12-2022

Sequential Frame-Interpolation and DCT-based Video Compression Framework

Yeganeh Jalalpour
Portland State University, Yeganeh@pdx.edu

Wu-chi Feng
Portland State University, wuchi@pdx.edu

Feng Liu
Portland State University, fliu@pdx.edu

Citation Details

Jalalpour, Y., Feng, W. C., & Liu, F. (2022, December). Sequential Frame-Interpolation and DCT-based Video Compression Framework. In Proceedings of the 4th ACM International Conference on Multimedia in Asia (pp. 1-7).

Sequential Frame-Interpolation and DCT-based Video Compression Framework

Yeganeh Jalalpour, Wu-chi Feng, Feng Liu
{yeganeh,wuchi,fliu}@pdx.edu
Portland State University
Portland, Oregon, USA

ABSTRACT

Video data is ubiquitous; capturing, transferring, and storing even compressed video data is challenging because it requires substantial resources. With the large amount of video traffic being transmitted on the internet, any improvement in compressing such data, even small, can drastically impact resource consumption. In this paper, we present a hybrid video compression framework that unites the advantages of both DCT-based and interpolation-based video compression methods in a single framework. We show that our work can deliver the same visual quality or, in some cases, improve visual quality while reducing the bandwidth by 10-20%.

CCS CONCEPTS

• **Computing methodologies → Image compression**; • **Computing methodologies → Neural networks**.

KEYWORDS

video compression, learning-based video frame interpolation

ACM Reference Format:
Yeganeh Jalalpour, Wu-chi Feng, Feng Liu. 2022. Sequential Frame-Interpolation and DCT-based Video Compression Framework. In *ACM Multimedia Asia (MMAsia '22), December 13–16, 2022, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3551626.3564959

1 INTRODUCTION

More video data than ever is being generated and transferred for applications such as online learning, video conferencing, social media, video broadcasting, TV streaming networks, smartphone photo or videography, (cloud) video games, telesurgery, and medical imaging [1, 2, 6, 7, 10, 14, 15, 17]. While current video compression standards have greatly reduced the amount of data to represent video streams, they still require significant resources. Cisco predicted video data would consume about 82% of the internet bandwidth [11]. Thus, even small continued improvements in video compression can have a significant impact on internet traffic. For example, an improvement of 2% can reduce internet traffic by 11 Tbps.

The discrete cosine transform (DCT) has served as the basis of nearly all video compression standards for the last three decades. DCT-based video compression techniques explicitly encode data through predictive coding and differential residual coding. This allows for great flexibility in the encoder to trade off video quality and bitrate for a particular application. Currently, H.264 is the most widely deployed and used video codec and is now embeded on many devices; H.264 has lower latency and consumes less computational power than H.265 and VP9 [5, 9, 24].

With recent advances in GPU technology, the use of deep neural networks to aid in a large number of applications is becoming feasible. Video frame interpolation (VFI) is one such example where trained VFI models can be used to interpolate one or more frames between existing frames. If applied to video compression, one could imagine using VFI in compression to entirely skip coding individual (or multiple) frames and allowing the decoder side to interpolate and *synthesize* the missing frames. We will call such a scenario *generative video compression (GC).* The GC method can help us reduce bandwidth consumption by not explicitly encoding all the data as in traditional video compression algorithms.

In theory, while VFI can significantly reduce the amount of data required to represent a video stream, there are some limitations that impact their efficacy in general video compression. First, frame interpolation is extremely difficult for frames that exhibit non-linear motion (e.g., acceleration, object spinning). VFI would need to be able to predict the form of motion as well as the trajectory to be more accurate. Second, frame interpolation has typically been applied to increase the frame rate of an existing sequence, such as going from 30 frames per second to 60 frames per second to make the motion slightly smoother, and not to remove frames. Synthesizing high-quality intermediate frames using VFI in more temporally-distant input frames becomes problematic due to increased motion and occluded objects. One such solution to helping this is to find a way to combine VFI with techniques that can explicitly represent data.

In this paper, we present an efficient video compression framework that combines the advantages of DCT-based compression and video frame interpolation into a single framework. In particular, for areas in the frame that VFI can easily interpolate, we skip encoding these areas altogether, and for areas that VFI cannot easily interpolate, we explicitly use DCT-based encoding. Our results show that by using our framework, we can reduce the bandwidth of the video stream by 10-20% depending on the scene type and maintain similar video quality.

