EE596 – MINI PROJECT

DESIGN OF AN HYBRIC CODEC FOR IMAGE AND VIDEO CODING

KAJANANAN S.

E/18/165

24/01/2024

INTRODUCTION

Hybrid video coding is an advanced coding technique which is derived from both predictive coding and transform coding. Hybrid video coding framework is commonly used in modern video coding standards, e.g., H.26x, MPEG2/4, AVS, HEVC, etc. In this project, I will implement a simplified hybrid video codec with coding tools like discrete cosine transformation, quantization, prediction, Intra coding and entropy coding and the impact of each tool on the codec's performance will be investigated. This project will be contained a block based image compression system, block based video compression system and basic functions of image and video coding in different quantization levels will be analyzed. And also I will be analyzing the impact of the transmission rate with the distortion of a particular frame and how the image quality varying according to transmission rate in different quantization levels.

BASIC CONCEPTS USED FOR COMPRESSION SYSTEM

In this section we will be briefly discussing on some basic concepts of image and video coding which I have been using throughout the project.

DISCRETE COSINE TRANSFORM (DCT)

The discrete cosine transform (DCT) helps separate the image into parts (or spectral sub-bands) of differing importance (with respect to the image's visual quality). The DCT is like the discrete Fourier transform: it transforms a signal or image from the spatial domain to the frequency domain.

The general equation for a 2D (*N* by *M* image) DCT is defined by the following equation:

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] cos\left[\frac{\pi.v}{2.M}(2j+1)\right].f(i,j)$$

And the corresponding *inverse* 2D DCT transform is simple *F-1*(*u*, *v*), i.e.: Where

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

Each block consists of 8×8 elements, corresponding to x and y varying from 0 to 7. The highest value is shown in white. Other values are shown in grays, with darker meaning smaller

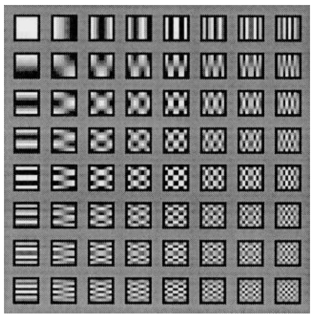
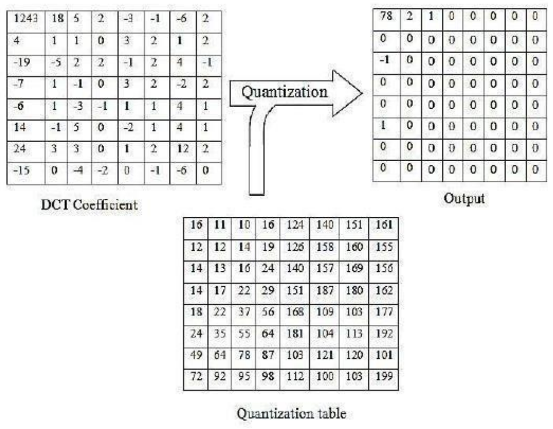


Figure 01: The 64 (8 x 8) DCT basis functions

QUANTIZATION

Quantization, involved in image processing, is a loss compression technique achieved by compressing a range of values to a single quantum value.A typical video codec works by breaking the picture into discrete blocks (8×8 pixels in the case of MPEG). These blocks can then be subjected to discrete cosine transform (DCT) to calculate the frequency components, both horizontally and vertically. The resulting block (the same size as the original block) is then pre- multiplied by the quantization scale code and divided elementwise by the quantization matrix, androunding each resultant element.

A common quantization process behaves like following



1243	18	5	2	-3	-1	-6	2
4	1	1	0	3	2	1	2
-19	-5	2	2	-1	2	4	-1
-7	1	-1	0	3	2	-2	2
-6	1	-3	-1	1	1	4	1
14	-1	5	0	-2	1	4	1
24	3	3	0	1	2	12	2
-15	0	-4	-2	0	-1	-6	0

DCT Coefficient

78	2	1	0	0	0	0	0
0	0	0	0	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
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Output

16	11	10	16	124	140	151	161
12	12	14	19	126	158	160	155
14	13	16	24	140	157	169	156
14	17	22	29	151	187	180	162
18	22	37	56	168	109	103	177
24	35	55	64	181	104	113	192
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	199

Quantization table

Figure 02: Quantization table represents common JPEG Quantization matrix

HUFFMAN CODING

Huffman Coding (also known as Huffman Encoding) is a lossless algorithm for doing data compression and it forms the basic idea behind file compression. In this algorithm, a variable- length code is assigned to input different characters. The code length is related to how frequently characters are used. Most frequent characters have the smallest codes and longer codes for least frequent characters.

