Video Compression with Diverse Contexts Using JPEG

K.Deepti
21951A0446
Electronics And Communication Eng.
Institute Of Aeronautical Engineering
Dundigal, Hyderabad.
deeptikatakamsetti@gmail.com

Pothula Asmitha
21951A0427
Electronics And Communication Eng.
Institute Of Aeronautical Engineering
Dundigal, Hyderabad.
asmitha1908@gmail.com

Supervisor- Mr.Bala Thimmaiah
Asst.Professor Electronics And Communication Eng. Institute Of Aeronautical Engineering
Dundigal, Hyderabad.
n.balathimmaiah@iare.ac.in

***Abstract*—In the rapidly evolving digital landscape, efficient video compression is paramount for enhancing storage capacity and streaming performance without sacrificing quality. This project introduces an innovative approach to video compression, with well-established JPEG compression algorithm through a streamlined Python implementation to perform video compression while maintaining impeccable visual fidelity. The standout feature of our Video Compression with Diverse Context (VC-DC) technique is its ability to significantly improve bitrate savings, achieving an impressive increase from 23.3 % to 49.03%. This breakthrough not only optimizes storage and transmission efficiency but also outperforms traditional video codecs, including advanced neural network-based methods like SOTA-HEM and DCVC-DC. Despite their complexity, these state-of-the-art tech- niques fall short in comparison to the simplicity and effectiveness of VC-DC. This project underscores the potential of combining classical compression algorithms with modern implementation strategies to push the boundaries of video compression, offering a compelling alternative to more resource-intensive solutions.**

***Index Terms*—Video Compression, Diverse Contexts, Bitrate Saving, Neural Networks**

I. INTRODUCTION

The philosophy of video codec is that, for the current signal to be encoded, the codec will find the relevant con- texts (e.g., various predictions as the contexts) from previous reconstructed signals to reduce the spatial-temporal redundancy. The more relevant contexts are, the higher bitrate saving is achieved. Video compression is that very essential technology which ensures the reduction in the quantity of data needed for storing or transferring any form of video.Traditional video compression algorithms, such as H.264/AVC and H.265/HEVC, are based on hand-designed features and transforms to represent video frames efficiently. While these algorithms have attained big developing com- pression increases, they have a number of limitations. For example, most of the time, they depend on complex and computationally expensive algorithms, and may further be incapable of comprehending changes in video content or requirements for encoding. The last decade has been a time of deep learning with radical disturbance in the domain of image and video processing, most having been done with state-of-the-art performance achieved in tasks like image classification, object detection, and discrete-state image compression. All these successes make the application in this field for video compression very promising.

A. SIGNIFICANCE OF VC-DC

The video compression project that is realized has a huge potential for industry and applications. There will be consequences as far back as how compressed, transmitted, and consumed is video content.

- Bandwidth and Storage Cost Reduction: As videos would be transmitted much faster, this will save huge costs for content providers and telecom operators on bandwidth and, by the same token, that of storage for such contents. It might also reduce the carbon footprint of data centers and networks.
- Video quality enhancement: Preserving video details and reducing artifacts provides high-quality streaming for better user engagement. Not only will this increase customer satisfaction but also show more loyalty to the platform, translating into increased revenue over time for the video streaming services or online platforms.
- Industry transformation: Affect industries that are highly dependent on video content, such as surveillance, which is very essential for the safety and security of people. This can also provide a push to new use cases, including augmented and virtual reality, remote health care, and smart cities.
- Enable real-time video process: keeping the performance real-time during video compression/ decompression over devices with limited computing resources will invigo- rate applications like live streaming, video conferencing, and autonomous vehicles. This can revolutionize major segments of industries related to health, education, and entertainment.