2 BACKGROUND AND RELATED WORK

DCT-based Video Compression. Traditional video compression standards that use DCTs have existed for many years. DCT-based video compression algorithms typically use 16x16 pixel *macroblocks.* The compression standards then use *motion-compensation* to find

MMAsia '22, December 13–16, 2022, Tokyo, Japan

ACM ISBN 978-1-4503-9478-9/22/12.
https://doi.org/10.1145/3551626.3564959

the closest match that can serve as a prediction for the macroblock and *residual coding* to provide any corrections to the prediction. To provide random access to frames, sets of video frames are typically grouped into a *group of pictures (GOPs)* and the size of each GOP is called the *Gop size*. Each iteration of the various H.26x and MPEG-x standards has typically added more options for motion-compensation and improved entropy encoding but are similar at a high level. Video codecs in use today include H.264 [27], H.265 [25], and H.266 [4]. We assume readers are familiar with MPEG video compression techniques and refer readers to overview papers for more information [25, 27].

Video Frame Interpolation. Video frame interpolation (VFI) is a computer-vision technique to improve the frame rate of existing video streams. Traditional frame synthesis approaches estimate dense motion between two consecutive input frames using optical flow or motion estimation algorithms and then leverage the estimated dense correspondences to temporally synthesize one or more frames between inputs [3, 29]. Inspired by the success of deep neural networks for computer vision applications, deep learning-based frame interpolation approaches [16, 19, 21–23, 28] have been proposed to perform image synthesis tasks. Niklaus *et al.* developed a deep convolutional neural network for frame interpolation that merges motion estimation and image synthesis into a single convolution process and trains the deep network to learn to estimate spatially-varying convolution kernels for frame synthesis [23]. Liu *et al.* developed a voxel flow method that employs a deep neural network to estimate optical flows from the target frame to the input ones to synthesize the target frame [16]. The PhaseNet work investigates the use of deep neural networks to perform frame interpolation in the frequency domain [19]. The task-oriented flow (TOFlow) research showed that optical flows can be fine-tuned to enable better video frame interpolation [28]. Jiang *et al.* developed a method that allows interpolating multiple frames between two inputs [13]. Niklaus *et al.* developed a context-aware neural network that explores the contextual information to guide frame synthesis [21], and their recent work further explores feature pyramids to help with frame synthesis [22].

For consistency of discussion, the VFI methods use two *anchor frames* for which it then interpolates middle/intermediate frame(s). We will assume a fixed distance (in frames) between the anchor frames, which we refer to as the *anchor frame distance (AFD).*

Hybrid compression. The FID video compression framework [12] introduces preliminary work of combining a DCT-based standard and a VFI method. The FID video compression framework only considers I-frames as anchor frames. Encoded B-frames contain less data and consequently produce smaller bitstreams than having only I-frames in the sequence. This paper explores the generalizing of the encoded sequence framework as well as a more thorough treatment of experimentation.

3 APPROACH

In this paper, we create an efficient hybrid video compression framework that takes advantage of DCT-based video compression and frame interpolation-based methods. DCT-based video compression standards, when compared to video interpolation-based methods, can keep necessary details when there are challenging regions in the presence of high-frequency data, large or non-linear motions, and fast changes in the scene. However, encoding all this data consumes bandwidth while trying to keep the video quality high. Current interpolation-based methods can synthesize regions with low-frequency data, linear, or small motions. Therefore, interpolation-based video compression can help save bandwidth and maintain high visual quality. However, interpolation methods can perform poorly when the scene is hard to synthesize, such as with non-linear motion.

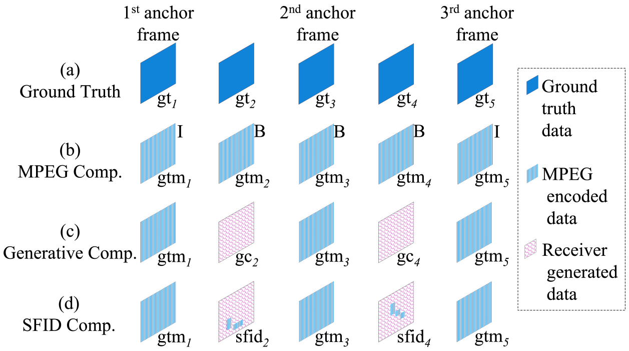

Figure 1: An overview of SFID. SFID encodes anchor frame sequences using DCT-based compression (shown in blue) and attempts to skip intermediate frames by recreating them by generative compression. For areas (macroblocks) that are not synthesized with high enough quality, SFID explicitly encodes the macroblock into the stream.

In Figure 1, we present a general overview of our sequential frame-interpolation and DCT-based (SFID) video compression framework. The first five ground truth frames, gt_i, where $1 \le i \le 5$ are represented in (a). In (b), We use any DCT-based codec, aka MPEG, to encode the ground truth frames, demonstrated by gtm_i where $1 \le i \le 5$, in an open-GOP setting, with a GOP size of 4, and only B intermediate frames.

