GameCodec: Neural Cloud Gaming Video Codec

Hoang Le[1]
hoanle@qti.qualcomm.com
Reza Pourreza[1]
pourreza@qti.qualcomm.com
Amir Said[1]
asaid@qti.qualcomm.com
Guillaume Sautiere[2]
gsautie@qti.qualcomm.com
Auke Wiggers[2]
auke@qti.qualcomm.com

[1] Qualcomm AI Research, San Diego, USA
[2] Qualcomm AI Research, Amsterdam, The Netherlands

Abstract

We present *GameCodec*, the first neural video codec designed for cloud gaming: a type of online gaming where the video game is streamed from a remote server to a user's device. This application is a challenge for video compression, as players often expect low latency and high visual quality. Additionally, gaming video is often inherently challenging to compress due to extreme camera and object motion, rich textures and visual effects. Although neural video codecs have already shown great progress on natural videos, there is so far little work on compressing gaming video. Furthermore, existing neural codecs are unable to take useful game engine information into account. In this work, we introduce a novel neural network based cloud gaming codec that leverages rendering information in an end-to-end fashion. Specifically, we introduce **DMC**, a decomposed motion compensation method that splits movement in the video into two separate steps: camera motion and object motion. We demonstrate the effectiveness of our method by showing substantial bit rate savings compared to both neural and classical baseline codecs.

1 Introduction

Modern video games require increasingly powerful hardware to render frames in high quality. For many consumers, this means the latest games can only be played using expensive desktop computers with heavy duty graphical processing units. Cloud gaming is a type of online gaming that aims to address this issue. Instead of rendering a game locally, frames are rendered on the server side and transmitted to a client device. The client device is only tasked to decode the video feed in real-time and transmit user input back to the server. We show this setup in Figure 1.

[‡] Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

© 2022. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

Rendering server-side has advantages for both users and game developers: users can play the latest games without requiring expensive hardware, while game developers can overcome the heterogeneity of client-side hardware, and can optimize their games for known server-side hardware. Cloud gaming is becoming increasingly popular, and most cloud providers offer their own platform, examples including Sony PlayStation Now [42], Microsoft Xbox Game Pass [29], NVIDIA GeForce Now [34], Amazon Luna [2] and Google Stadia [9].

Nevertheless, cloud gaming poses many technical challenges, in particular the coding of high quality frames (4K 60fps) under extremely low-latency constraints (100ms). Codecs such as H.264 [51] or H.265 [42] have poor coding efficiency in this extreme low-latency setting. Additionally, the extreme motion characteristics of gaming videos pose a challenge for video coding algorithms. Most existing video codecs are designed for natural videos with well-controlled motion, and therefore cannot handle large and abrupt camera movement or chaotic object motion well. [31]. This results in large bandwidth need, with recommended bandwidth between 15 (720p 60fps) to 40 (4K 60fps) Mbps for comfortable use [9, 32]. In contrast, streaming services such as Netflix recommend 15Mbps at 4k 24fps [33]. This is a potential issue both for the cloud service provider, which will have a hard time scaling to a large number of users, and the end-user, who will need to have a stable and fast connection.

Many works [16, 25, 31, 40] have tried to improve the coding efficiency in the cloud gaming setting. Existing solutions [16, 25, 31] typically detect focal points using rendered context, and exploit the spatial rate control capabilities of H.264 and H.265 codecs by setting per-block quality parameters. These works introduce algorithms that automatically retrieve the regions-of-interest (ROI) that correspond to areas the player is likely to pay attention to. They use the resulting importance map to modify the bitrate allocation spatially. The sacrifice in bitrate in large non-important regions together with the high fidelity of the relatively small important region lead to lower bandwidth requirement in expectation, all while preserving perceptual quality. Current solutions are often combinations of learned components and handcrafted codecs. However, despite recent advances in neural video codecs for natural video, no solution exists that is fully neural network based.

In this work, we propose *GameCodec*, the first neural video codec designed for rendered content in cloud gaming applications. Learned codecs have several advantages over standard codecs in the cloud gaming setting such as optimization of perceptually relevant objectives through ROI-coding [36] or easy adaptation to a new domain by finetuning of their parameters for it [43, 46, 47]. Especially, learned codecs can make better use of rendered context from the game engine to improve coding efficiency, using a holistic approach to integrate context information in the coding process. By leveraging this property, our *GameCodec* introduces a novel *Decomposed Motion Compensation* method that employs rendering depth and camera poses to compensate for the extreme motion in gaming videos in two consecutive steps: camera motion and object motion. We show the benefit of using rendered content as input to our newly introduced camera motion compensation by showing a 26.68% Bjøntegaard Delta (BD) [4] rate savings on the TartanAir [49] dataset and 44.22% on AirSim [40] dataset compared to a baseline neural codec (SSF [1]).

In summary, the contributions of this paper are:

1. The first end-to-end neural video codec designed for cloud gaming applications,
2. A decomposed motion compensation method that addresses the fast and abrupt motion typical for gaming videos,
3. The complete pipeline from data collection to development and evaluation of neural video codecs on rendered content videos with auxiliary rendering information.