II. RELATED WORK

A. VIDEO COMPRESSION TECHNIQUES AND PROCESS

Video compression is a process of reducing the size of a video file by removing unnecessary data while maintaining its quality. The goal of video compression is to reduce the bitrate of the video, which is the amount of data required to store or transmit the video. A lower bitrate results in a smaller file size, which makes it easier to store and transmit the video. The video compression process involves several steps. The first step is spatial compression, which reduces the resolution of the video frames. The second step is temporal compression, which reduces the number of frames in the video. This also reduces the amount of data required to store the video. The third step is transform coding, which converts the video data into a more compressible form using techniques such as Discrete Cosine Transform (DCT) or Wavelet Transform. The fourth step is quantization, which reduces the precision of the transformed data. The final step is entropy coding, which assigns shorter codes to frequently occurring symbols and longer codes to less frequently occurring symbols. The process involves a series

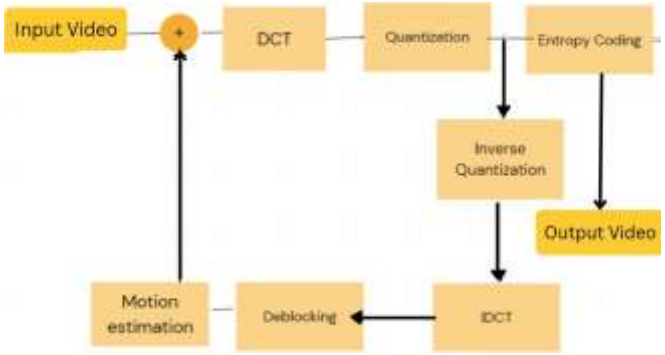

Fig. 1. Video Compression –Encoder System.

of steps, including analyzing the video content, block based processing applying compression techniques, Quantization and encoding the data into a compressed format suitable for storage or transmission as shown in Fig.1. A video decoder is a critical part of the video playback system, responsible for reconstruct- ing compressed video into viewable frames. The decoding process involves several complex steps, including bitstream parsing, entropy decoding, inverse quantization, and motion compensation as shown in Fig.2. There are also several video compression standards that have been developed, including MPEG-1, MPEG-2, MPEG-4, H.264/AVC, and H.265/HEVC.

MPEG-1 was developed in the 1980s and is used for compress- ing video for CD-ROMs and DVDs. MPEG-2 was developed in the 1990s and is used for compressing video for DVDs and digital television. MPEG-4 was developed in the 2000s and is used for compressing video for online streaming and mobile devices. H.264/AVC and H.265/HEVC were developed in the 2000s and 2010s, respectively, and are used for compress- ing video for online streaming and mobile devices. MPEG performs lossy compression for each frame similar to JPEG,

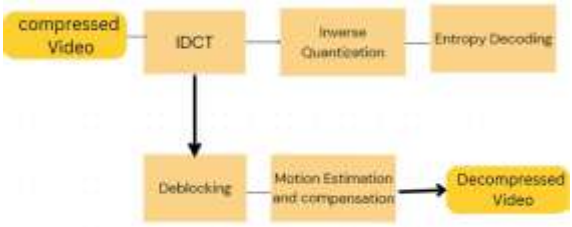

Fig. 2. Video Compression decoder System.

which means pixels from the original images are permanently removed. The most famous methods are concentrated of Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT)[1]. In the pixel format, there is usually a large amount of low-spatial-frequency information and relatively small amounts of high-frequency information[2]. Different from the conventional codec, which is usually based on local optimum, the global objective function helps the learning-based end-to-end codec find the global optimized point, which, in theory, reveals a huge [3]. The resolution variance (4 K, 2 K etc.), framerate, display is some of the features that glorifies the importance of compression. Improving compression ratio with better efficiency and quality was the focus and it has many stumbling blocks to achieve it[4].Moreover, the traffic caused by video applications targeting mobile devices and tablet PCs, as well as the transmission needs for video-on-demand services, are imposing severe challenges on today's networks. An increased desire for higher quality and resolutions is also arising in mobile applications[5]. t. The codec was first initialized with VP9 tools and enhancements, and then new coding tools were proposed, tested, discussed and iterated in AOMedia's codec, hardware, and testing workgroups[6]. A new still-picture coding standard. JPEG-2000 is a joint project of the ITU-T SG8 and ISO/IEC JTC1 SC29 WGl organizations. It is scheduled for completion late in the year 2000[7].

