EE 596 : IMAGE AND VIDEO CODING MINI PROJECT

WEERASINGHA W.K.H.M.
E/18/383

SEMESTER 7

24/01/2024

1) OBJECTIVE

1) To investigate the basic functions of an image coding system.
2) To investigate the basic functions of a block-based video coding system.
3) To investigate the Rate-Distortion optimization techniques used in video coding.

2) INTRODUCTION

This project dives into the realm of advanced techniques by crafting a simplified hybrid video codec. This special codec borrows the best from predictive coding and transforms coding, making it a key player in modern video standards like H.26x, MPEG2/4, AVS, and HEVC.

The goal of this research is to develop a more straightforward hybrid video codec that integrates key coding functions such quantization, prediction, entropy coding, and discrete cosine transformation (DCT). By using these tools and analyzing their effects on codec performance, we may learn a great deal about their individual contributions and how they work together to maximize video compression. This study will give important insights into the trade-offs involved in video coding, which will advance our understanding of the critical elements impacting the effectiveness and caliber of contemporary video coding standards.

As the first stage, a basic image coding system featuring three levels of quantization for varied quality outputs for low, medium, and high qualities. It has introduced an intelligent algorithm that enables the compression output to adapt to any given bitrate, offering flexibility and efficiency. The second stage has built upon the success of the image compression system and enhanced the design to support video compression. This involves incorporating macro block-based coding, basic motion estimation, and intra-prediction, providing the groundwork for efficient video encoding.

This project promises a hands-on exploration of the intricate world of video coding, unraveling the impact of discrete cosine transformation, quantization, prediction, and entropy coding on the overall codec performance.

3) METHODOLOGY

3.1) Stage 1: Basic implementation: Image compression

Image compression is the technique of lowering the file size of an image without sacrificing its quality. In order to conserve storage space or lower the amount of bandwidth needed for transmission, it attempts to eliminate unnecessary or redundant data from photographs. Figure 1, shows the basic image compression system.

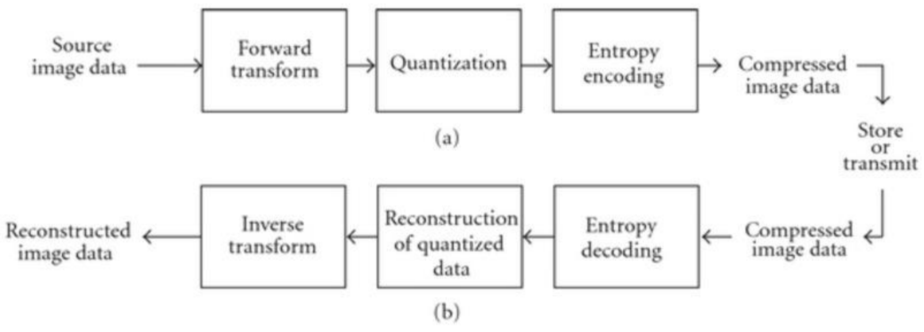


Figure 1: Basic Image Compression System

Steps in basic image compression system:

1) Forward transform
   The Discrete Cosine Transform (DCT) is the transform that is most frequently employed in image compression. By converting the picture data from the spatial domain to the frequency domain, the forward transform makes the frequency components in the image visible.
2) Quantization
   Through the use of a quantization factor or predefined step size, the precision of the coefficients is reduced through the process of quantization. This procedure lowers the quantity of data required to depict the image by eliminating less important information. Quantization establishes the trade-off between compression ratio and image quality and introduces a loss of information. Image compression algorithms divide images into discrete blocks (8x8 pixels), calculate frequency components using DCT, and pre-multiply by the quantization scale code. The resulting block is then divided by the quantization matrix, with each element rounded, resulting in a rounded image.

   JPEG standard quantization 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, 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]

3) Entropy encoding
   Entropy encoding methods, such as run-length encoding, arithmetic encoding, or Huffman encoding, are used to encode the quantized coefficients. The size of the compressed data is further decreased using entropy encoding, which gives shorter codes to coefficients that appear more frequently.
4) Compressed image data
   Following entropy encoding, the image data is compressed and saved. The encoded coefficients, which serve as a condensed representation of the original image, make up this representation.
5) Store or transmit
   Depending on the application, the compressed image data may be sent over a network or saved in a file. By reducing the file size, compression facilitates image storage and transmission. This text file contains compressed material that has been stored in it.
6) Entropy decoding
   The compressed picture data is subjected to the entropy decoding procedure at the receiving end. This entails applying the same entropy coding techniques that were applied during compression to decode the encoded coefficients back to their original form. The quantized data is represented by the decoded coefficients.
7) Reconstruction of quantized data
   Next, the quantized coefficients are multiplied by the factor or step size of the quantization process to reconstruct them. The goal of this phase is to approximate the original transformed coefficients using the quantized data.
8) Inverse transform
   An inverse transform is applied to the reconstructed quantized data to undo the effects of the forward transform. The inverse DCT (IDCT) is used when there is DCT compression. The data is transformed back into the spatial domain from the frequency domain using this technique.
9) Reconstructe the image data
   The reconstructed image data is the final product of the inverse processed data. Because of the lossy nature of compression, the reconstructed image data might not be a perfect duplicate of the original picture, but it does its best to maintain adequate visual quality while using less storage space or transmission bandwidth.

Figure 2, shows the flow of which have done in stage 1.