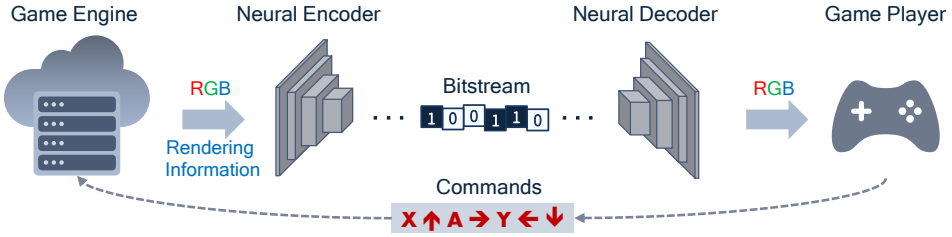

Figure 1: Cloud-Gaming with end-to-end neural cloud gaming codec

2 Related Work

Neural video compression. There has been a substantial increase in neural video coding research in recent years [1, 12, 14, 22, 26, 27, 36, 37, 38, 39, 53, 55]. Most works [1, 12, 22, 26, 27, 38, 39, 53] have focused on the *low-delay*, or *P-frame* setting. In this mode, video frames are transmitted using previously decoded frames as context, enabling real-time video transmission. This setting is therefore well-suited for cloud gaming.

In the low-delay setting, the latest neural P-frame codecs [1, 12, 39, 53] outperform the H.264 [51] and H.265 [44] codecs on standard compression datasets like UVG [28] and MLC-JVC [48] in the RGB color space. Motivated by this success, we build a neural P-frame codec for rendered content based of the neural codec SSF [1], as it is likely to outperform H.264. H.264 is the workhorse of cloud gaming platforms like NVIDIA GeForce Now [34], although other competitors like Stadia [9] reportedly use the more recent codecs VP9 [32] and AV1 [15].

Cloud gaming codecs. Research on video coding for cloud gaming applications is split into two broad categories: (1) coupling the game engine and coding process to render and code frames more efficiently [41] and (2) modifying the video encoding process only [16, 25, 31]. The first category is inconvenient for game developers, as it requires adapting the game engine, and assumes availability of non-trivial operations on end-user devices. For example, the double 3D image warping strategy of Shi et al. [41] requires the game engine to detect a future probable viewpoint that can be used as reference for encoding later frames, then render and encode it to later be used as reference for encoding frames that will be displayed. Additionally, it requires the end-user device to perform 3D image warping.

The second category, where only the video encoding strategy is modified, is more appealing to game developers. Here, cloud gaming platforms carry the burden of adapting the video coding strategy, as the inter-dependency between the game engine and the video codec is reduced to a minimum. For instance, CAVE [16] makes use of the depth map and pencil buffer to define an importance map, later used to spatially control the rate, exploiting the ROI-coding capability of HEVC [44]. Spatial rate control allows to code areas the player is likely to attend with greater quality, at the expense of other areas, allowing on average a reduction in bitrate for the same perceived quality. This exploits a property of the human visual system where acuity degrades exponentially from the focal points [50]. It is interesting to note that while a similar work, DeepGame [31], makes use of a neural network to generate the importance map, to the best of our knowledge, there is no existing work that uses a fully neural video codec in the gaming setting.

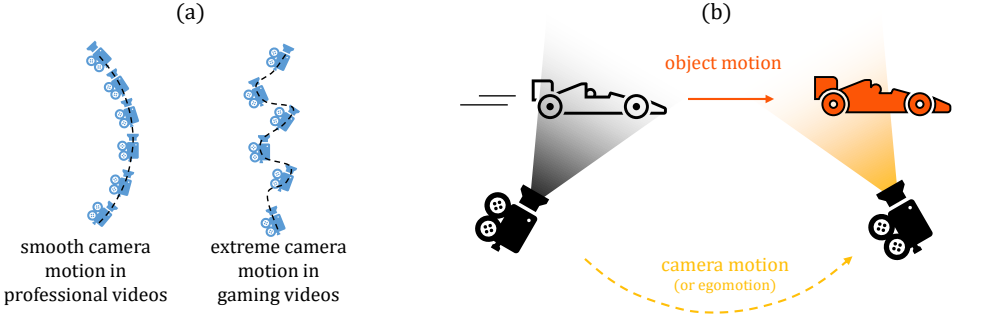

Figure 2: (a) Camera motion in professional videos versus camera motion in gaming videos. (b) Schematic view of the difference between camera and object motion in a scene.

3 GameCodec: a cloud gaming video codec

Motion in cloud gaming video. In natural videos or animations, the content creator has full control of the camera movement, and the viewer has no control over the viewpoint. Because of this lack of control, abrupt viewpoint changes typically lead to motion sickness [19]. Therefore, most professionally created video content tend to have steady, predictable camera movements to allow the viewer to follow and understand the content [6, 7]. For casually captured videos, on a phone for instance, camera movement is often more arbitrary. To alleviate shaking or other chaotic motions, stabilization methods that are hardware-based [10] or software based [20, 24] are sometimes used.

In contrast, the camera in cloud gaming videos is controlled by the player, leading to higher tolerance for abrupt scene changes. As a result, camera motion is often fast and abrupt changes happen regularly as the player reacts to events in the game, especially for first-person and third-person action games. Figure 2(a) illustrates the differences in camera motion between the two settings schematically. Extreme camera movements make the motion compensation step when coding the video more challenging. As a result, video codecs will likely rely more heavily on residual coding to compensate for errors resulting from ineffective motion compensation. As residuals are more expensive to transmit than the often highly correlated motion vectors, this can result in a higher bit cost.