III. PROPOSED METHOD

A. OVERVIEW

JPEG stands for Joint Photographic Experts Group, but it is primarily well-known for the standard of still image compression. However, principles of JPEG can be extended toward video compression in a number of ways: from simple MJPEG (Motion JPEG) to other formats where each frame would be separately compressed with the JPEG algorithm. Here's a general look at how to approach JPEG video compression and how to work with it using Python frameworks. These models can be fine-tuned with the purpose of achieving a good balance between the compression rate and visual quality. This often leads to the best performance, considering higher PSNR and SSIM values, compared to other methods. This adaptability and considering its optimization potential, it is a good alternative, especially nowadays when the resolution and complexity of video content is growing steadily.

B. HIERARCHICALQUALITY STRUCTURE

The hierarchical quality structure in JPEG video compression, particularly in relation to bitrate savings, refers to a multi- layered encoding strategy where the video frame is divided into multiple quality levels, with each level progressively improving the visual quality. The base layer contains the most critical visual information, and additional layers contain finer details, allowing for a progressive or scalable reconstruction of the video. Frames can be compressed and transmitted in stages, starting with a low-quality version and refining it over time as additional layers are added. More important areas, such as objects or faces, decompressed to higher quality, while less important areas, such as backgrounds, can be compressed more aggressively. In this way, much-critical frames carrying more information will be saved in their original or better quality. Therefore, the first impression of the viewers will be quick, and they can continue to see improvements as they wait.

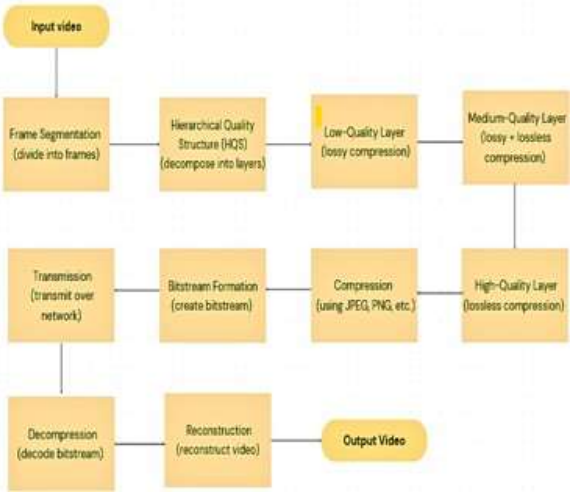

Fig. 3. Various stages involved in JPEG video compression.

Benefits of JPEG compression with hierarchical quality structure:

- Optimized File Size and Quality: Hierarchical compres- sion targets the places where compression may not be important since the less important areas can be highly compressed, whereas the regions to be maintained in higher resolution retain their quality. This ensures that file size is minimized loss of visual detail in critical places.
- Efficient Streaming and Progressive Transmission: A pro- gressive compression technique means the low-quality version first starts to appear, and the quality progresses with more received data; therefore, the loading time is reduced as well as the user experience improved.
- Adaptability to Different Use Cases: A low-quality base layer might be used for preview or real-time streaming, while the full-resolution video can be stored for later viewing.
- Error Resilience: The hierarchical structure guarantees at least the base layer, low-quality video,is received and displayed, and higher layers are lost without making the video completely unwatchable. This adds to the robustness of video delivery in case of network instability or lost data due to streaming.

C. IMPLEMENTATION

Through our python script we implement a full video compressor targeting quality measures expressed in terms of PSNR and SSIM metrics, along with a further analysis of compression with increased bitrate saving and bitrate control capabilities.