MOTION ESTIMATION AND MOTION VECTOR

Motion estimation examines the movement of objects in an image sequence to try to obtain vectors representing the estimated motion. Motion compensation uses the knowledge of object motion so obtained to achieve data compression. In interframe coding, motion estimation and compensation have become powerful techniques to eliminate the temporal redundancy due to high correlation between consecutive frames.

In real video scenes, motion can be a complex combination of translation and rotation. Such motion is difficult to estimate and may require large amounts of processing. However, translational motion is easily estimated and has been used successfully for motion compensated coding.

BLOCK MATCHING ALGORITHM

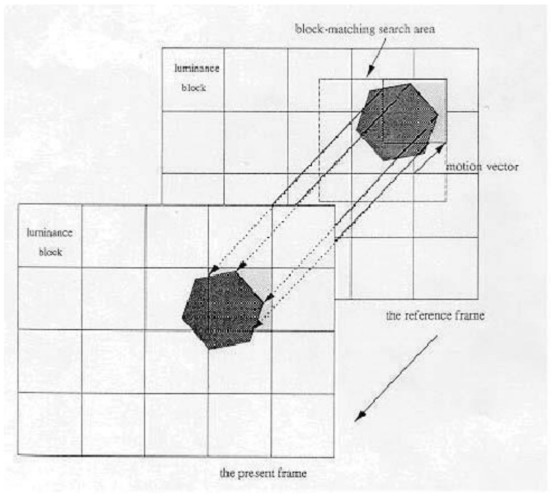


Figure 03: Block Matching Algorithm

Figure:03 Illustrates a process of block-matching algorithm. In a typical BMA, each frame is divided into blocks, each of which consists of luminance and chrominance blocks. Usually, for coding efficiency, motion estimation is performed only on the luminance block. Each luminance block in the present frame is matched against candidate blocks in a search area on the reference frame. These candidate blocks are just the displaced versions of original block. The best (lowest distortion, i.e., most matched) candidate block is found, and its displacement (motion vector) is recorded.

Consequently, the motion vector and the resulting error can be transmitted instead of the original luminance block; thus, interframe redundancy is removed and data compression is achieved. At receiver end, the decoder builds the frame difference signal from the received data and adds it to the reconstructed reference frames. The summation gives an exact replica of the current frame. The better the prediction the smaller the error signal and hence the transmission bit rate.