Beside extreme camera motion, complex object motion poses an additional challenge. For example, cars will display fast rigid motion in racing games, and particle effects like smoke and motion blur result in chaotic motions. The difference between these two types of motion is visualized in Figure 2(b): the camera motion is caused by the movement of the camera, and the object motion is caused by objects changing or moving in the scene.

Decomposed Motion Compensation. Motivated by video stabilization methods [20, 24], we address this challenge of extreme motion in gaming videos by introducing **DMC**, a **D**ecomposed **M**otion **C**ompensation strategy that breaks down the complex motion into distinct camera motion $\hat{\mathbf{f}}^{cam}$ and object motion $\hat{\mathbf{f}}^{obj}$, and performs motion compensation in a two-step approach. This strategy is visualized schematically in Figure 2.

We incorporate this strategy in a neural codec, as shown in Figure 3. First, the *Camera Motion AE* estimates the camera flow field $\hat{\mathbf{f}}_t^{cam}$ using rendered content information, including depth map $\mathbf{d}_t$ and camera poses C_{t-1} and C_t. We obtain a first prediction $\tilde{\mathbf{x}}_t^{cam}$ by warping $\hat{\mathbf{x}}_{t-1}$ with $\hat{\mathbf{f}}_t^{cam}$. Second, the *Object Motion AE* estimates the object scale-space flow field $\hat{\mathbf{f}}_t^{obj}$

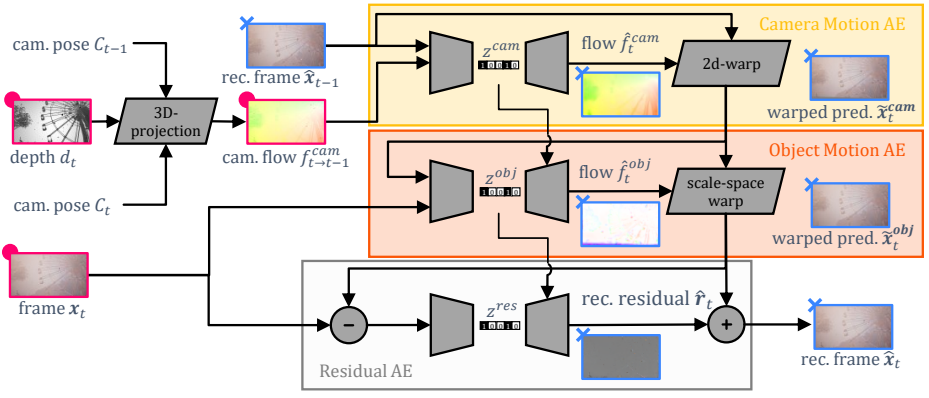

Figure 3: The GameCodec architecture. Content produced by the game engine is shown in pink and is annotated with a circle, content from the codec is shown in blue and is annotated with a cross. All autoencoders are mean-scale hyperpriors [8, 30].

using both the ground-truth frame $\mathbf{x}_t$ and the camera motion warped prediction $\tilde{\mathbf{x}}^{cam}_t$. Finally, we obtain a refined prediction $\tilde{\mathbf{x}}^{obj}_t$ by warping $\tilde{\mathbf{x}}^{cam}_t$ with $\hat{\mathbf{f}}^{obj}_t$. This warped prediction is used to compute an input to the *Residual AE*, which transmits the residual. The quantized latent of each autoencoder is shared with the subsequent autoencoder, similar to SSF [1].

Camera Motion Compensation: To compensate for camera motion – or egomotion – of a video, a method first needs to estimate the camera motion. For instance, Liu et al. [23] use 3D camera motion estimation to obtain a 3D camera path for motion compensation, while Liu et al. [24] model camera motion as a subspace of the motion field throughout a sequence. While such methods produce satisfactory results for the video stabilization applications, they often rely on the accuracy of feature tracking and fail when the motion is too extreme or feature points are missing. For cloud gaming video, it is possible to extract auxiliary rendering information such as depth, camera pose, normal, or albedo from the *G-buffer*: a screen space representation of geometry and material information stored as intermediate representation by most recent game engines like Unreal [3] or Unity [13].

We leverage this camera pose and depth to enable egomotion compensation. Specifically, given a depth d_t, camera pose C_t of a frame t and camera pose C_{t-1} of its previous frame $t-1$, we can compute the motion vector $\mathbf{f}^{cam}_{t\to t-1}$ to map pixels from from t to $t-1$ [45, 49]. Note, $\mathbf{f}^{cam}_{t\to t-1}$ only captures the appearance changes cause by camera motion of static objects in the scenes. It does not capture the motion of dynamic objects or dynamic visual effects in the scene. We use the Camera Motion AE, parametrized as a mean-scale hyperprior AE network [8, 30], to compress and send the motion vector from a transmitter to a receiver. The estimated motion vector $\hat{\mathbf{f}}^{cam}_t$ is then used to warp the previous reconstructed frame $\hat{x}_{t-1}$.

