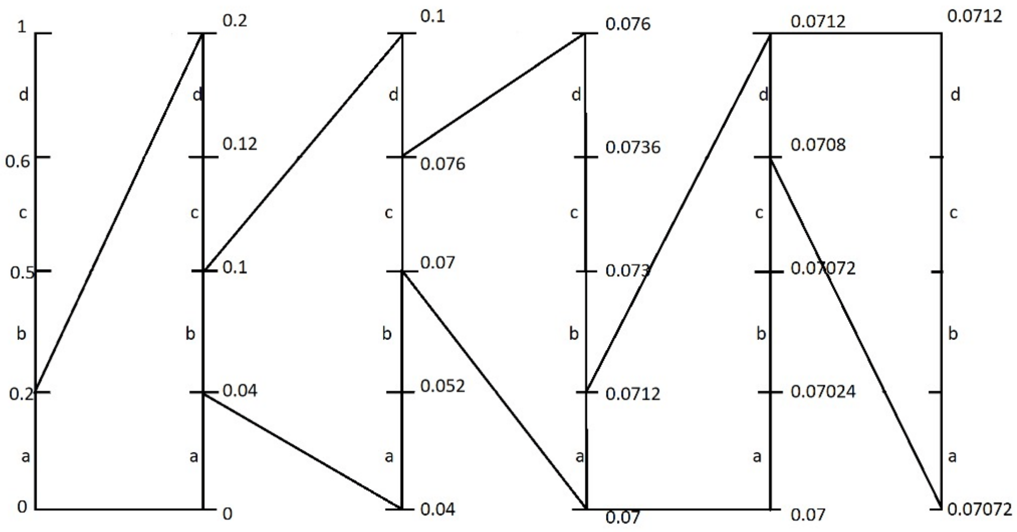
$$= 2.55$$

$$\eta = \frac{2.503}{2.55}$$

$= 97.97\%$ efficiency

7.1.2 Arithmetic Encoding

Consider the input string abcad and their respective probabilities are P(a)=0.2,P(b)=0.3,P(c)=0.1,P(d)=0.4 encode using AE.



Difference(d) = upper limit - lower limit
Range:- lower limit + d * (Probability of particular symbol)

Example:-
In case-1,
When a is expanded, we get d=0.2 - 0 = 0.2
Range a: 0 + 0.2 * (0.2) = 0.04
Range b: 0.04 + 0.2 * (0.3) = 0.1
Range c: 0.1 + 0.2 * (0.1) = 0.12
Range d: 0.12 + 0.2 * (0.4) = 0.2
Now we will get the final coded output as
  Final = (lower-upper)/2
  Final =(0.07012 + 0.07072)/2
  Final = 0.07096

7.2 Decoding

7.2.1 Huffman Decoding

Decode the messege “00010110011011000” for given symbols and probability.

Alphabet	Probability
a	$\frac{4}{9}$
l	$\frac{2}{9}$
m	$\frac{2}{9}$
y	$\frac{1}{9}$