INTRA PREDICTION FRAME CODING

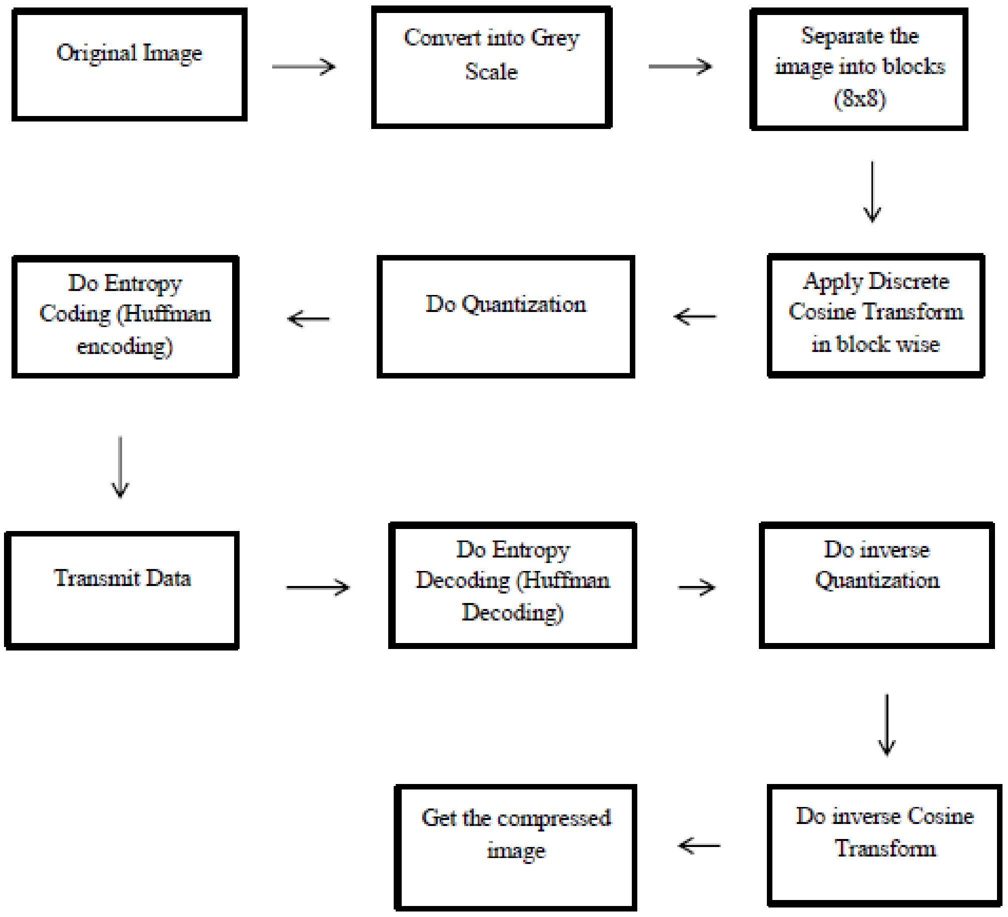
Intra-frame prediction exploits spatial redundancy, i.e., correlation among pixels within one frame, by calculating prediction values through extrapolation from already coded pixels for effective delta coding. It is one of the two classes of predictive coding methods in video coding. Its counterpart is inter-frame prediction which exploits temporal redundancy. Temporally independently coded so- called intra frames use only in intra coding. The temporally coded predicted frames (e.g. MPEG's P- and B-frames) may use intra- as well as inter-frame prediction.

The term intra-frame coding refers to the fact that the various lossless and lossy compression techniques are performed relative to information that is contained only within the *current frame* and *not relative* to any other frame in the video sequence. In other words, no temporal processing is performed outside of the current picture or frame. Non-intra coding techniques are extensions tothese basics. It turns out that this block diagram is very similar to that of a JPEG still image video encoder, with only slight implementation detail differences.

IMAGE COMPRESSION SYSTEM

This image compression system consists of concepts like DCT/IDCT, Quantization, Huffman encoding, macro block-based coding and etc.

Flow Chart for the system


Original Image
Convert into Grey Scale
Separate the image into blocks (8x8)
Apply Discrete Cosine Transform in block wise
Do Quantization
Do Entropy Coding (Huffman encoding)
Transmit Data
Do Entropy Decoding (Huffman Decoding)
Do inverse Quantization
Do inverse Cosine Transform
Get the compressed image

Pseudo Code of Image Compression in Grey Scale

```
1. Get the image into Matlab using the command ‘imread’.
2. Convert the image into greyscale using the command ‘rgb2grey’.
3. Convert pixel values to double precision.
4. Apply Discrete Cosine transform after separating the image matrix into blocks of 8x8using the
command ‘blockproc’.
5. Round off pixel values to upper digit after multiplying by 1000 by using ‘ceil’ command.
(Elements are multiplied by 1000 to increase the accuracy as the values are very small)
6. Do the Quantization by dividing the matrix by the different quantization level.
7. Do Entropy encoding by applying Haffman Encoding function.
Now the compressed image code can be transmitted or stored as text format.
8. Do Entropy Decoding to the matrix after receiving by applying Haffman Decoding function.
9. Divide elements by 1000 to get them into the range of 0-1.
10. Do inverse quantization by multiplying elements by quantization level.
Do the inverse Cosine Transform on blocks using ‘blockproc’ function. 12.Find PSNR value and Compression ratio.
```