We select the anchor frames from MPEG-encoded frames for the generative compression (GC) frames based on GOP size and anchor frame distance (AFD). In theory, any such encoding is fine; in this paper, however, we encode frames such that GOP Size is evenly divisible by the AFD size. In this paper, we choose 16 for our GOP size since it is the largest even GOP size in our selected codec (H.264) that we can have the GOP structure of I, B, ..., B, I without enforcing P-frames in between. To get uniformly distributed AFDs, we selected AFDs of 2, 4, and 8 for the GOP size of 16.

In Figure 1-(c), the anchor frames selected are gtm_1 (encoded I-frame), gtm_3 (encoded B-frame), and gtm_5 (encoded I-frame). These anchor frames are then fed into the GC model to generate the intermediate frames (gc_2 and gc_4).

To further improve the efficiency of the framework, in this paper, SFID has different criteria for deciding if a macroblock is to be explicitly encoded into the final bitstream compared with the FID [12] framework. In SFID, for each intermediate frame i of gc_i generated by the GC model, we calculate the peak-to-signal-noise

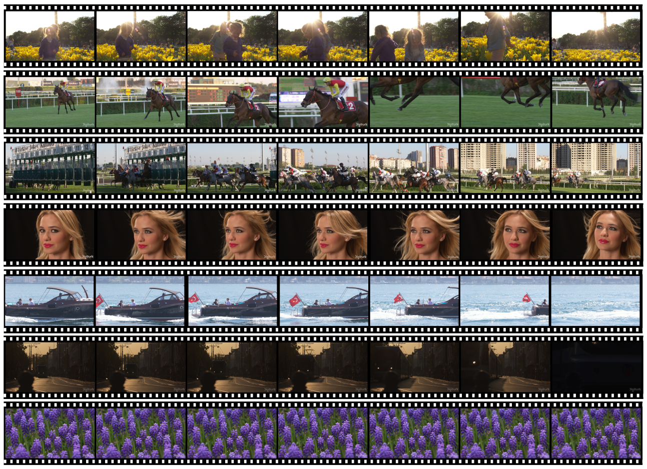

Figure 2: Motion progression from the start to the end of some sequences used in this paper. From top to bottom from Category A sequences we have *FlowerKids, Jockey,* and *ReadySetGo.* From Category B, we have *Beauty* and *Bosphorus.* From Category C and D we show *Twilight* and *Honeybee,* respectively.

(PSNR) of each 16x16 macroblock (*MB*) compared to its corresponding ground-truth MB. Higher PSNR values represent better visual quality. For the intermediate frame i, we decide whether to include data in the bitstream or not by checking the following criteria. If the PSNR value for an *MB* in frame i generated by GC is high enough as shown in Equation (1), or it has better quality than its DCT-based encoded *MB* as shown in Equation (2), we do not explicitly encode that *MB* into the bitstream. Therefore, if

$$PSNR(MB(gc_i), MB((gt_i)) \geq threshold \quad (1)$$

or

$$PSNR(MB(gc_i), MB(gt_i)) \geq PSNR(MB(gtm_i), MB(gt_i)) \quad (2)$$

are true, we do not include the encoded $MB(gt_i)$ to the bitstream. On the receiver/decoder, MBs that were not included in the bitstream will be generated by GC. In this paper, we select PSNR as the metric to perform the visual quality comparison at a macroblock level; however, this comparison can be made by any other visual quality metric.

4 EXPERIMENTATION

4.1 Experimental Setup

Dataset. For evaluation, we apply our video compression framework to sequences from the UVG dataset [18], and Netflix Research dataset [20]. We resize the dimension of all video sequences down to 1920x1072 using the convert function of ImageMagick [8] by the *resize* parameter, so the resolution of all frames is divisible by 16 (the size of an MB).

We use the full UVG dataset video sequences. These sequences have a variety of contrast, texture types, and motion types. These sequences contain 300 to 600 frames per sequence and have frame rates of 50 or 120 frames per second (fps). From the Netflix dataset sequences we use, *Tango* and *Narrator*, contain 294 and 300 frames, respectively, and both have a frame rate of 60 fps.

As one would expect, some sequences are easier to interpolate, such as *Narrator* and *Honeybee*. In *Narrator*, a narrator slowly walks towards the camera with few people in the background walking and contains a blurred less-detailed background. In *Honeybee*, the sequence has a very static background and a few bees flying between flowers. In other sequences, such as *Jockey* and *ReadySetGo*, which are horse racing videos, there is rapid foreground object motion and the camera panning to the left, causing rapid changes in the background. In *Jockey*, the focus of the object changes within the sequence, from focusing on the horse racer to focusing on the legs of the horse. Some representative examples are shown in Figure 2 with frame numbers 1, 100, 200, ..., and 600 displayed.

Implementation details. We use *ffmpeg* (version 4.3.1) as the video compression framework for our experiments [26]. For the video codec, we used libx264.