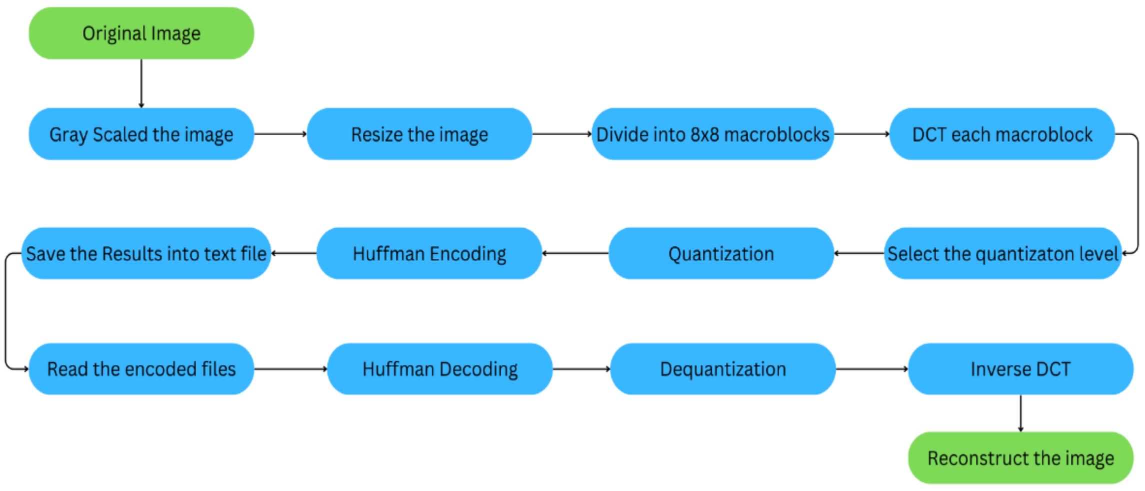


Figure 2: Basic Image Compression System used in stage 1

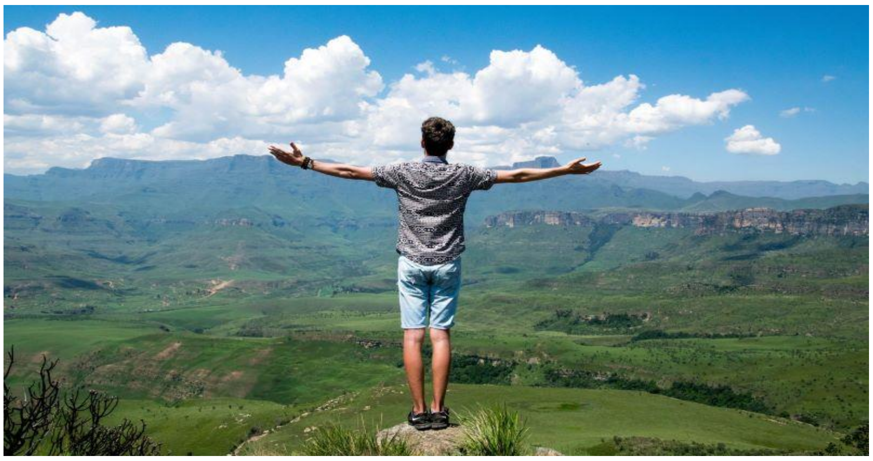


Figure 3: Original Colored Image

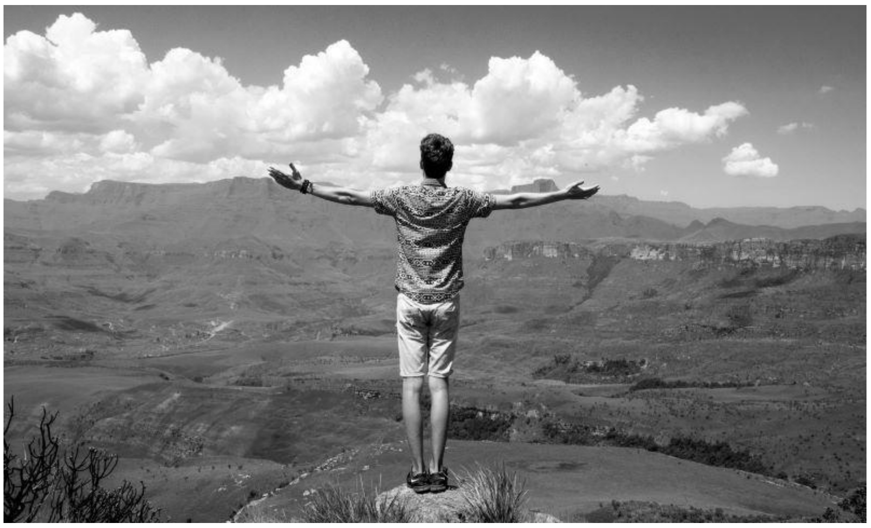


Figure 4: Original Gray scale Image

3.1.1) Results

Low Quality Quantization

Standard quantization matrix multiplication factor = 8
PSNR value = 31.13
Compression ratio = 7.27