An alternative strategy would be to transmit the depth map d_t and camera poses C_t, C_{t-1} from a transmitter to a receiver and compute the motion vector at the receiver side. However, it is often cheaper to send motion vectors. For example, when the camera is static, the motion vector would be all-zeros, but depth information would still be expensive to transmit.

Object Motion Compensation: Camera motion compensation takes care of egomotion, but likely cannot handle the changes caused by dynamic object motion and visual effects in the scene. Furthermore, it is challenging to estimate object motion in case of occlusions [5, 11] and chaotic motions. To handle these challenging motions, we use *scale-space* motion

Figure 4: Example images from the Tartan Air dataset [49].

Figure 5: Example images from our dataset created using AirSim [41].

warping [1]. The *Object Motion AE* is based on the Motion AE in the SSF model, i.e., it is a mean-scale hyperprior AE which estimates the object motion scale-space vector field $\hat{\mathbf{f}}_t^{obj}$. This field is used to warp the previous warped prediction $\tilde{\mathbf{x}}_t^{cam}$ into a final warped prediction $\tilde{\mathbf{x}}_t^{obj}$ before residual coding.

Loss function. GameCodec is trained end-to-end using a rate-distortion loss, with a parameter β used to trade off rate for distortion:

$$\mathcal{L}_{\text{RD}}(x) = \mathcal{L}_D(\mathbf{x}, \hat{\mathbf{x}}) + \beta \cdot \mathcal{L}_R(\mathbf{z}). \tag{1}$$

Here, the distortion loss measures error between ground truth $\mathbf{x}$ and reconstruction $\hat{\mathbf{x}}$, and the rate loss measures the transmission cost for all quantized latent variables $\mathbf{z}$. The distortion loss $L_D = \mathbb{E}\left[||x-\hat{x}||_2^2\right]$ measures the mean squared error between target x and reconstruction $\hat{x}$ in the pixel domain. We use a rate loss $\mathcal{L}_R = \mathbb{E}\left[-\log p_\theta(z)\right]$, which is the expected negative log-likelihood of the quantized latent under the (learned) prior, typically measured in bits per pixel. Using an entropy coding algorithm, one can compress the given latents under the learned prior to (nearly) this bitrate.

4 Experiments

4.1 Dataset

TartanAir Dataset. We use the TartanAir dataset [49] for training and evaluation. This dataset contains 18 photo-realistic simulation environments generated using the Unreal Engine [8], with challenging visual effects like day-night cycles, weather effects or seasonal changes. The video frames have a resolution of 640×480.

TartanAir contains multiple video sequences per environment, with an actor traversing different paths through each environment. The videos were captured with both *left* and *right* cameras attached on the actors (a simulated drone) in two different modes: *easy* and *hard*, based on their motion patterns. We use the *easy* and *left* videos for our experiments. Specifically, we select $n-1$ *left* and *easy* sequences from each environment as a training set (*TartanAir-train*) and the last sequence as a test set (*TartanAir-test*), where n is in range from 6 to 36 and is different for each environment. In total, we use 188 sequences for our training and 18 sequences for our test dataset. Figure 4 shows example frames from the datasets.

AirSim Dataset. To further assess the performance of ***GameCodec***, we follow [49] and leverage the tool AirSim [40] to collect a high-resolution testing dataset. The environments used for this dataset are different from the ones provided in the TartanAir dataset. We refer to this dataset as ***AirSim-test***. Specifically, we used eight interesting environments with dynamic content provided by AirSim, including: *AbandonedPark*, *Africa*, *AirSimNH*, *Building_99*, *CityEnviron*, *Coastline*, *LandscapeMountains*, and *ZhangJiajie*. Using the AirSim [40] tool, we collect two sequences of 150 frames from each environment and use it as our testing dataset. In total, this dataset contains 16 videos of 150 frames/video. The videos were captured at a 1280×704 resolution. To simulate realistic gaming environments with visual effects and chaotic motions, we enable all the weather effect simulation including *raining*, *snowing*, *foggy*, *cloudy* and *falling leaves*. For each frame, we capture a rendered RGB image and its corresponding depth planar and camera pose. While the interactive mode from AirSim provides a convenient way to collect these data, we realized a few issues with this approach, including slow capturing rate and mismatching capturing times (image, depth, and camera pose of a same frame are captured at slightly different times). These mismatched data can hinder the effect of using rendering information. We thus follow [49] and use the *pause feature* to enforce consistent time of the data captured for each frame. Figure 5 shows example frames from the dataset. Please refer to our supplementary material for the metadata details to reproduce this dataset.

One criticism of using AirSim as evaluation data is that this dataset is not representative for the many diverse game settings. Our counterargument to this point is that the AirSim data is representative of an important subset of common games: those with realistic environments, where the player sees the scene from a first-person view. Although the collected scenes are simpler than games with multiple agents moving in the scene, their simplicity facilitates easier reproduction by others.

4.2 Training and performance

Training details. We train GameCodec using the original loss $\mathcal{L}$ (Eq. 1) for 1.1M steps, using a batch size of 8, where each example in the batch is a sequence of 3 frames (one I-frame and two P-frames). We first train GameCodec using the Adam optimizer with a learning rate of 10^{-4} for 1M steps with frames cropped at 256×256 resolution. We then finetune the models for another 100K steps with a learning rate of 10^{-5} where frames are cropped at 384×384.