For the video codec's parameter settings, we used most of the default values except as noted below. For each coding run in the experiments, we use variable-bitrate encoding (a fixed Quantization Parameter, QP) instead of constant-bitrate encoding (a fixed Constant Rate Factor, CRF). Using fixed QP allows better quality comparisons and evaluation between different encodings. If many macroblocks are skipped, using CRF causes the quantization parameter to decrease to "fill" the bitrate rather than keeping the quality the same and letting the bitrate drop for easier-to-encode areas (while maintaining the quality constant). Other parameters that need to be specifically set include anchor frame quality, choice of I, P, and B frames, open-GOP feature, fixed intervals of I-frames, a fixed number of B-frames, etc. We have used the following line when compressing the videos using our chosen DCT-based codec.

```
ffmpeg -y -r 1 -i input_frame%04d.png -qp 23 -c:v libx264
-x264-params b_adapt=0:open_gop=1:keyint=gopSize:
bframes=(gopSize-1):scenecut=0:b_pyramid=0
output_video.mp4
```

We use the above command to encode the anchor frame sequences (including both I and B frames) to pass to the selected generative compression model. We have used the default QP value of 23 to best balance quality and bitrate. We used $gopSize = 16$ to create a maximal consecutive sequence of B-frames without introducing a P-frame and allowing us to have the AFD divisibility requirement mentioned above. We can apply smaller GOP sizes; however, the maximum bitrate reduction is achieved with larger GOP sizes. In order to control the GOP size, we set $keyint = gopSize$. To employ more compression and maintain consistency throughout each sequence. We force all the inter-frames to be B-frames and exclude P-frames by applying $bframes = (gopSize - 1)$ and $b\_adapt = 0$. We have also selected $scenecut = 0$ to reduce the sensitivity of the encoder and avoid aggressive enforcement of I-frames. In this paper, we disabled the hierarchical B-frames coding mode by setting $b\_pyramid = 0$. Therefore, for motion compensation, all B-frames use the surrounding I-frames as their reference frames and not other B-frames in the GOP.

Metrics. For comparisons, we use peak-to-signal-noise (PSNR) and structural similarity (SSIM) index to perform a pixel-wise and perceptual visual quality comparison of constructed frames using our models with the ground truth images. We use the $scikit-image$ package in Python for calculating PSNR and SSIM.

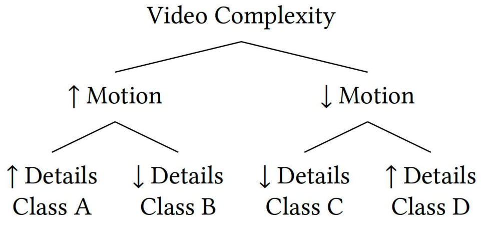

Figure 3: Categories of video complexity.

4.2 A Video Categorization

In order to better understand the advantages and limitations of our proposed framework, we break the video streams into a number of categories. We have found that the performance within each category is reasonably consistent and that the performance between categories can vary. We categorize the video streams into four general complexity categories (A-D) in Figure 3 to analyze the different videos based on their features and behaviors. The videos are categorized based on the combination of criteria of whether there is significant motion (in either the camera or subject) and whether there is a large amount of small detail in the frames.

Category A (High motion, High detail): Sequences in this category exhibit both high motion and fine details. Such sequences are particularly challenging for video interpolation frameworks because the interpolation needs to manage complex motion (e.g., non-linear motion and occluded areas) while keeping details aligned. Because DCT-based video compression algorithms explicitly encode data, they can handle the motion slightly easier but also need to use more data to encode the high-frequency data (details). This category contains *FlowerKids*, *Jockey*, *RaceNight*, *ReadySetGo*, *SunBath*, and *Tango*.

Category B (High motion, Low detail): Sequences in this category exhibit high motion but generally have less detail to be encoded or interpolated. Similar to Category A with respect to motion, these streams can be challenging for video interpolation frameworks. They are, however, slightly easier to interpolate since areas of relatively few details can be more easily created by the interpolation framework. Category B is also relatively easier to compress since there are fewer details to compress. This category contains *Beauty*, *Bosphorus*, *Lips*, *RiverBank*, *ShakeNDry*, and *YachtRide*.

Category C (Low motion, High detail): Sequences in this category exhibit low motion while containing many fine details. These sequences are generally easier for the video interpolation frameworks since the small amount of motion means that the ability to do the point correspondences is easier, and the motion will appear more linear with fewer occlusions. The main issue for the video interpolation frameworks for this category, however, is recreating the fine details throughout the image. These sequences are also relatively easy for DCT-based video compression because there is relatively small motion, and motion compensation can more easily match fine details. Thus, the overall gain from video interpolation relative to DCT-based compression may be more limited. This category contains the *FlowerFocus*, *Narrator*, and *Twilight* sequences.