RESULTS OF IMAGE CODEC

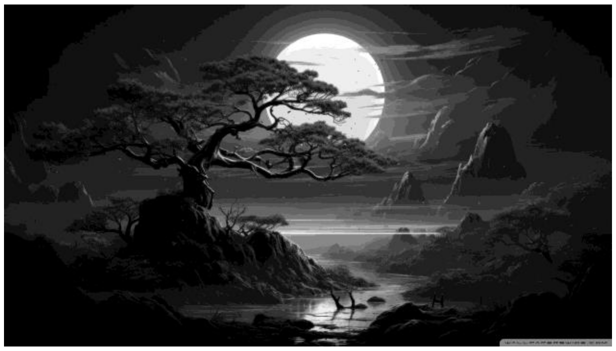


Figure 04: Original gray scale image (Size: 164 kB)

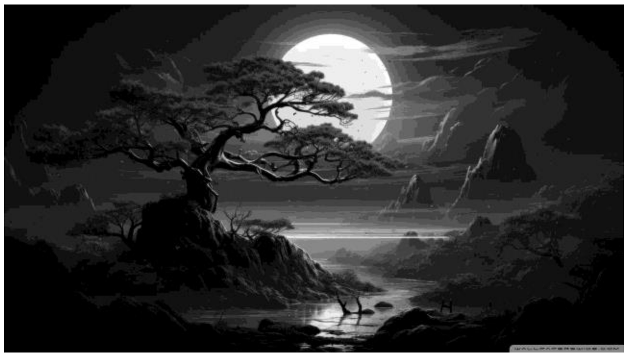


Figure 05: Reconstructed image(Size: 106kB)

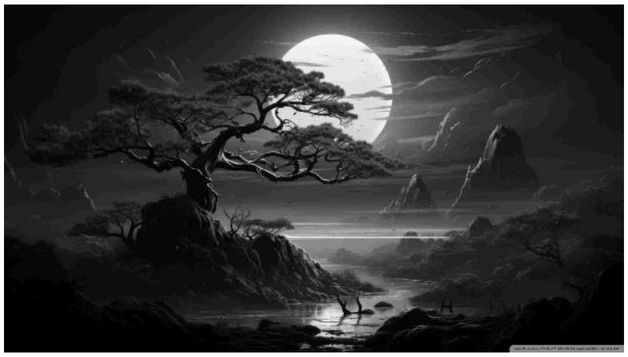


Figure 06: Compressed image when Qmin (Size = 77.8 kB)

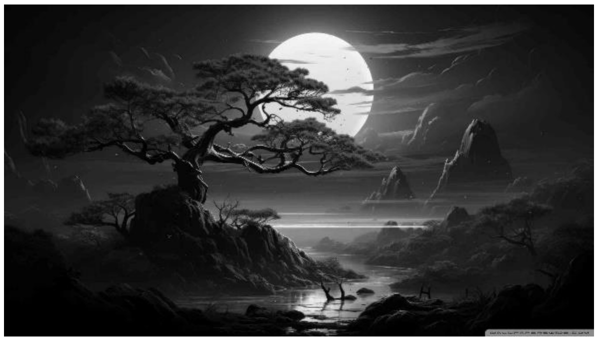


Figure 07: Compressed image when Qmid (Size = 64 kB)

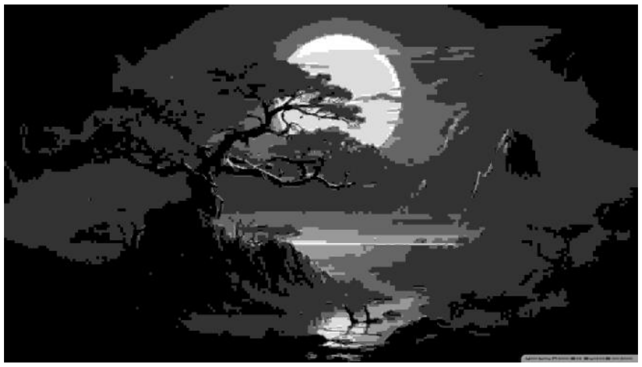
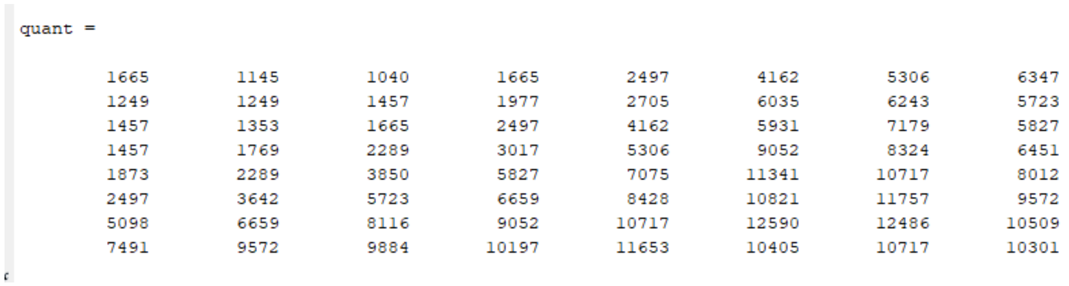