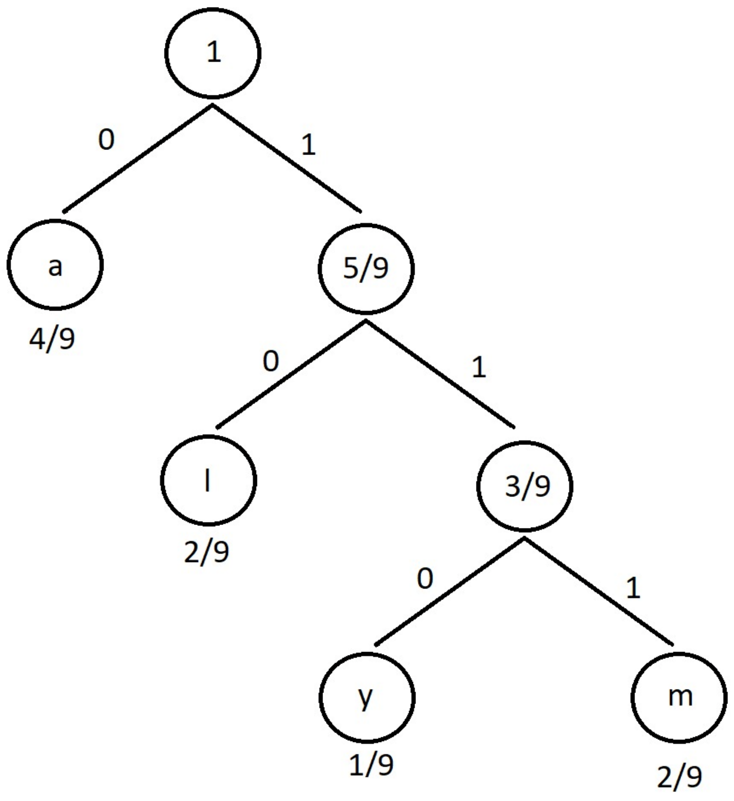


Figure 3: Decoded Hufmann Tree

'MALAYALAM' is decoded string.

Properties of Huffmann code:-

It gives unique encoded and decoded code.

Its process is instantaneous.

It produces a lossless compression of image.

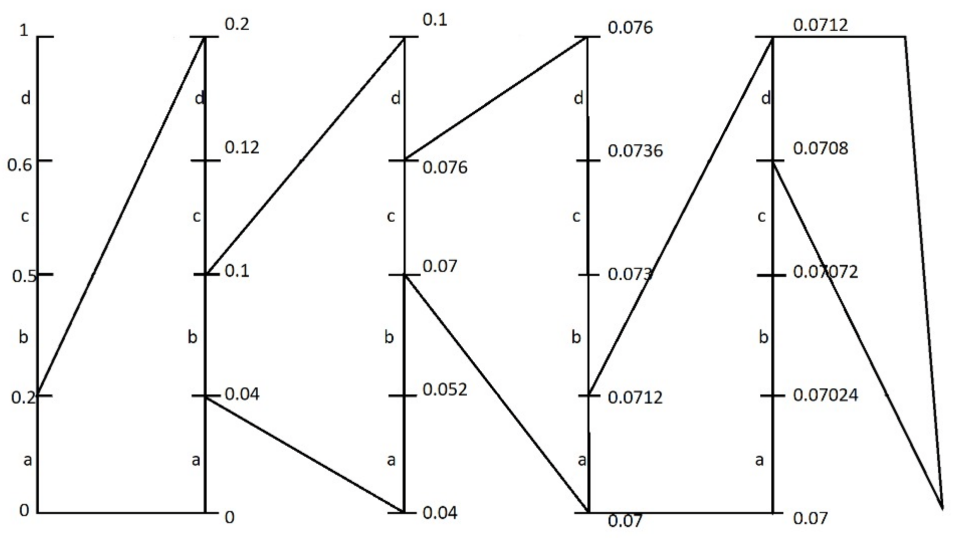
7.2.2 Arithmetic Decoding

Let us decode the encoded messege as we have get in the previous example.

To decode the encoded messege $\rightarrow$ 0.07096 $\rightarrow$ messege

We have been given the probabilities

$P(a) = 0.2$, $P(b) = 0.3$, $P(c) = 0.1$, $P(d) = 0.4$, Input $= 0.07096$



We have to select the interval in which the encoded code lies and zoom in it and further we have to do the same thing till we come very rear (3 places after decimal) to the encoded code.

We also have to note whose range we have zoomed in to get our decoded code.

In this case it is ‘abcad’.
The method to find the gap between two consecutive range remains the same as we have done in arithmetic encoding.

Advantages:-

It follows lossless compression methods in which no data is lost on being compressed.
It encodes the whole life as a sequence of symbols into a single decimal number.
It has advantages over Huffmann encoding as its compression ratio is 5-10 % letter than HE.
It only consists of few arithmetic operations thus its complexity is reduced AC is better than HC.

8 Compression Of B-P Frame

8.1 Colour Space Converter

The process of colur space converter is same as discussed in Compression Of I-Frame.