Category D (Low motion, Low detail): Sequences in this category have both little to no motion and very few fine details. This category is the least challenging for video interpolation as well as DCT-based video compression. This category contains the *CityAlley*, *FlowerPan*, and *Honeybee* sequences.

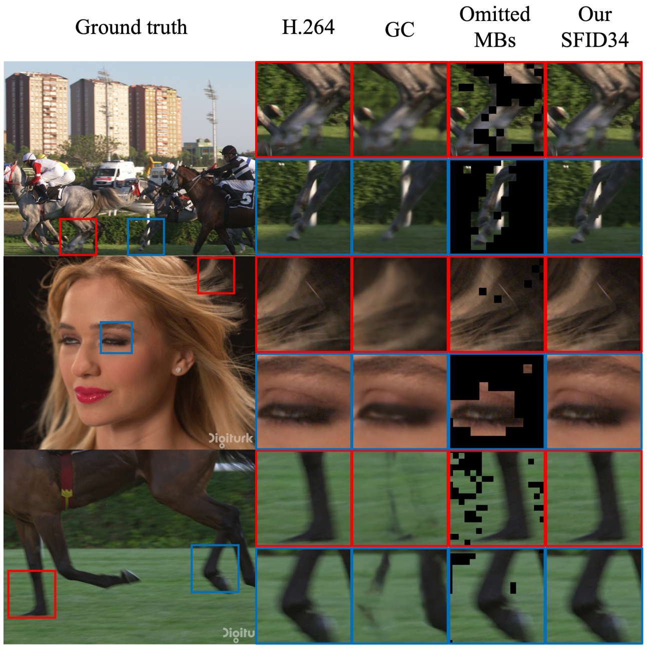

Figure 4: Visual quality comparison for H.264, GC, map of skipped macroblocks, and our framework, SFID, in the ReadySetGo, Beauty, and Jockey sequences.

4.3 Experimental Results

In this section, we present the evaluation of our proposed video compression framework. We first show several examples of purely DCT-based, purely interpolation-based, and our hybrid compression framework. We then describe numerical, category-based results and describe their impact on rate-distortion curves.

4.3.1 Video Examples. We have several representative examples shown in Figure 4. For these examples, the AFD for both SFID and GC is set to 2. For SFID, we set the threshold for macroblock inclusion to 34 dB, *SFID*34. Recall, in SFID, a macroblock is included in a hybrid frame if (i) the encoded SFID macroblock has higher PSNR than the interpolated MB at the same location, and (ii) the interpolated MB at the same location is less than the PSNR threshold value.

In the top images of Figure 4, the *ReadySetGo* sequence is a panning shot with multiple competing race horses with fast, non-linear motion of horses (particularly the legs). Macroblocks were included to fix the hooves in the lower left part of the images. Note the detail in the right front hoof of the horse and the surrounding bushes, as well as the detail in the left back hoof. The "Omitted MB" column shows, in black, the areas that are not encoded and where GC data is used. In the bottom detail, we see that the interpolation framework had great difficulty predicting the location of the horse's front legs and is missing the two white dots in the upper left. These were corrected in the SFID34 result by including the appropriate macroblocks in the hybrid frame. By blending interpolation and explicit macroblock encoding, the SFID34 sequence achieves a 21.2% reduction in bitrate while also improving the average PSNR and SSIM by 0.24 dB and 0.01, respectively, compared to just H.264.

In the middle set of images, *Beauty* is a motion video sequence (Category B) with a mixture of details. The motion is mainly in the hair of the person, which is being affected by a fan blowing. Here, we see that there is hair that goes from the middle top towards the middle right, and the really fine details of the hair are not high enough quality in the GC sequence. These are corrected in the SFID frames. Because of the very small motion in the eye area, the GC frame, even with the detail, requires little correction. By blending interpolation and explicit macroblock encoding, the SFID34 sequence achieves a 13.08% reduction in bitrate while also improving the average PSNR and SSIM by 0.416 dB and 0.010, respectively, when compared with the H.264 standard.

In the bottom set of images, we see where video frame interpolation has difficulty: significant non-linear motion. Here, the video is zoomed in on the legs of the horse running. Since the motion is more radial (i.e., swinging from the joint), the interpolation framework has great difficulty. Here, the advantage of DCT-based video compression becomes clear; with explicit coding of all data, the video quality is expected to improve. Nevertheless, the simpler motion areas (e.g., the grass in the background) still benefit from using the interpolation techniques. By blending interpolation and explicit macroblock encoding, the SFID34 sequence achieves a 12.74% reduction in bitrate while also improving the average PSNR and SSIM by 0.022 dB and 0.007 dB, respectively, when compared to H.264. It is important to note that as video frame interpolation techniques improve, they can easily be incorporated into our framework. This means, in general, that less macroblocks will be encoded into the hybrid stream.