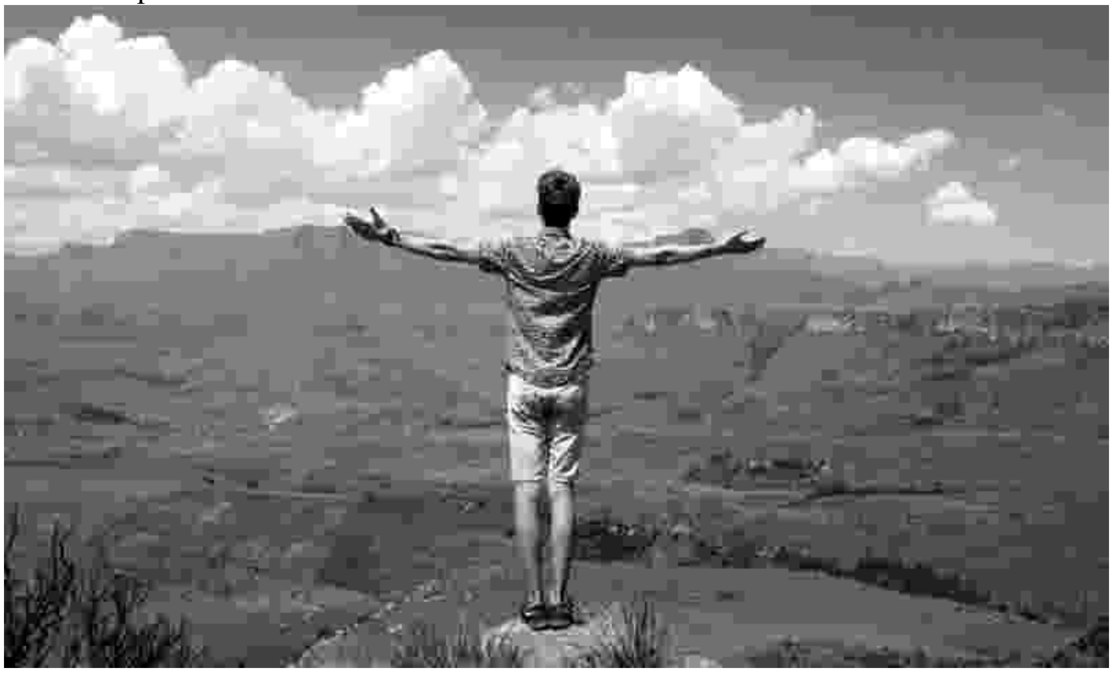


Figure 5: Decoded low quality image

Medium Quality Quantization

Standard quantization matrix multiplication factor = 1
PSNR value = 35.796
Compression ratio = 5.43

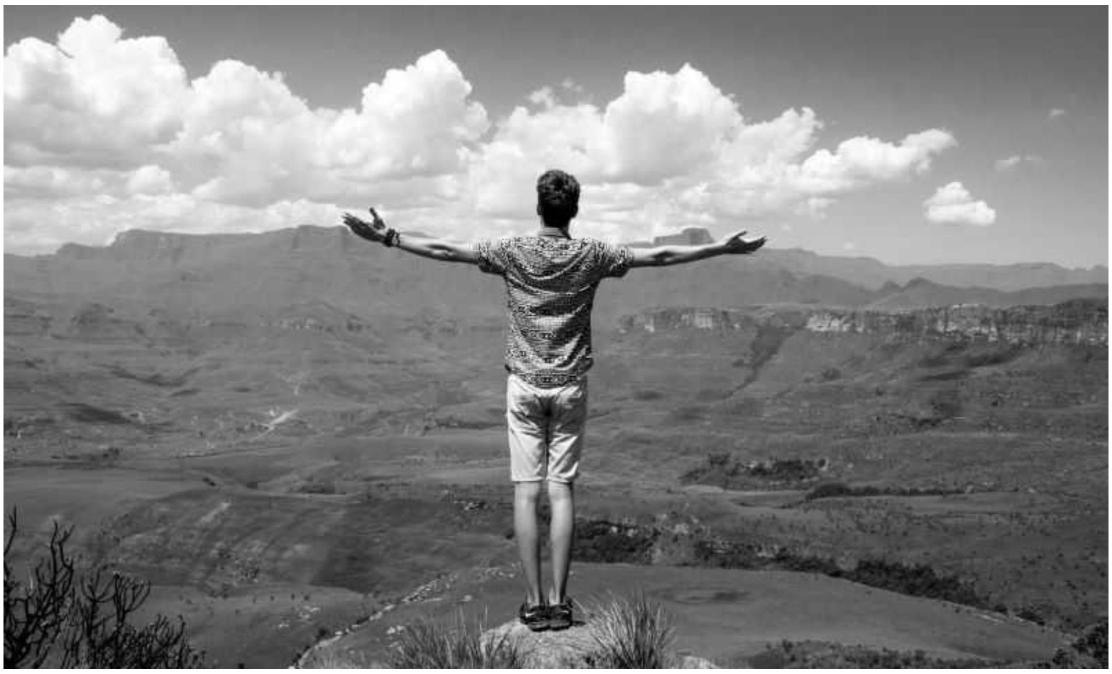


Figure 6: Decoded medium quality image

High Quality Quantization

Standard quantization matrix multiplication factor = (1/6)
PSNR value = 47.55
Compression ratio = 3.76