Baseline methods. We compare our work to both learned and handcrafted low latency video codecs.

Neural codecs: We reimplemented SSF [1], a recent high performance neural codec, and report our own scores on the test datasets, as the original paper did not report scores on any rendered content datasets. As SSF was originally designed for natural videos without rendering information, we also test a modified version of SSF in which its motion autoencoder takes both camera-motion-based optical flow and two corresponding frames as input. We call this method as SSF-*enhanced* in our experiments. These methods are also trained from scratch on the *TartanAir-train* using the same training scheme as GameCodec. Although more recent neural baselines with strong rate-distortion performance exist, they do not have open source implementations available, making direct comparison on rendered content data difficult.

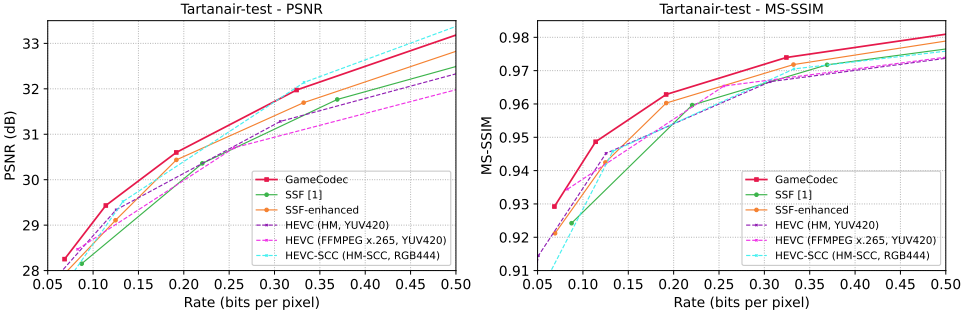

Figure 6: Rate-distortion performance on TartanAir-test dataset [29]

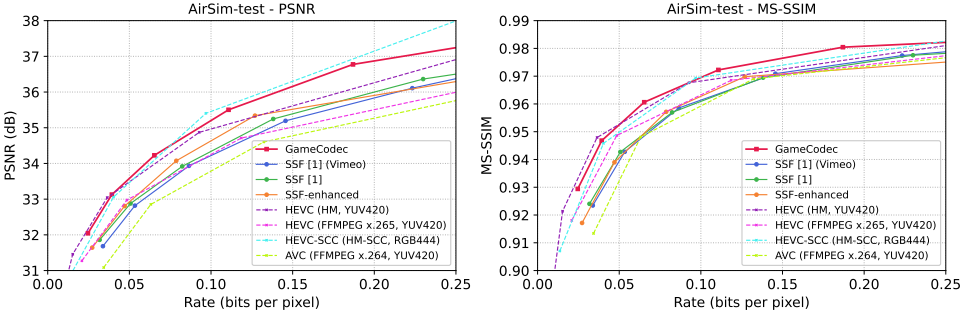

Figure 7: Rate-distortion performance on AirSim-test dataset [40]

Handcrafted codecs: We also compare to the standard codec AVC (or H.264) [51] and H.265 (or HEVC) [44], which are the common video codecs used in many existing cloud gaming settings. We used FFMPEG's implementation for both H.264 and H.265 codecs, as well as the HM reference implementation for H.265 codecs [18]. As the images were rendered in RGB color space, we converted the captured frames to YUV420 color space to use these codecs and converted their results back to RGB for the comparison. In addition, we used HEVC-SCC (HM-SCC), a reference implementation of the extended screen content version of the codec that is able to handle rendered RGB frames.

Rate-Distortion Performance

***GameCodec*:** Figure 6 and 7 show the rate-distortion (R-D) performance on *TartanAir-test* and *AirSim-test* respectively with the distortion in term of PSNR and MS-SSIM. We report Bjontegaard-Delta (BD) rate savings [4]. All results consistently show that our GameCodec outperforms the competitive neural codecs including SSF [1] and SSF-enhanced by a large margin. Specifically, it achieves 26.68% and 14.32% BD rate gain on *TartanAir-test* and 44.22% and 26.61% BD rate gain on *AirSim-test* compared to SSF [1] and SSF-*enhanced* respectively. In comparison to the handcrafted codecs, GameCodec achieves better performance than HEVC (with both HM's and FFMPEG's implementation) and AVC (FFMPEG's implementation) by a significant margin. In particular, GameCodec outperforms HM by 23.89% and 20.16% in BD rate gain on *TartanAir-test* and *AirSim-test* respectively. All these results demonstrate the effectiveness GameCodec's DMC approach to leverage rendering information to enhance video compression performance. Figure 8 visualizes the effects of its camera and object motion compensation.