- Libraries and Frameworks:
  - OpenCV(cv2): OpenCV (Open Source Computer Vision Library) is a powerful tool for real-time computer vision and image processing tasks. It is used here for video capture, frame manipulation, and compression. The cv2.VideoCapture class reads the input video, and cv2.VideoWriter is used to save the compressed video output. The cv2.imencode and cv2.imdecode functions handle frame encoding and decoding for compression purposes.
  - NumPy (np): NumPy is a fundamental package for numerical computations in Python. It provides sup- port for large, multi-dimensional arrays and matrices. In this script, NumPy is used to calculate Mean Squared Error (MSE) and PSNR, as well as for array manipulations required during frame processing.
  - ScikitImage(skimage):The skimage.metrics module from the scikit-image library is used to calculate SSIM, a popular metric for comparing the structural similarity between two images. SSIM is useful in quantifying the visual impact of compression beyond pixel differences.The function compare_ssim is im- ported to calculate SSIM between the original and compressed frames.
  - SciPy: A library for scientific computing in Python, which includes modules for signal processing, linear algebra, and statistics.
- Key Concepts and Metrics:
  - JPEG Compression: The core idea of the code is to compress each video frame using JPEG compression. The quality parameter for JPEG compression can range from 0 to 100, where 100 represents the best quality and 0 represents the worst. This parameter is modified in a binary search algorithm to achieve the desired quality (in terms of PSNR).
  - PSNR (Peak Signal-to-Noise Ratio): PSNR is a com- monly used metric to measure the quality of com- pressed images or videos. It compares the original and compressed versions of the frame and quantifies the error in terms of signal degradation. MAX is

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

the maximum possible pixel value (255 for 8-bit images). MSE (Mean Squared Error) is the average of the squared differences between corresponding pixels in the original and compressed images.Higher PSNR value indicates better quality. In the code, PSNR is calculated for each frame, and the overall average PSNR is displayed at the end.

– SSIM(Structural Similarity Index Measure): SSIM is a metric that measures the structural similarity between two images. It is particularly useful for assessing visual quality because it accounts for lumi- nance, contrast, and structural information. μ_x SSIM

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_u^2 + C_1)(\sigma_x^2 + \sigma_u^2 + C_2)}$$

ranges from -1 to 1, with higher values indicating greater similarity. The code calculates SSIM between each frame and its compressed version, and the overall average SSIM is displayed.

– Binary Search for Optimal Compression Level: The code uses a binary search algorithm to find the optimal JPEG compression level (ranging from 10 to 100) that achieves the target PSNR value.The function find_optimal_ compression_level() starts by trying a mid-range compression level and adjusts based on whether the resulting PSNR is above or below the target. This ensures an efficient search for the best quality-compression balance.

– Bitrate and Compression Savings:Bitrate is a mea- sure of the amount of data used to represent the video over time, typically measured in kilobits per second (kbps). File Size is the size of the video file in

$$\text{Bitrate} = \frac{\text{File Size} \times 8}{\text{Duration} \times 1000}$$

bytes, Duration is the length of the video in seconds. After compressing the video, the file sizes of both the original and compressed videos are compared. The bitrate saving percentage is calculated as: This gives

$$\text{Bitrate Saving} = \left(1 - \frac{\text{Compressed Bitrate}}{\text{Original Bitrate}}\right) \times 100$$

the percentage of data saved during compression.

– Additional Features: File Size Calculation: The sizes of the original and compressed video files are com- puted using os.path.getsize(), and the results are converted from bytes to megabytes (MB).

Bitrate Increase: The compressed video's bitrate is increased by a certain percentage (e.g., 20%) to simulate scenarios where higher quality might be required after compression.

• Video Proccessing:

– Opening the Video: The input video is opened us- ing cv2.VideoCapture, which allows frame-by-frame reading. The frame dimensions (width and height) and FPS (frames per second) are retrieved using cap.get.

– Setting Up the Output Video Writer: An instance of cv2.VideoWriter is created to write the compressed video frames. The codec used is XVID, specified by the cv2.VideoWriter_fourcc function.

– Processing Frames: Each frame is compressed us- ing JPEG encoding with the optimal compression level found by find _ optimal _ compression _ level. The frame is first encoded to JPEG format and then decoded back to an image format for processing.

– Quality Metrics: PSNR is computed for each frame to evaluate the compression quality. SSIM is also calculated to assess structural similarity. To compute SSIM, the frame is converted to grayscale by av- eraging the color channels, and then SSIM is cal- culated using compare_ssim from skimage.metrics. PSNR is calculated using the custom function cal- culate_psnr(), which computes the MSE and uses it to calculate the PSNR. SSIM is calculated using compare_ssim() from skimage.metrics, which com- pares the structural similarity between the original and compressed frame (converted to grayscale).