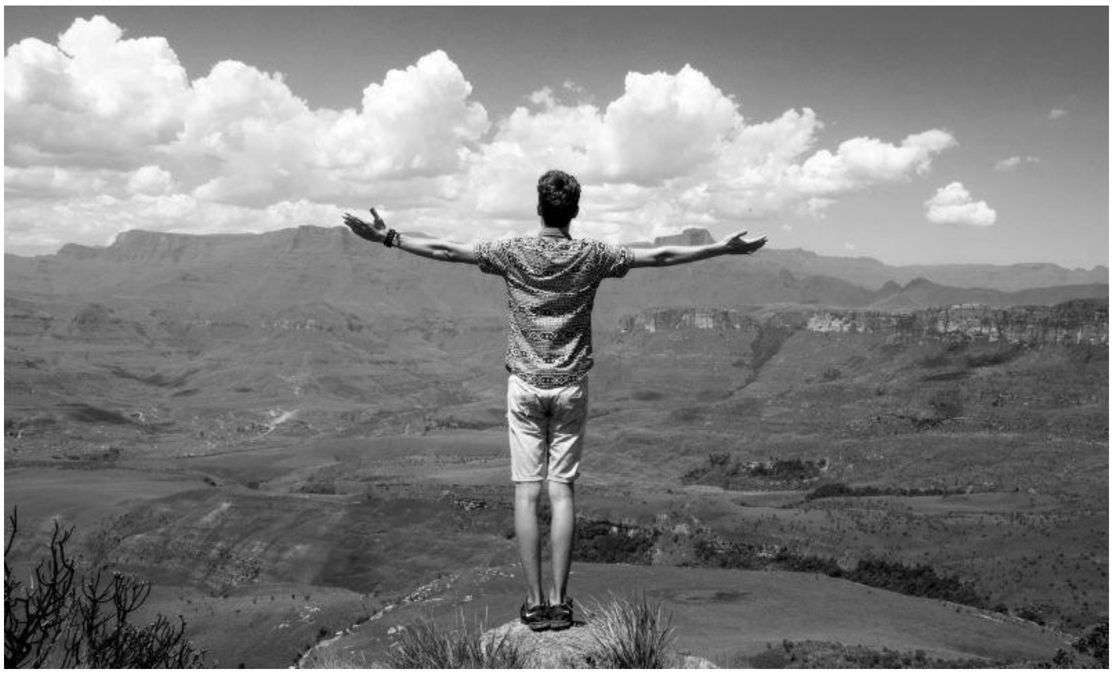


Figure 7: Decoded high quality image

3.1.2) Results

Bit rate of the channel (kbps) = last 3 digit of your E-number + 300

= 383 + 300

= 683

Standard quantization matrix multiplication factor = 1/0.667

PSNR value = 38.33

Compression ratio = 4.507

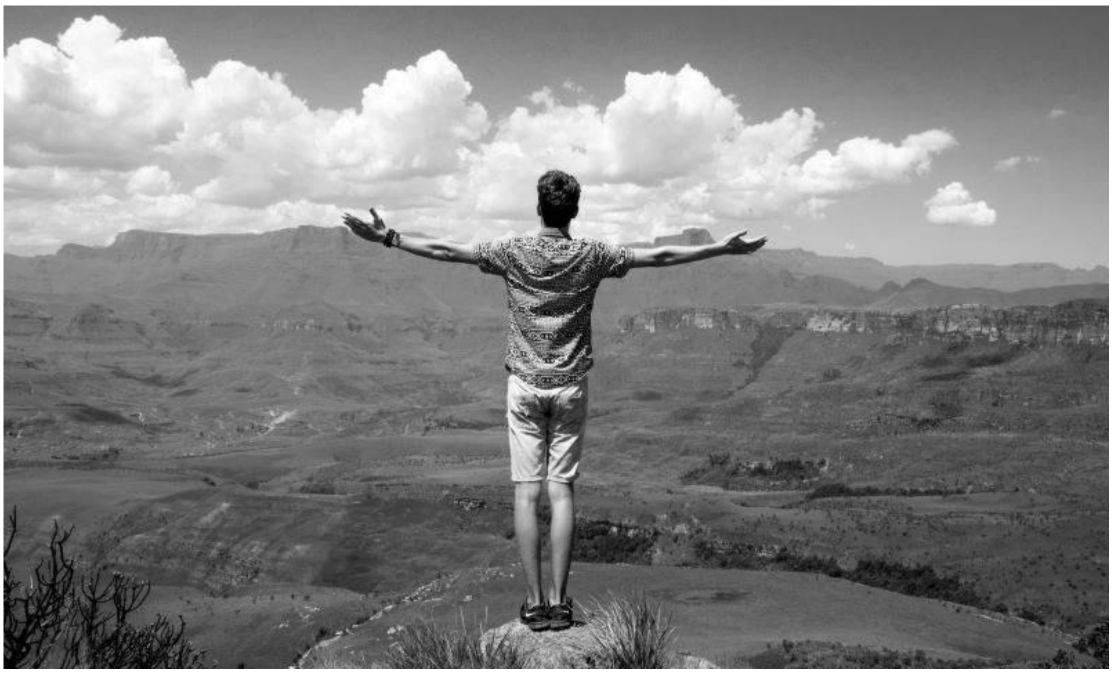


Figure 8: Reconstructed image according to E-number

3.1.3) Results

Bit rate of the channel (kbps) = 500

Standard quantization matrix multiplication factor = 1.9

PSNR value = 34.33

Compression ratio = 6.149

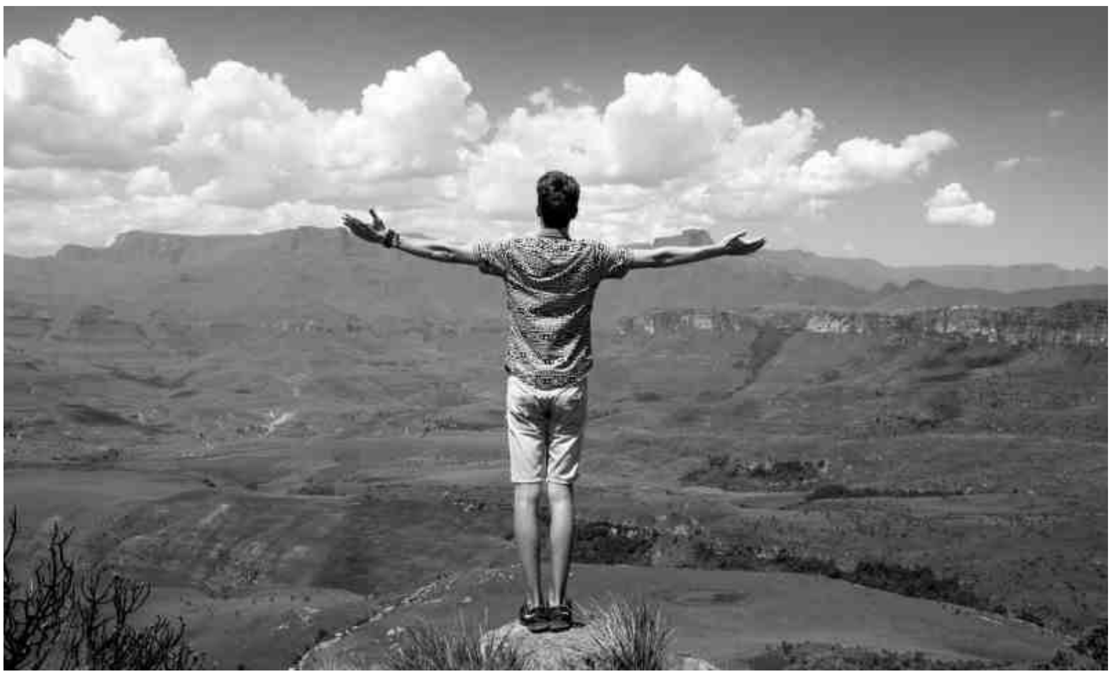


Figure 9: Reconstructed image according to Custom bitrate(500kbps)

In (3.1.3) stage, there was set a condition to fit the image compression in range 30 to 50 PSNR range. If when apply bitrate manually, the PSNR is not fit to the PSNR range code automatically fit the bit rate according to the PSNR range. For that there compare the compression ratio.

Figure 10, shows the flow chart of the python code for stage 1.