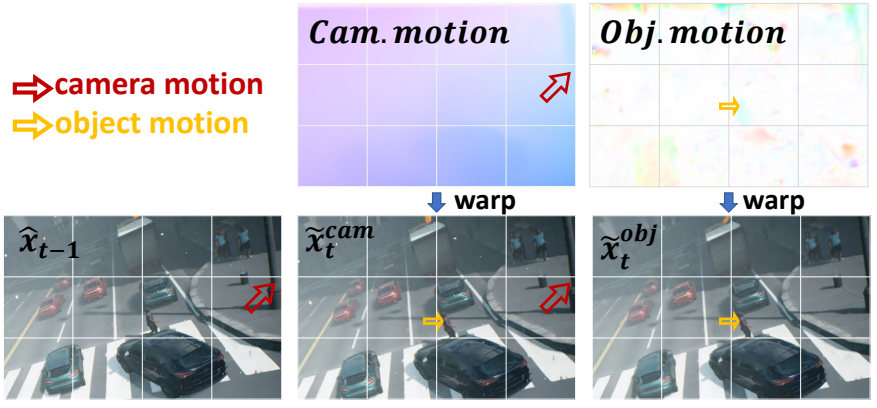

Figure 8: Example of decomposed motion compensation via camera and object motion.

***Video codecs designed for rendered content*:** The results on both datasets show that a video codec designed for rendered content videos outperforms generic ones designed for natural videos. For handcrafted video codecs, the extended version of HEVC specifically designed for screen content, HEVC-SCC, boosts the performance of HEVC significantly on both the *TartanAir-test* and the *AirSim-test* datasets. Specifically, it saves 26.07% BD rate on *TartanAir-test* and 12.02% BD rate on *AirSim-test*. We observe a similar improvement for neural video codec with the baseline model SSF [1] trained on natural video (the Vimeo90K dataset [52]) versus one trained on *TartanAir-train*. The improvement is 26.63% BD rate saving on *TartanAir-test* and 4.79% BD rate saving on AirSim.

Thanks to the flexibility of a data-driven approach, in contrast to a handcrafted video codec, a neural codec can be trained to work better on a new data distribution such as gaming videos in general or even videos from a specific game. However, simply training a neural codec originally designed for natural video (SSF [1] and SSF-*enhanced*) is not enough to fully exploit the useful rendering information. Instead, our GameCodec designed with a DMC method can effectively leverage rendering information to boost compression performance in an end-to-end approach. The results show that GameCodec can even obtain comparable RD performance to the powerful reference implementation HM-SCC.

Ablation Study. We conducted an ablation study to investigate the contribution of each of the components of GameCodec. Specifically, we conducted a leave-one-out study for the camera motion module and object motion module. In addition, as our camera motion module estimate the camera motion optical flow from the transmitter side and send it to the receiver side, an alternative approach could be transmitting its corresponding planar depth and perform depth-based 3D warping in the receiver side. This method is similar to the ones used to enhance the conventional methods [25, 41].

Figure 9 shows the performance of the methods in our ablation study. The results show that both camera and object motion compensation modules play an important role to the overall performance of GameCodec, and that their separation is useful. The alternative approach where we transmit depth instead of flow performs better than SSF-enhanced, but still underperforms the full GameCodec that transmits camera motion flows.

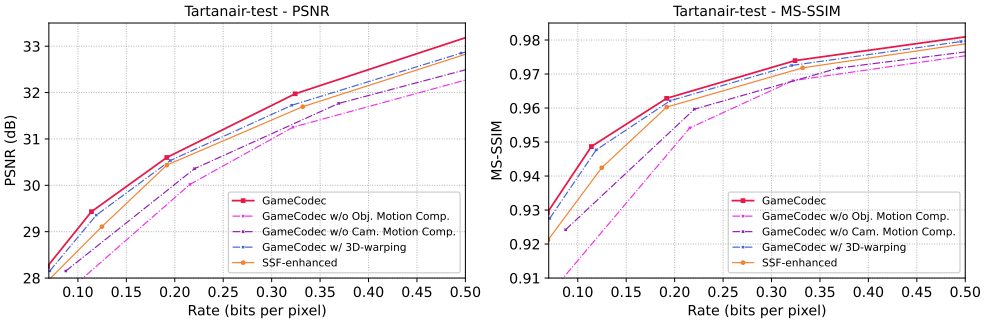

Figure 9: Ablation study showing rate-distortion performance on the TartanAir dataset, for distortion metrics PSNR (left) and MS-SSIM (right).

Complexity. As each codec is optimized for a different platform – HM and HM-SCC run on CPU, GameCodec and SSF [1] run best on GPU – it is not easy to directly compare the complexity of these methods. As an anchor for the reader, we report here the complexity comparison between the neural codecs: SSF [1] has 28.9M parameters and operates at 606 kMACs/pixel while GameCodec has slightly more parameters, 36.6M, but operates at a smaller complexity of 537 kMACs/pixel.

Discussion. We verified that TartanAir [49] is sufficiently large for training neural codecs by measuring the ability of models trained on TartanAir to generalize to the unseen AirSim dataset [40] in Figure 7, indicating overfit is unlikely. The above notwithstanding, we argue that the ability to overfit is possibly an advantage of neural (gaming) codecs. The cloud-gaming provider knows which games it serves, so it is possible to overfit the neural codec to a certain game or environment to further enhance its performance. This strategy is already deployed by NVIDIA [35] for gaming content super-resolution. However, such an approach does require training data curation, and we therefore do not explore it in this work. Future work could incorporate more rendering information such as segmentation or albedo, and could optimize directly for perceptual metrics such as LPIPS [54] or VMAF [21].