8.2 FDCT

The proccess of FDCT(Forward Discrete Cosine Transform) is same as discussed in Compression of I$_{Frame}$.

8.3 Motion Estimator

8.3.1 PEL recursive algorithm

A new pel-recursive motion estimation algorithm for video coding applications is presented. The derivation of the algorithm is based on recursive least-squares estimation that minimises the mean-square prediction error. A comparison with the modified steepest-descent gradient estimation technique algorithm shows significant improvement in terms of mean-square prediction error performance

8.3.2 Block Matching Algorithm

The Block Matching is a temporal compression technique used in the video encoding. The main purpose of this method is to determine the displacements of each block of pixels between two successive frames. This technique, performed in the step of motion estimation, occupies the majority of the total time of video coding. The aim of this work is to give a comparative study various search algorithms by Block Matching.It simply use to reduce the amount of data which is required to store and transmit the video. The motion estimation process is an inextricable part of the video coding as it removes the temporal redundancy between successive frames of video sequences.

Algorithm:-

The PSNR or Signal to Noise Ratio is given by where MSE is the Mean Square Error calculated by. (H×L) is the frame size, X and X are respectively the original frame and the predicted frame. We note that

for color frame, equations and are applied separately on the Red component (R), green component (G) and blue component (B) of the RGB color space. The global PSNR of the R, G and B components is the PSNR average of these three components.
$PSNR = 10 \log_{10}(2552MSE)MSE$

8.4 Entropy Encoding and Decoding

The proccess of Entropy Encoding and Decoding is same as discussed in Compression of I-Frame.

9 Dezigzag scan

After getting 1×64 matrix from entropy decoder, to pass this matrix to dequantizer, we have to convert it into a 8×8 array.

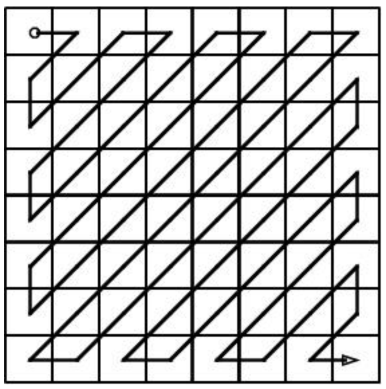


Figure 4: DeZigZag Scan

10 Dequantization

The input to the dequantisation block is a vector of 1x64 matrix for a case of 8×8 pixel image and based on the particular colour channel there is a fixed matrix of of N×N, where N is the pixel of the image and we get the output as the product of a particular element of the vector with that of the common dequantiser matrix. This output gets passed on as the input to IDCT.

11 IDCT (Inverse Discrete Cosine Transform)

The output of IDCT will be the image itself which is to be displayed and its input would be a N×N matrix which is given as output by the dequantizer.

Its formula is:

$$U(N) = \frac{2}{N}\alpha(k)\sum_{n=0}^{N-1} V(K)\cos\left(\frac{\pi(2n+1)K}{2N}\right), N = 0, 1, 2, 3, \ldots, N-1$$

U(n) is also given by U = C' V C which is calculated from the above formula.
Example:-
If we consider the above example only, then by the above formula of IDCT we should be able to get the original image U. Lets have fun and verify it.

We have got C matrix as from the example of DCT as:

$$C = \begin{bmatrix} 0.5 & 0.65 & 0.5 & 0.27 \\ 0.5 & 0.27 & -0.5 & -0.65 \\ 0.5 & -0.27 & -0.5 & 0.65 \\ 0.5 & -0.65 & 0.5 & -0.27 \end{bmatrix}$$

$$\mathrm{V} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The image matrix U is given by U $= C^T V C$

$$\mathrm{U} = \begin{bmatrix} 0.5 & 0.65 & 0.5 & 0.27 \\ 0.5 & 0.27 & -0.5 & -0.65 \\ 0.5 & -0.27 & -0.5 & 0.65 \\ 0.5 & -0.65 & 0.5 & -0.27 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.65 & 0.27 & -0.27 & -0.65 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.27 & -0.65 & 0.65 & -0.27 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Thus , we have got the original image with the help of IDCT which would be displayed as a final image.