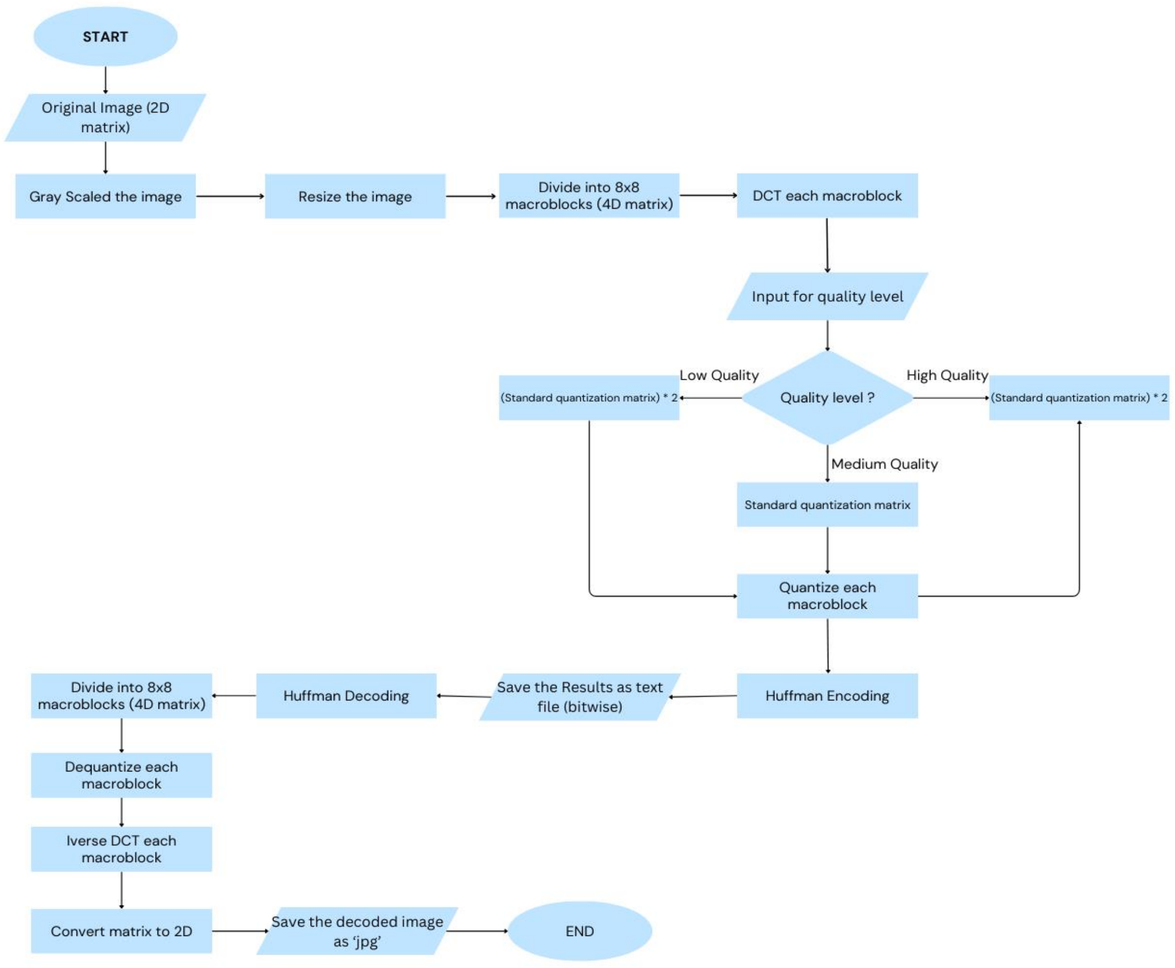

START
Original Image (2D matrix)
Gray Scaled the image
Resize the image
Divide into 8x8 macroblocks (4D matrix)
DCT each macroblock
Input for quality level
Quality level ?
Low Quality
High Quality
Medium Quality
(Standard quantization matrix) * 2
(Standard quantization matrix) * 2
Standard quantization matrix
Quantize each macroblock
Huffman Encoding
Save the Results as text file (bitwise)
Huffman Decoding
Divide into 8x8 macroblocks (4D matrix)
Dequantize each macroblock
Iverse DCT each macroblock
Convert matrix to 2D
Save the decoded image as 'jpg'
END

Figure 10: Flow chart for the code for stage 1

3.2) Stage 2: Basic implementation 2: Video compression

Video compression is the process of reducing the video file size while keeping visual quality. It uses a variety of approaches and algorithms to reduce redundancy and efficiently describe video frames.

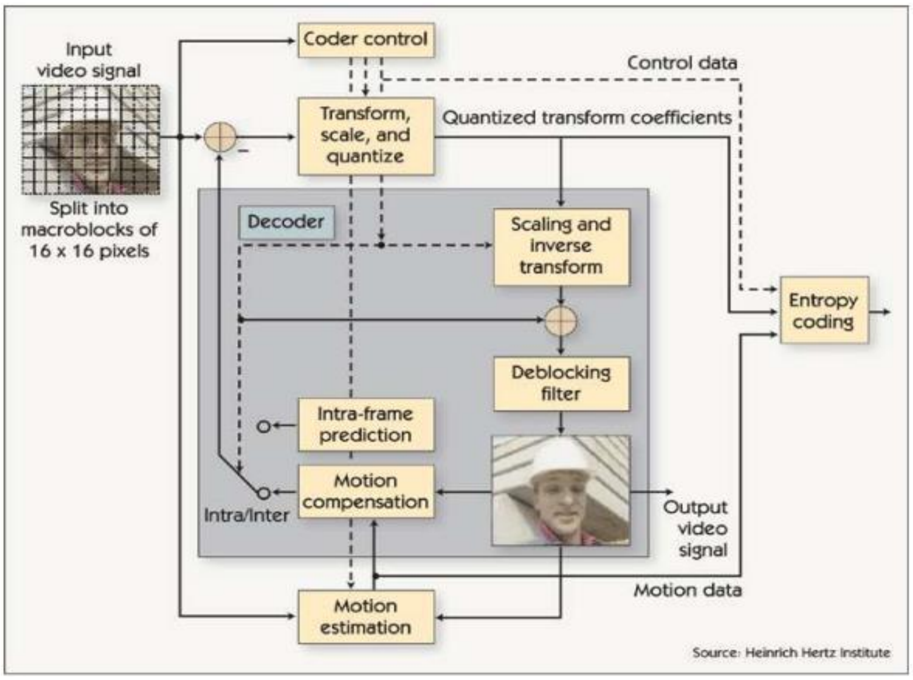