Figure 08: Compressed image when Qmax (Size = 31.8 kB)



```
quant =

        1665        1145        1040        1665        2497        4162        5306        6347
        1249        1249        1457        1977        2705        6035        6243        5723
        1457        1353        1665        2497        4162        5931        7179        5827
        1457        1769        2289        3017        5306        9052        8324        6451
        1873        2289        3850        5827        7075       11341       10717        8012
        2497        3642        5723        6659        8428       10821       11757        9572
        5098        6659        8116        9052       10717       12590       12486       10509
        7491        9572        9884       10197       11653       10405       10717       10301
```

Figure 09: The matrix which used for achieve required bitrate

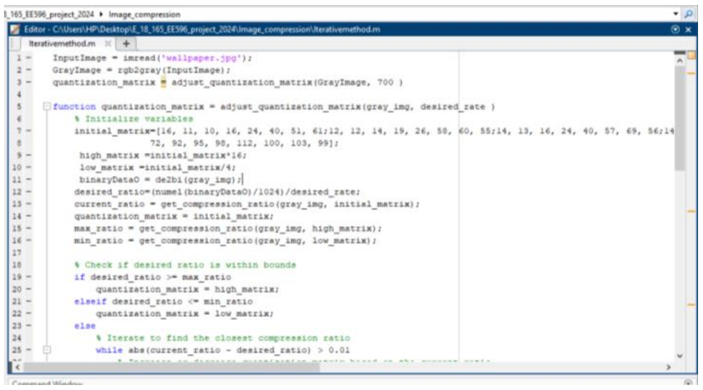
IMAGE COMPRESSION OPTIMISATION

Iterative method

In the image compression Adjusting the compression ratio of the image encoder to meet the following bit rates is an important thing that needs to be optimized. For that, I used an iterative method that scales a basic quantization matrix and achieves the required compression ratio. I used the required bit rate as a stopping criterion for my iterative algorithm. And it will give a quantization matrix as an output.

Pseudo code for iterative method

```
Initialize variables
Calculate desired,current compression ratio
If desired ratio is greater than or equal to maximum ratio
            quantization_matrix = high_matrix
  Otherwise
            quantization_matrix = low_matrix
Else
  While current ratio is not close enough to desired ratio
      If current ratio is less than desired ratio
            quantization_matrix = quantization_matrix * 1.1
      Otherwise
            quantization_matrix = quantization_matrix * 0.9
      Calculate new compression ratio
  End while
End if
```