5 Conclusion

In this work, we present GameCodec, a neural video codec specifically designed for cloud gaming content. Our codec integrates rendering information provided by the game engine to improve coding efficiency. In particular, it decomposes motion compensation into compensation for egomotion, which is available from the game engine, and compensation for object motion, estimated from the video content. It performs warping for these two motions separately, which effectively makes the object motion estimation task easier to perform. As neural codecs can specialize to the data distribution they are trained on, we furthermore observe that we can easily finetune it to specific game environments through finetuning. We demonstrate 44.22% Bjontegaard-delta rate savings compared to a recent neural video codec that does not make use of rendered info, and show competitive performance against hand-crafted baselines such as HM. We have shown that neural codecs are a promising direction for the gaming and rendered content setting, and hope this work opens up the way for follow-up work in this area.

References

[1] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Ballé, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8500–8509, 2020. doi: 10.1109/CVPR42600.2020.00853.

[2] Amazon. Amazon Luna. https://www.amazon.com/luna/landing-page/. Accessed: 2022-05-18.

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkcQFMZRb.

[4] Gisle Bjøntegaard. Calculation of average psnr differences between rd-curves. 2001.

[5] Karlis Martins Briedis, Abdelaziz Djelouah, Mark Meyer, Ian McGonigal, Markus Gross, and Christopher Schroers. Neural frame interpolation for rendered content. *ACM Trans. Graph.*, 40(6):1–13, December 2021.

[6] Cantine, John., Howard, Susan., Lewis, and Brady. *Shot by shot : a practical guide to filmmaking*. Pittsburgh Filmmakers, Pittsburgh, PA, 2011.

[7] John S Douglass and Glenn P Harnden. *The Art of Technique an Aesthetic Approach to Film and Video Production*. Pearson College Division, 1996.

[8] Epic Games. Unreal engine. URL https://www.unrealengine.com.

[9] Google. Stadia. https://stadia.google.com/. Accessed: 2022-05-18.

[10] Wilko Guilluy, Laurent Oudre, and Azeddine Beghdadi. Video stabilization: Overview, challenges and perspectives. *Signal Processing: Image Communication*, 90:116015, 2021. ISSN 0923-5965. doi: https://doi.org/10.1016/j.image.2020.116015. URL https://www.sciencedirect.com/science/article/pii/S0923596520301697.

[11] Jie Guo, Xihao Fu, Liqiang Lin, Hengjun Ma, Yanwen Guo, Shiqiu Liu, and Ling-Qi Yan. ExtraNet: real-time extrapolated rendering for low-latency temporal supersampling, December 2021.

[12] Zongyu Guo, Runsen Feng, Zhizheng Zhang, Xin Jin, and Zhibo Chen. Learning cross-scale prediction for efficient neural video compression, 2021.

[13] John K Haas. A history of the unity game engine. 2014.

[14] Amirhossein Habibian, Ties Van Rozendaal, Jakub Tomczak, and Taco Cohen. Video compression with rate-distortion autoencoders. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019. doi: 10.1109/iccv.2019.00713. URL http://dx.doi.org/10.1109/ICCV.2019.00713.

[15] Jingning Han, Bohan Li, Debargha Mukherjee, Ching-Han Chiang, Adrian Grange, Cheng Chen, Hui Su, Sarah Parker, Sai Deng, Urvang Joshi, Yue Chen, Yunqing Wang, Paul Wilkins, Yaowu Xu, and James Bankoski. A Technical Overview of AV1. *Proceedings of the IEEE*, 109(9):1435–1462, Sep 2021. ISSN 1558-2256. doi: 10.1109/jproc.2021.3058584. URL http://dx.doi.org/10.1109/JPROC.2021.3058584.

[16] Mohamed Hegazy, Khaled Diab, Mehdi Saeedi, Boris Ivanovic, Ihab Amer, Yang Liu, Gabor Sines, and Mohamed Hefeeda. Content-aware video encoding for cloud gaming. In *Proceedings of the 10th ACM Multimedia Systems Conference*, MMSys '19, pages 60–73, New York, NY, USA, June 2019. Association for Computing Machinery.

[17] Zhihao Hu, Guo Lu, Jinyang Guo, Shan Liu, Wei Jiang, and Dong Xu. Coarse-To-Fine deep video coding with Hyperprior-Guided mode prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5921–5930, 2022.

[18] Joint Video Experts Team (JVET). HM reference software for HEVC. https://vcgit.hhi.fraunhofer.de/jvet/HM. Accessed: 2022-07-11.

[19] Andreas Koch, Ingolf Cascorbi, Martin Westhofen, Manuel Dafotakis, Sebastian Klapa, and Johann Peter Kuhtz-Buschbeck. The Neurophysiology and Treatment of Motion Sickness. *Deutsches Arzteblatt international*, 2018. ISSN 1866-0452. doi: 10.3238/arztebl.2018.0687. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6241144/.

[20] Johannes Kopf, Michael F Cohen, and Richard Szeliski. First-person hyper-lapse videos, July 2014.

[21] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. Toward a practical perceptual video quality metric. *The Netflix Tech Blog*, 6(2), 2016.

[22] Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu. M-lvc: Multiple frames prediction for learned video compression. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. doi: 10.1109/cvpr42600.2020.00360. URL http://dx.doi.org/10.1109/CVPR42600.2020.00360.

[23] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3D video stabilization. *ACM Trans. Graph.*, 28(3):1–9, July 2009.