Figure 11: Basic video compression system

3.2) Process and Result

Step 1: **Video Preprocessing**

First load the frames of the video as a sequence. Then convert to gray scale and resize the gray scaled image to fit 8 by 8 marcoblocks.

In this scenario, the IPPP... structure is used for the video compression. As a result, the first frame is compressed without any motion vectors. The identical technology had previously been utilized for image compression. The resulting rebuilt image serves as the reference frame for the second frame. So, consider that first frame is the I frame for each 10 frames.

Figure 12: First frame of the video (I frame)

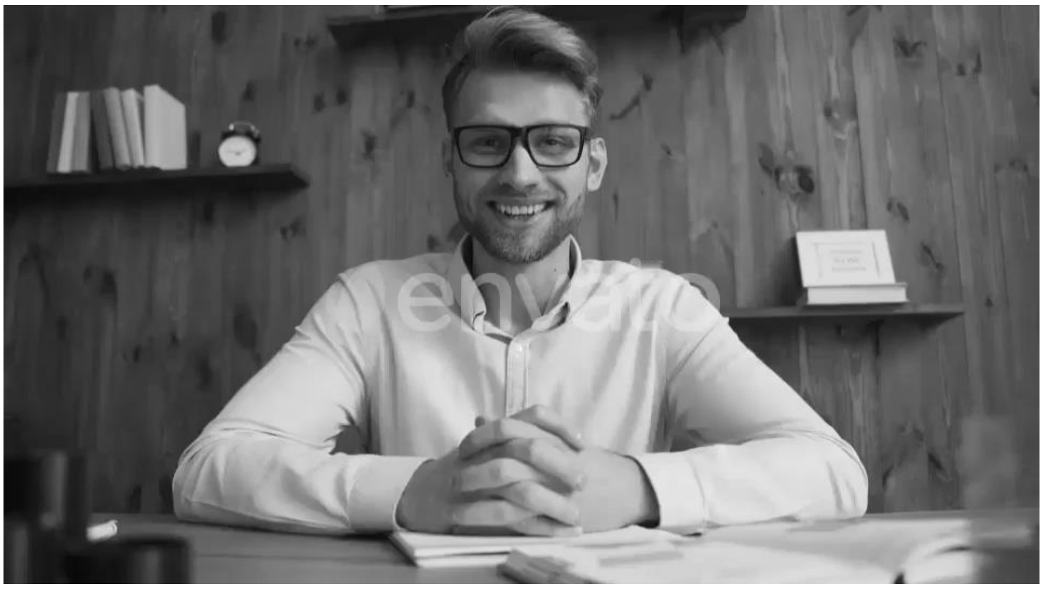


Figure 13: Gray scaled First frame (I frame)

Step 2: Macroblock-based Coding

There, divide each frame into 8 by 8 size macroblocks. In the python coding part, 4D matrix was created after divide the frame into macroblocks. Thereafter, handle a 4D matrix in the code.