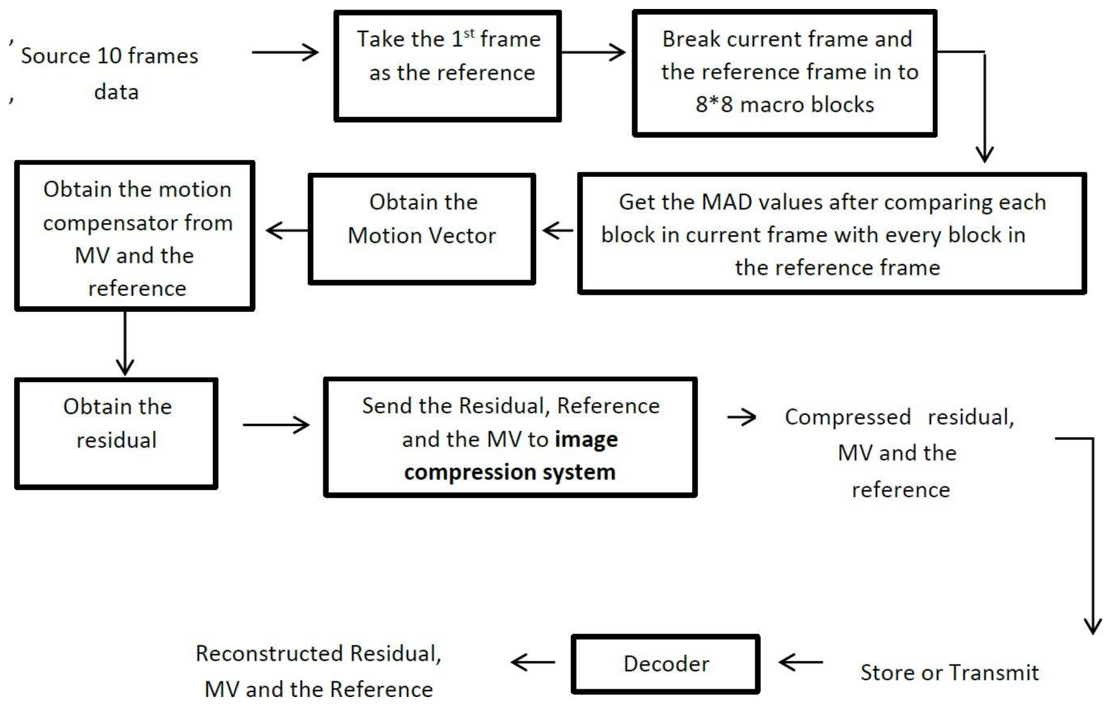


```
InputImage = imread('wallpaper.jpg');
GrayImage = rgb2gray(InputImage);
quantization_matrix = adjust_quantization_matrix(GrayImage, 700 )

function quantization_matrix = adjust_quantization_matrix(gray_img, desired_rate )
    % Initialize variables
    initial_matrix=[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
                    72, 92, 95, 98, 112, 100, 103, 99];
     high_matrix =initial_matrix*16;
     low_matrix =initial_matrix/4;
     binaryData0 = de2bi(gray_img);
    desired_ratio=(numel(binaryData0)/1024)/desired_rate;
    current_ratio = get_compression_ratio(gray_img, initial_matrix);
    quantization_matrix = initial_matrix;
    max_ratio = get_compression_ratio(gray_img, high_matrix);
    min_ratio = get_compression_ratio(gray_img, low_matrix);

    % Check if desired ratio is within bounds
    if desired_ratio >= max_ratio
        quantization_matrix = high_matrix;
    elseif desired_ratio <= min_ratio
        quantization_matrix = low_matrix;
    else
        % Iterate to find the closest compression ratio
        while abs(current_ratio - desired_ratio) > 0.01
```

Figure 10: The code of iterative method

VIDEO COMPRESSION SYSTEM

Video compression is the process of reducing the total number of bits needed to represent a given image or video sequence. Here I used 20 frames. This video compression system consists of concepts like Motion Vectors, Motion estimation, Intra prediction frame coding, macro block based coding and etc.



Source 10 frames data
Take the 1st frame as the reference
Break current frame and the reference frame in to 8*8 macro blocks
Get the MAD values after comparing each block in current frame with every block in the reference frame
Obtain the Motion Vector
Obtain the motion compensator from MV and the reference
Obtain the residual
Send the Residual, Reference and the MV to image compression system
Compressed residual, MV and the reference
Store or Transmit
Decoder
Reconstructed Residual, MV and the Reference

Figure 11: Flow chart for the video codec

Pseudo code

Encoder site

```
Start
Read video frames ,resize ,gray conversion and Macroblock division
  For each frame:
        Calculate motion vectors using SAD and stored
        Calculate residual by subtracting previous frame
  For each frame:
        apply down sampling
        Encode the residuals using quantization and huffman
Store the dictionary and encoded residual
End
```

```
Decoder site
Start
 Load encoded data, dictionary and motion vectors
Initialize variables and parameters
Read encoded residuals from text file
        For each frame:
                Decode the residual & I using dictionary and quantization table
                Store the decoded residual, I
                Apply up sampling
For each frame:
        Calculate predict frame using motion vectors and previous frame
        Store the predicted frame
        Reconstruct final frames by adding decoded residuals to predicted frames
Apply deblocking filter to reconstructed images
Save filter applied reconstructed frames as a video
End
```