4.3.2 Category by Category Results. We now describe specific experimental results. As previously mentioned, the benefits of SFID vary depending on the type of encoded video sequence. Rather than lumping the results into aggregate numbers across all videos, we do so by category. For the discussion in this section, we have graphed representative rate-quality curves for various macroblock inclusion thresholds and AFDs for the videos in each category in Figure 5. We have also listed the aggregate performance under AFD 2 for all the videos within each category in Table 1.

CATEGORY A: Category A video sequences are the hardest for interpolation frameworks to handle due to the significant motion, particularly where there is non-linear motion. In Figure 5, the rate-distortion curve is for varying the AFD between 2, 4, and 8.

For GC, the graph trails off to the left using AFDs 2, 4, and 8. While using a larger AFD reduces what needs to be compressed in the anchor frame sequence, this comes with a significant impact on image quality. The reason for this is that with the motion in the video, the ability to interpolate accurately gets significantly harder with larger AFDs.

For SFID30, we see AFD 2 results in both the highest average PSNR and lowest bitrate, while AFD 8 is both lower in quality and larger in bitrate. The key to this phenomenon is that in hybrid compression, we are trading off explicitly encoding information into the anchor frame sequence and the performance of the interpolation frameworks without having to provide corrections. For AFD 8, the anchor frames are spaced too far apart, meaning the interpolation results are ineffective, causing significant numbers of macroblocks to be encoded to fix the interpolation results.

Category	Methods	PSNR	SSIM	Bpp
A	H.264	38.451	0.928	0.198
	GC	35.465	0.926	0.118
	SFID34	**38.463**	**0.938**	0.163
B	H.264	36.164	0.866	0.253
	GC	35.764	0.868	0.156
	SFID34	**36.461**	**0.876**	0.204
C	H.264	39.729	0.927	0.065
	GC	39.754	0.935	0.046
	SFID34	**40.069**	**0.937**	0.049
D	H.264	37.734	0.921	0.082
	GC	38.202	0.932	0.073
	SFID34	**38.213**	**0.932**	0.073
Overall	H.264	37.782	0.905	0.177
	GC	36.736	0.908	0.113
	SFID34	**38.021**	**0.914**	0.144

Table 1: Visual quality and bitrate comparison for different categories and methods within each category. For SFID and GC, the AFD is 2. Overall on average, our SFID34 framework reduces the bandwidth consumption by 18.6% while improving visual quality by 0.239 dB compared with H.264.

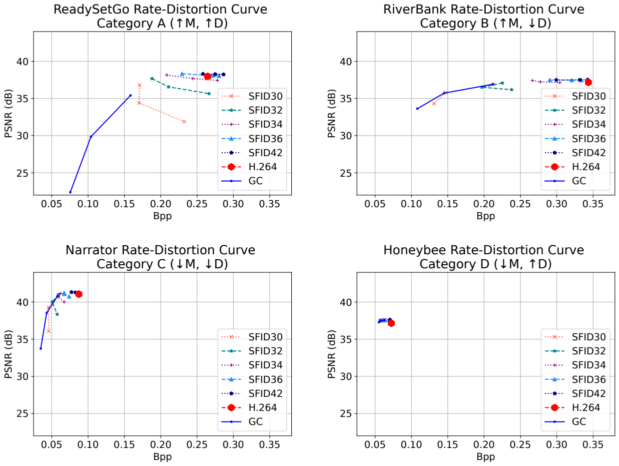

Figure 5: Rate-Distortion Curve for representative sequences. Using the categorization of Figure 3.

As the macroblock inclusion threshold increases, we see that the curves move toward the H.264 baseline numbers. The reason for this is that to meet the threshold, a significant percentage of the macroblocks need to be encoded to ensure that it meets the PSNR threshold. At SFID34, we see that using AFD 2 (far left point for SFID34 line), we can achieve a 21.2% reduction in bitrate while also improving the average PSNR and SSIM by 0.24 dB compared with the H.264 baseline. Moreover, similar to the SFID30 case, using

AFDs greater than two are more detrimental since the results of interpolation become very ineffective.

Because of the significant motion complexity, Category A streams always have the best performance under AFD 2. Using our framework for Category A sequences, we can, on average, achieve a significant bandwidth reduction of 17.7% for SFID34 while improving the quality by 0.012 dB compared with the H.264 standard.

CATEGORY B: Category B video sequences are slightly easier to compress and apply interpolation compared to Category A. This is because while they still have extensive motion, more areas within the video sequence have less detail. A general result of interpolation frameworks is that fine details are harder to recreate properly.

For GC, the drop in PSNR from AFD 2 to AFD 8 (far left point) is not as drastic in the Category A videos. With less fine detail, the interpolation methods can easily create the interpolated frame data.