Step 3: Motion vector calculation

In video compression, motion vectors represent motion estimation and compensation data. They define how image blocks, or macroblocks, move between frames. There have several methods to find the motion vectors. Hierarchical search, sequential search and, Logarithmic search some of them. Motion vectors are used to create predicted image.

Step 4: Find the residual images for P frames.

Residual Image = Reference Frame - Predicted Image

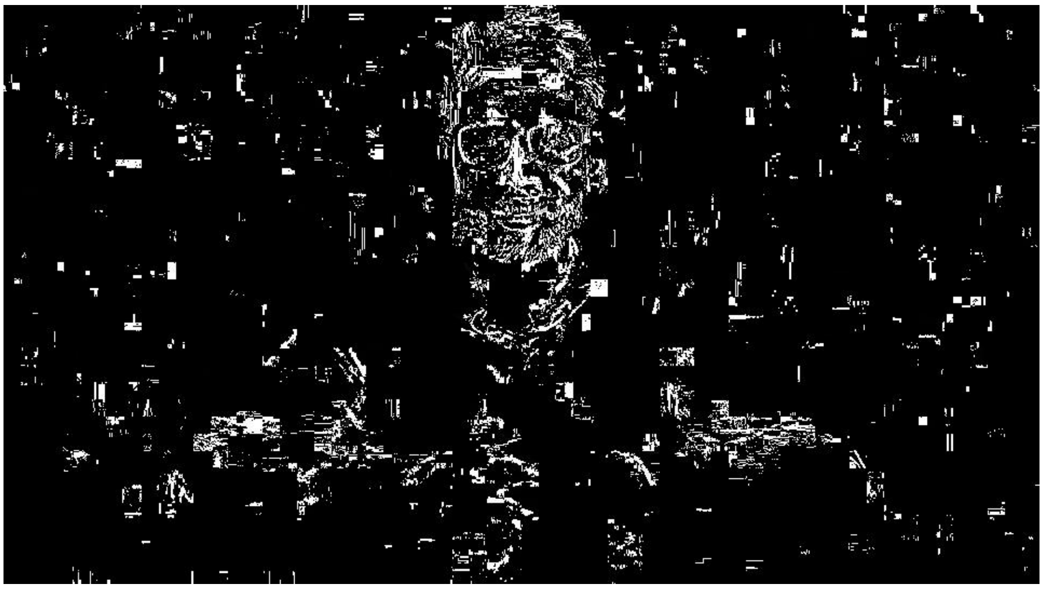


Figure 14: Residual image of 2nd frame (P frame)

Step 5: Transform coding and Quantization

After applying discrete cosine transform for each macroblock, quantize each macroblock using the quantization matrix. The quantization done according to the quantization quality. (Low, Medium and High quality or custom bitrate.) Below 15 and 16 figures was compressed using in high quality.

Step 6: Entropy encoding

In entropy coding, used "Haffman" coding to compress the I-frames and, residual frames. There used the same image compression technique which was used in stage 1, to compress each I frame and residual frames.

Step 7: Decoding

There used the same decoding techniques which used in stage 2. After decompressing, obtain the residual frames of the P frames and I frames. After that, using these images and motion vectors, reconstruct the P frames.

Reconstruct the P frame = Decoded reference frame + Decoded residual frames + Motion Vectors

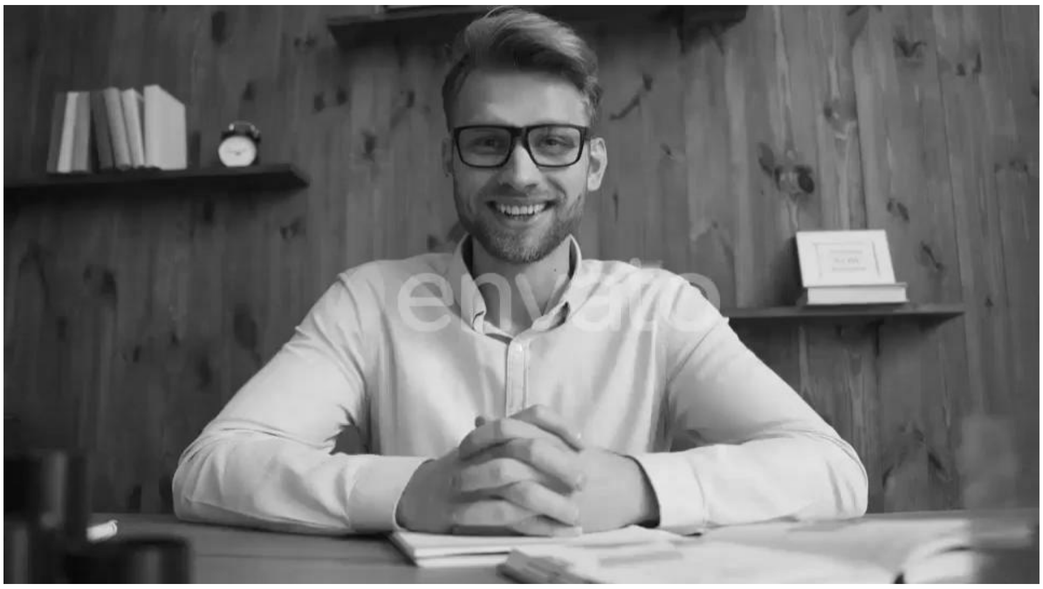


Figure 15: Decode and Reconstructed 1st frame (I frame)