[24] Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. Subspace video stabilization. *ACM Trans. Graph.*, 30(1):4:1–4:10, February 2011.

[25] Yao Liu, Sujit Dey, and Yao Lu. Enhancing video encoding for cloud gaming using rendering information. *IEEE Trans. Circuits Syst. Video Technol.*, 25(12):1960–1974, December 2015.

[26] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. doi: 10.1109/cvpr.2019.01126. URL http://dx.doi.org/10.1109/CVPR.2019.01126.

[27] Fabian Mentzer, Eirikur Agustsson, Johannes Ballé, David Minnen, Nick Johnston, and George Toderici. Neural video compression using gans for detail synthesis and propagation, 2021.

[28] Alexandre Mercat, Marko Viitanen, and Jarno Vanne. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*, MMSys '20, page 297–302, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368452. doi: 10.1145/3339825.3394937. URL https://doi.org/10.1145/3339825.3394937.

[29] Microsoft. Xbox Game Pass. https://www.xbox.com/en-US/xbox-game-pass/. Accessed: 2022-05-18.

[30] David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression, 2018.

[31] Omar Mossad, Khaled Diab, Ihab Amer, and Mohamed Hefeeda. DeepGame: Efficient video encoding for cloud gaming. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1387–1395. Association for Computing Machinery, New York, NY, USA, October 2021.

[32] Debargha Mukherjee, Jingning Han, Jim Bankoski, Ronald Bultje, Adrian Grange, John Koleszar, Paul Wilkins, and Yaowu Xu. A Technical Overview of VP9 – The Latest Open-Source Video Codec. In *SMPTE 2013 Annual Technical Conference and Exhibition*, pages 1–17, 2013. doi: 10.5594/M001518.

[33] Netflix. Netflix recommended bandwidth. https://help.netflix.com/en/node/13444. Accessed: 2022-05-23.

[34] NVIDIA. NVIDIA GeForce Now. https://www.nvidia.com/en-us/geforce-now/. Accessed: 2022-05-18.

[35] Nvidia. NVIDIA DLSS. https://www.nvidia.com/en-gb/geforce/technologies/dlss. Accessed: 2022-05-18.

[36] Yura Perugachi-Diaz, Guillaume Sautière, Davide Abati, Yang Yang, Amirhossein Habibian, and Taco S Cohen. Region-of-interest based neural video compression, 2022.

[37] Reza Pourreza and Taco Cohen. Extending neural p-frame codecs for b-frame coding. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2021. doi: 10.1109/iccv48922.2021.00661. URL http://dx.doi.org/10.1109/iccv48922.2021.00661.

[38] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander Anderson, and Lubomir Bourdev. Learned video compression. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019. doi: 10.1109/iccv.2019.00355. URL http://dx.doi.org/10.1109/ICCV.2019.00355.

[39] Oren Rippel, Alexander G. Anderson, Kedar Tatwawadi, Sanjay Nair, Craig Lytle, and Lubomir Bourdev. Elf-vc: Efficient learned flexible-rate video coding. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2021. doi: 10.1109/iccv48922.2021.01421. URL http://dx.doi.org/10.1109/iccv48922.2021.01421.

[40] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL https://arxiv.org/abs/1705.05065.

[41] Shu Shi, Cheng-Hsin Hsu, Klara Nahrstedt, and Roy Campbell. Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 103–112, New York, NY, USA, November 2011. Association for Computing Machinery.

[42] Sony. PlayStation Now. https://www.playstation.com/en-us/ps-now/. Accessed: 2022-05-18.

[43] Yannick Strümpler, Janis Postels, Ren Yang, Luc Van Gool, and Federico Tombari. Implicit neural representations for image compression. *arXiv preprint arXiv:2112.04267*, 2021.

[44] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012. doi: 10.1109/TCSVT.2012.2221191.

[45] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, September 2022.

[46] Ties van Rozendaal, Johann Brehmer, Yunfan Zhang, Reza Pourreza, and Taco S. Cohen. Instance-adaptive video compression: Improving neural codecs by training on the test set, 2021.

[47] Ties van Rozendaal, Iris A. M. Huijben, and Taco S. Cohen. Overfitting for fun and profit: Instance-adaptive data compression, 2021.

[48] Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C.-C. Jay Kuo. Mcl-jcv: A jnd-based h.264/avc video quality assessment dataset. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1509–1513, 2016. doi: 10.1109/ICIP.2016.7532610.

[49] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2020. doi: 10.1109/iros45743.2020.9341801. URL http://dx.doi.org/10.1109/IROS45743.2020.9341801.

[50] Zhou Wang and A.C. Bovik. Embedded foveation image coding. *IEEE Transactions on Image Processing*, 10(10):1397–1410, 2001. doi: 10.1109/83.951527.

[51] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 2003.

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

[53] Ruihan Yang, Yibo Yang, Joseph Marino, and Stephan Mandt. Hierarchical autoregressive modeling for neural video compression. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=TK_6nNb_C7q.

[54] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. doi: 10.1109/cvpr.2018.00068. URL http://dx.doi.org/10.1109/CVPR.2018.00068.

[55] Yunfan Zhang, Ties van Rozendaal, Johann Brehmer, Markus Nagel, and Taco Cohen. Implicit neural video compression, 2021.