For SFID32, we see the balance in the trade-off of using more data for anchor frame sequences (i.e., smaller AFDs) versus adding macroblocks to the hybrid frames to fix areas. At AFD 2 (top middle point), the interpolation framework can nearly achieve the same PSNR as the original H.264 sequence. At AFD 4 (far left point of the three), interpolation still works reasonably well with a further reduction in PSNR. At AFD 8, however, a significant number of macroblocks need to be added to the hybrid frame to correct interpolation errors. At SFID34, we see that *RiverBank* can be compressed with SFID using 22.4% less bits with slightly higher PSNR.

Across the entire data set, using our framework for Category B sequences, we can, on average, achieve a significant bandwidth reduction of 19.36% for SFID34 while improving the quality by 0.297 dB compared with the H.264 standard.

CATEGORY C: The Category C sequences have the characteristics of containing slow motion with great blurred or smoothed low complexity regions. Similar to Category B, the GC model drops video quality as the AFD is increased due to the fine details not being able to be synthesized with a greater distance between anchor frames. In addition, because of the relatively small amount of motion, the basic H.264 compression works reasonably well; that is, motion compensation works well, resulting in our baseline video compression averaging a very low 0.065 bits per pixel (bpp). This is lower than what we have observed in Category A (0.198 bpp) and Category B (0.253 bpp) sequences, even at AFD 2. Overall for Category C, the SFID34 framework lowers the bitrate from 0.065 bpp to 0.049 bbp, a savings of 24.61%.

CATEGORY D: The Category D sequences have the characteristics of being relatively static and having medium to high amount of details as video complexity. Due to the very little to almost no motion in the sequence, the bitrate compression that H.264 performs is very close to that of the GC model (from 0.0.82 down to 0.073 bpp). Given the low complexity, this category is much easier for the interpolations methods to synthesize, even for larger AFDs.

For the SFID framework and GC model, we see that the interpolation can improve the video quality slightly while using less data. This is because the quantization parameters for B-frames in DCT compression are usually higher (lower quality), which means that for easier-to-interpret frames, a very marginal improvement can be gained for the areas that require encoding in the B-frames in the H.264 sequence. Overall for Category D, we can achieve a bitrate reduction of 10.9% and improve the PSNR by 0.479 dB compared with H.264. Although not as high a bitrate reduction, there is still quite a reduction in bitrate with an improved PSNR.

5 DISCUSSION

In this section, we highlight some discussion points, limitations, and future work of our proposed hybrid video compression framework.

Video frame interpolation models: The VFI models need to be either transmitted or stored with the decoder. This implies that the VFI models will most likely be general in nature (i.e., have to apply to all video types) or will need to be relatively small to save the cost of transmitting the model for decompression. Further, they will require a bit more computation to generate the image.

In their current form, VFI models are relatively limited as the distance between anchor frames grows. While VFI models may eventually be better at capturing non-linear motion, one of the fundamental limitations will be that in order to capture such motion better, multiple anchor frames may be required. One possible remedy would be to analyze the video for the interpolated frames and provide a hint as to the motion on the encoder side. This, of course, would require additional data to be transmitted with the stream.

Improvements in VFI models can be integrated into our framework. These would presumably help reduce the number of macroblocks included in the hybrid stream.

DCT-based coding: DCT-based video coding is block-based, which can result in blocking artifacts. Such artifacts can have a negative impact on video frame interpolation since they introduce features that do not exist in ground truth images.

Our implementation currently uses fixed anchor frame sequences. Tailoring the anchor frame distance to the underlying video itself could improve both the bitrate while keeping the video quality the same. For scenes similar to Class D, longer AFD sequences can be used to save additional bandwidth. Then, for scenes with a lot of motion or detail moving to shorter AFDs would be useful.

Some of the ways that we can improve our framework are as follows. We have chosen to use PSNR as the basic metric for whether or not to include a macroblock into a stream. The effect of more visual metrics like SSIM and their impact on SFID compression is unknown. Another possibility is to provide a more dynamic, content-tailored MPEG GOP Size and AFD. We can find a dynamic way to find whether to apply our framework or not based on video complexity. Furthermore, we can find an automated work or algorithm to measure the complexity of sequences.

6 CONCLUSION

In this paper, we have presented our sequential hybrid DCT-based and interpolation-based video compression framework. SFID can effectively combine the best of DCT-based approaches and video frame interpolation approaches. Our results show that for videos with significant motion, we can save nearly 20% of the bandwidth with no loss of visual quality. For videos with less motion that are easier for DCT-compression algorithms to compress, savings of 10% in bandwidth with slightly improved video quality are still possible.

REFERENCES

[1] Muzzafer Ali and Suchetana Chakraborty. 2022. Enabling video conferencing in low bandwidth. In *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 487–488.