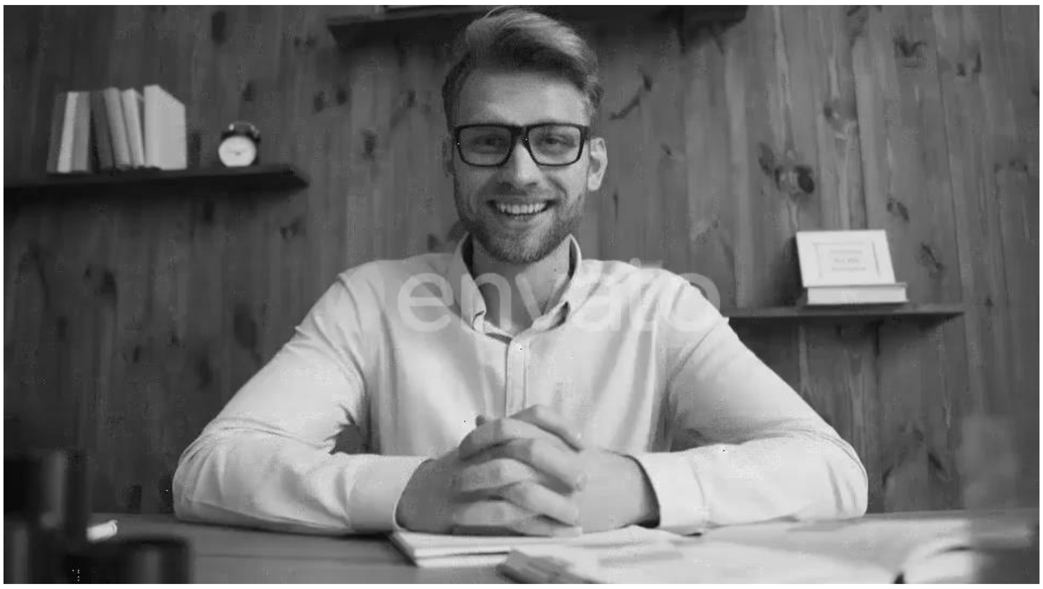
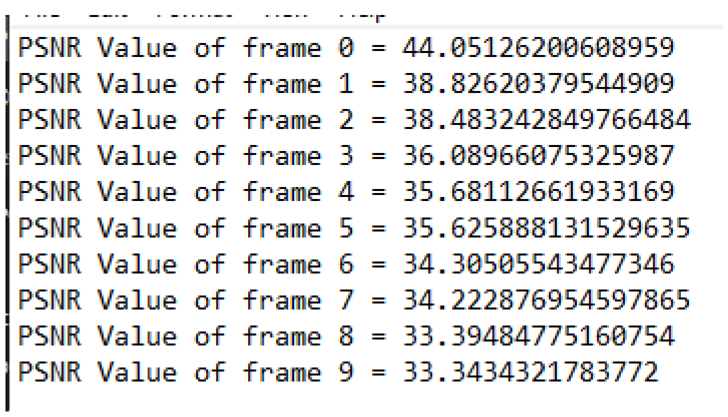


Figure 16: Decode and Reconstructed 2nd frame (P frame)

According to the result in figure 17, I-frames has high PSNR values. Also, PSNR values are reduced sequencially from 1st P frame to 9th P frame for each 10 frame set.



```
PSNR Value of frame 0 = 44.05126200608959
PSNR Value of frame 1 = 38.82620379544909
PSNR Value of frame 2 = 38.483242849766484
PSNR Value of frame 3 = 36.08966075325987
PSNR Value of frame 4 = 35.68112661933169
PSNR Value of frame 5 = 35.625888131529635
PSNR Value of frame 6 = 34.30505543477346
PSNR Value of frame 7 = 34.222876954597865
PSNR Value of frame 8 = 33.39484775160754
PSNR Value of frame 9 = 33.3434321783772
```

Figure 17: PSNR values of first 10 frames.

Step 8: Recreate the video

Finally, recreated the video using decoded images.

3.3) Stage 3: Improved hybrid video codec

The distortion changes with frame transmission rate and is inversely related to reconstructed matrix quality. The PSNR (Peak Signal-to-Noise Ratio) is an image quality metric. As a result, distortion can be viewed as the inverse of PSNR value. To calculate the PSNR value and reconstructed image size, multiply the JPEG quantization matrix by various scaling factors.

Python code was implemented to where output of the video compression can adopt to any given bitrate

REFERENCES

1) GeeksforGeeks, “Huffman Coding | Greedy Algo-3,” GeeksforGeeks, Nov. 03, 2012. https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

2) K. Guru, “Motion estimation,” www.academia.edu, Accessed: Jan. 24, 2024. [Online]. Available: https://www.academia.edu/7361697/Motion_estimation

3) “Compression ratio estimates,” www.ibm.com, Jul. 14, 2023. https://www.ibm.com/docs/en/informix-servers/12.10?topic=compression-ratio-estimates.