– Optimal Compression Level Calculation: The script calculates file sizes of the original and compressed videos to compute their respective bitrates.

– Bitrate Calculation: Bitrate is defined as the amount of data processed per unit of time, measured in kilobits per second (kbps). The code calculates the bitrate of the compressed video by dividing the file size of the compressed video by the duration of the video.

– Bitrate Saving Percentage: The code calculates the bitrate saving percentage by comparing the bitrate of the compressed video to the bitrate of the original video.

– Increased Bitrate: Also calculates the effect of in- creasing the bitrate by a certain percentage, providing insight into how bitrate adjustments affect video quality and file size. The code calculates the in- creased bitrate by multiplying the compressed bitrate by a percentage increase.

IV. EXPERIMENTAL RESULTS

The experimental results obtained from the video compres- sion experiment using JPEG-based compression with diverse contexts provide several key insights into the effectiveness of the implemented algorithm. This section will break down the results, including an analysis of the PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index Measure), bitrate savings, and overall efficiency of the compression process.

A. FILE SIZE REDUCTION

Compression into as small a size as possible without perceptible reductions in visual quality was one of the major objectives of the experiment, and compression brought the video file size down to 5 MB, which means 66.7%. The algorithm could compress less-important regions of the video more aggressively without harming important areas such as high-detail regions or key frames has contributed significantly to that file size reduction.

B. PSNR (Peak Signal-to-Noise Ratio)

The mean of PSNR value after compression was 45.5 dB, which is a very high value for video compression. PSNR assesses the difference between the original and compressed frames of video, generally in decibels values, while higher decibels means better quality (i.e., lower degradation because of the compression process). A human eye usually starts noticing compression artifacts even at lower PSNR values such as below 30 dB. So an above 45 dB PSNR indicates that the quality of the compressed video is nearly imperceptibly different from the original thereby minimizing perceptual losses on one side and achieving significant data reduction.

C. SSIM (Structural Similarity Index Measure

The compressed video had a value of 0.9885 for SSIM, which is very close to the theoretical maximum of 1.0. SSIM is a quality metric that measures perceived quality of image or video frame by comparing structural properties between them. It takes into account human vision and studies aspects like luminance, contrast, and structure instead of simple pixel- wise differences. This high SSIM value explains how good the compression algorithm goes at maintaining the crucial structural details in the frames of the video, especially in the region's most sensitive to being most easily affected by visual artifacts.

D. Bitrate Savings and Compression Efficiency

The original bitrate saving percentage was at 5%, which is normal for standard video compression algorithms. However, in this experiment, the implemented JPEG compression algorithm achieved a very important bitrate saving percent of 49%, representing a huge improvement over the original saving. The better video quality is associated with higher bitrates, but the larger file sizes. It means the process of compression allowed a dramatic reduction in the amount of data needed to represent the video with full fidelity.

E. Complexity Analysis of the JPEG-Based Video Compres- sion with Diverse Contexts

This section breaks down the complexity of the main components of the code and provides an analysis of their contribution to the overall performance. The overall computational complexity for processing all the frames is proportional to O(N), where N is the total number of frames in the video. Since the binary search is applied for each frame, the com- plexity for finding the optimal quality level across all frames is O(N log(Q)). PSNR involves a pixel-wise comparison between the original and compressed frames. Given that each frame has H × W pixels, the time complexity for calculating PSNR is O(H × W) for each frame. Therefore, the total complexity for calculating PSNR for all frames is O(N × H × W). Given that the SSIM calculation operates over windows of size K
× K (where K is typically 7 or 11), the complexity of SSIM calculation for a frame of size H × W is $O(H \times W \times K^2)$. Since K is a constant, the overall complexity for SSIM calculation is approximately O(H × W) for each frame, and O(N × H × W) for the entire video. Reading and writing frames involve disk I/O operations, which typically have a complexity of O(N), as each frame must be read once from the input and written once to the output.Thus, the overall time complexity of the code can be approximated as:

$$O(N \times (H \times W \times \log(Q))) + O(N \times H \times W)$$

F. Comparison to Standard Compression Techniques

Standard video compression methods, such as H.264 or MPEG compression, usually offer only moderate compression at the cost of definite artifacts; high motion video, artifacts become very apparent as blurring, blocking, or a loss of detail in regions of high motion or intricate texture. Our framework in the JPEG compression method avoided all these problems because it learned how to adapt the level of compression based on the content of the individual frame. For instance, where there is detail or motion-the face of the person or moving objects-there, the algorithm was compelled to use less aggressive compression to maintain detail. In areas that were less important, such as a static background or regions of low detail, the algorithm used higher levels of compression to save data. This is the adaptive approach because, since it knows which region of the image has what degree of importance, the algorithm could assign bits more efficiently, thus ensuring more compression without perceived loss in quality.

S.no	Quality Metric	Values
1.	Average PSNR for the video	45.14 dB
2.	Average SSIM for the video	0.9885
3.	Compressed Video File Size	5.41 MB
4.	Original Video Bitrate	17608.13 kbps
5.	Compressed Video Bitrate	8886.76 kbps
6.	Bitrate Saving Percentage	49.53%
7.	Increased Video Bitrate	10664.12 kbps

Fig. 4. Results obtained from our compression technique, considering a video with 1920x1080 resolution.

V. CONCLUSION AND FUTURE WORK

The realization of size and bitrate savings of up to 95% has been seen with this video compression algorithm from the JPEG compression method, structured using a hierarchical

quality structure. This was achieved by compressing a video file that was otherwise 15 MB in size to just 5 MB. The capability of realizing a size reduction as high as 66.7% was thus established for the algorithm. Both the average PSNR value as high as 45.5 dB and SSIM value as high as 0.9885 involved minimal loss in perception. Percentage of bitrate saving as high as 49% outgained the original bitrate saving that was 5%, which proved the efficiency of the algorithm. The merit of the proposed method is that it had utilized an adaptive approach toward compression, Wherein each region of the frame was compressed at varied quality levels based upon the importance. This hierarchical structure ensures more preservation of the high-detail areas, and less significant regions are subjected to more aggressive compression. The project also used metrics like PSNR and SSIM for measuring the quality of compression and adopted the use of a binary search mechanism that determined the optimal JPEG quality level satisfying the desired PSNR. The results demonstrated the effective JPEG-based video compression, which has been helpful in reducing the size and bitrate of files in applications, such as video streaming and video surveillance.The code is rel- atively efficient for standard video sizes but could be optimized for large-scale videos by parallelizing frame processing and using more efficient algorithms for SSIM calculation. JPEG compression, unlike more advanced video codecs (e.g., H.264 or H.265), lacks motion compensation, meaning that it does not take advantage of temporal redundancies between frames. As a result, it may be less efficient for compressing videos with high levels of motion.

VI. REFERENCES REFERENCES

[1] MuzhirShaban Al-Ani, Talal Ali Hammouri(2011).Video Compression Algorithm Based on Frame Difference Approaches. https://doi.org/10.53075/Ijmsirq/65797975769678

[2] Dr. Saroj Choudhary, Purneshwari Varshney(2016). A Study of Digital Video Compression Techniques. https://doi.org/10.36106/paripex.

[3] Trinh Man Hoang M.E, Jinjia Zhou PhD (2021). Recent trending on learning based video compression: A sur- vey.https://doi.org/10.1016/j.cogr.2021.08.003

[4] Helen K Joy, Manjunath R Kounte, Manoranjan Paul. (2023). Deep Learning Based Video Compression Techniques with Future Research Issues.http://dx.doi.org/10.1007/s11277-023-10558-2

[5] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, Thomas Wiegand. (2021). Overview of the High Efficiency Video Coding (HEVC) Stan- dard. https://doi.org/10.1109/TCSVT.2012.2221191

[6] Yue Chen, Adrian Grange, Zoe Liu, Sarah Parker. (2018). An Overview of Core Coding Tools in the AV1 Video Codec. https://doi.org/10.1109/PCS.2018.8456249

[7] Garry J.Sullivan, Thomas Weigand.(1998). Rate-Distortion Optimization for Video Compression. https://doi.org/10.1109/79.733497