RESULTS OF VIDEO CODEC

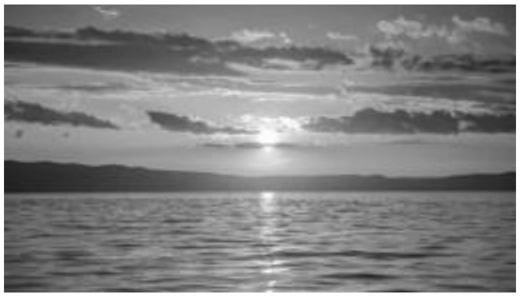


Figure 12:Reference frame

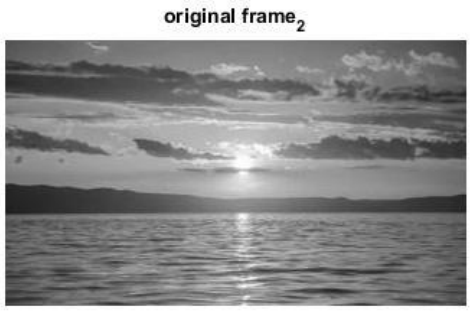


Figure 13:Original frame 2

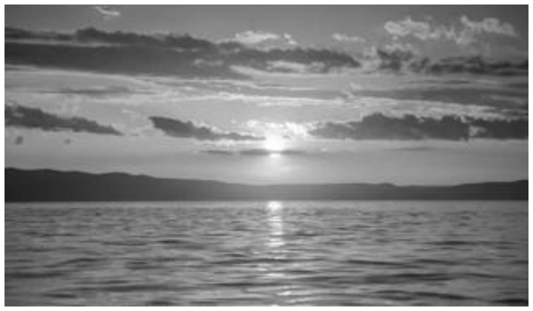


Figure 14:Predicted frame 2

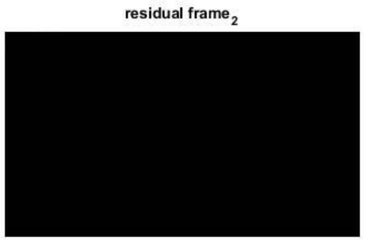


Figure 15: Residual frame 2

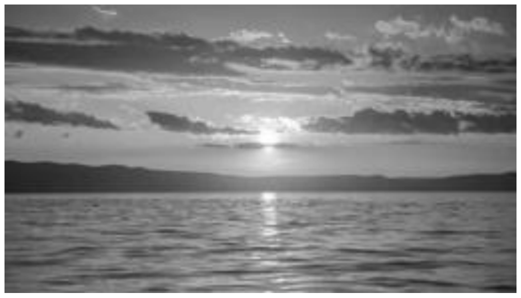


Figure 16: Reconstructed frame

OPTIMIZATION IN VIDEO COMPRESSION

- To remove spatial redundancy did down sampling(encoder) and up sampling(decoder)
- Improve quality using Deblocking filters A deblocking filter is a video filter applied todecoded compressed video to improve visual quality and prediction performance by smoothing the sharp edges

- By using the above iteration method ,deblocking filter(not that much optimizable)and ratedistortion quality curve can control the video image codec bit rate

IMPROVEMENTS

- Build Lagrangion optimization to control the bit rate.
- We can build context adaptive quantization matrix finder with help of deep learning techniques it will help to control the bit rate.

REFERENCES

1. https://huyunf.github.io/blogs/2017/11/20/h264_deblocking_algorithm/ (Access date 23/01/2024)
2. https://www2.seas.gwu.edu/~ayoussef/papers/ImageDownUpSampling-CISST99.pdf(Access date 24/01/2024)
3. https://www.mediakind.com/blog/improving-video-compression-with-ai/ (Access date 24/01/2024)