[2] Ahmed Aliyu, Abdul H Abdullah, Omprakash Kaiwartya, Yue Cao, Jaime Lloret, Nauman Aslam, and Usman Mohammed Joda. 2018. Towards video streaming in IoT Environments: Vehicular communication perspective. *Computer Communications* 118 (2018), 93–119.

[3] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. 2011. A database and evaluation methodology for optical flow. *International journal of computer vision* 92, 1 (2011), 1–31.

[4] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J Sullivan, and Jens-Rainer Ohm. 2021. Overview of the versatile video coding (VVC) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 10 (2021), 3736–3764.

[5] Ronald Bultje. 2015. VP9 encoding/decoding performance vs. HEVC/H.264. https://blogs.gnome.org/rbultje/2015/09/28/vp9-encodingdecoding-performance-vs-hevch-264/. [Online]. Accessed: July 11, 2022.

[6] Naveen Cheggoju and Vishal R Satpute. 2022. Blind quality scalable video compression algorithm for low bit-rate coding. *Multimedia Tools and Applications* (2022), 1–16.

[7] Seong-Ping Chuah, Chau Yuen, and Ngai-Man Cheung. 2014. Cloud gaming: A green solution to massive multiplayer online games. *IEEE Wireless Communications* 21, 4 (2014), 78–87.

[8] John Cristy. 1999. ImageMagic. https://imagemagick.org. [Online]. Accessed: May 22, 2022.

[9] Andjela Dasic. 2021. H.264 vs. H.265: What they are and which one is better. https://www.brid.tv/h-264-vs-h-265-what-they-are-and-which-one-is-better/. [Online]. Accessed: July 10, 2022.

[10] Walter Fischer. 2020. Video coding (mpeg-2, mpeg-4/avc, hevc). In *Digital Video and Audio Broadcasting Technology*. Springer, 125–175.

[11] GMDT Forecast et al. 2019. Cisco visual networking index: global mobile data traffic forecast update, 2017–2022. *Update* 2017 (2019), 2022.

[12] Yeganeh Jalalpour, Li-Yun Wang, Wu-Chi Feng, and Feng Liu. 2020. FID: frame interpolation and DCT-based video compression. In *2020 IEEE International Symposium on Multimedia (ISM)*. IEEE, 218–221.

[13] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. 2018. Super SloMo: high quality estimation of multiple intermediate frames for video interpolation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] MB Kaplan, D Ward, CA Hagberg, G Berci, and M Hagiike. 2006. Seeing is believing: the importance of video laryngoscopy in teaching and in managing the difficult airway. *Surgical Endoscopy and Other Interventional Techniques* 20, 2 (2006), S479–S483.

[15] Lucie Lévêque, Wei Zhang, Christine Cavaro-Ménard, Patrick Le Callet, and Hantao Liu. 2017. Study of video quality assessment for telesurgery. *IEEE Access* 5 (2017), 9990–9999.

[16] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. 2017. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision*. 4463–4471.

[17] Mina Malekzadeh. 2020. IP video compression comparison for smartphones in big data era. *Journal of Engineering Science and Technology* 15, 1 (2020), 600–616.

[18] Alexandre Mercat, Marko Viitanen, and Jarno Vanne. 2020. UVG dataset: 50/120fps 4K sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*. 297–302.

[19] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. 2018. Phasenet for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 498–507.

[20] Inc. Netflix. 2015. Digital video sequence at xiph.org video test media. https://media.xiph.org/video/derf/. [Online]. Accessed: May 22, 2022.

[21] Simon Niklaus and Feng Liu. 2018. Context-aware synthesis for video frame interpolation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[22] Simon Niklaus and Feng Liu. 2020. Softmax splatting for video frame interpolation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[23] Simon Niklaus, Long Mai, and Feng Liu. 2017. Video Frame Interpolation via Adaptive Convolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[24] Red5 Pro. 2019. Top 3 reasons why low-latency live video streaming matters. https://www.red5pro.com/blog/top-3reasons-why-low-latency-live-video-streaming-matters/. [Online]. Accessed: July 10, 2022.

[25] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology* 22, 12 (2012), 1649–1668.

[26] FFmpeg team. 2000. FFmpeg. http://ffmpeg.org. [Online]. Accessed: May 12, 2022.

[27] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H. 264/AVC video coding standard. *IEEE Transactions on circuits and systems for video technology* 13, 7 (2003), 560–576.

[28] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. 2019. Video enhancement with task-oriented flow. *International Journal of Computer Vision* 127, 8 (2019), 1106–1125.

[29] Zhefei Yu, Houqiang Li, Zhangyang Wang, Zeng Hu, and Chang Wen Chen. 2013. Multi-level video frame interpolation: Exploiting the interaction among different levels. *IEEE Transactions on Circuits and Systems for Video Technology* 23, 7 (2013), 1235–1248.