ImmerScope: **Multi-view Video Aggregation at Edge towards Immersive Content Services**

Bo Chen[1], Hongpeng Guo[1], Mingyuan Wu[1], Zhe Yang[1], Zhisheng Yan[2], Klara Nahrstedt[1]
[1]University of Illinois at Urbana-Champaign, [2]George Mason University

Abstract

The multi-camera capture system is an emerging visual sensing modality. It facilitates the production of various immersive contents ranging from regular to neural videos. Although the delivery of immersive content is popular and promising, it suffers from the bandwidth bottleneck when streaming multi-view videos to the cloud (*i.e.,* multi-view video aggregation). Existing works fail to provide a bandwidth-efficient and content-generic solution. Even the closest effort to ours based on the SOTA multi-view video codecs suffers from issues of underutilized dependency and content distortion. In this paper, we present ImmerScope, a multi-view video aggregation framework at the edge with a neural multi-view video codec. It outperforms existing solutions with highly-utilized dependency via neuron connections and distortion awareness via end-to-end training. Evaluations on diverse multi-camera setups show that ImmerScope outperforms single-view codecs by at least 64% bandwidth savings in peak-signal-to-noise ratio with a frame rate of 50 fps.

CCS Concepts

• **Networks** → **Network services**; • **Computing methodologies** → **Online learning settings**; **Cooperation and coordination**; **Image compression**; • **Information systems** → **Multimedia streaming**.

Keywords

Multi-view Cameras, Immersive Computing, Video Streaming, Neural Codec, Edge Computing

ACM Reference Format:

Bo Chen[1], Hongpeng Guo[1], Mingyuan Wu[1], Zhe Yang[1], Zhisheng Yan[2], Klara Nahrstedt[1], [1]University of Illinois at Urbana-Champaign, [2]George Mason University . 2024. ImmerScope: Multi-view Video Aggregation at Edge towards Immersive Content Services. In *ACM Conference on Embedded Networked Sensor Systems (SenSys '24), November 4–7, 2024, Hangzhou, China.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3666025.3699324

1 Introduction

Recent years have witnessed the significant growth of multi-camera capture systems (MCS), an emerging visual sensing modality built from multiple 2D cameras, as illustrated in Figure 1 (left) [3, 10, 12, 50, 62, 66, 70]. The advancement in MCS gives rise to the evolution of immersive contents such as regular, stereo, 360-degree [29, 33, 64], volumetric [31, 48, 49, 53, 99], and neural [43, 50, 59, 73, 82] videos. Figure 1 (right) shows these contents benefit from more high-resolution camera views to provide better immersiveness (*i.e.,* the degree of freedom in viewing (VDoF) and visual quality).

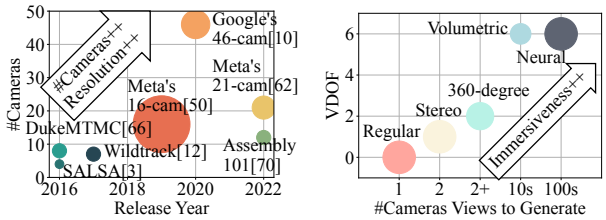


Figure 1: Left: Camera number and resolution (proportional to bubbles' area) of MCS are growing. Right: Contents benefit from more high-resolution camera views to provide better immersiveness in VDOF and visual quality (proportional to bubbles' area).

The delivery of immersive content to customers is popular and promising. One such example is that the 3D movie Avatar is the highest-grossing film of all time. Besides, the delivery of the most immersive one, the neural video [43, 50, 59, 73, 82], holds the potential to transform our lives in healthcare [20, 86], traveling [68, 69], gaming [79, 88], and shopping [61, 78]. The service that delivers immersive content from the MCS to customers is termed *immersive content service*. Figure 2 presents the workflow of immersive content service involving (1) *Multi-view Video Aggregation* that compresses multi-view videos at the edge and streams them to the cloud via real-time streaming protocols [1, 24, 40], (2) *Immersive Content Preparation* that reconstructs multi-view videos and produces immersive content in the cloud, and (3) *Content Delivery* that streams immersive content to the client device adaptively (*e.g.,* via Dynamic Adaptive Streaming via HTTP [75]). We defer the detailed motivations behind this workflow to the background (§2). The key bottleneck is the bandwidth demand for real-time streaming of high-quality multi-view videos in multi-view video aggregation. For instance, it requires an average bandwidth of at least 300 Mbps to transmit seven 2k streams at 60 fps in the Wildtrack [12] dataset with a standardized codec [84], which substantially exceeds the bandwidth available in consumer-grade networks [23].

Existing literature presents several solutions as follows.

(E1) Traditional single-view video codecs (TSVC) compress each camera view independently with handcrafted parameters [47, 76, 84]. However, they cannot effectively compress the redundant information across different views.

SenSys '24, November 4–7, 2024, Hangzhou, China

ACM ISBN 979-8-4007-0697-4/24/11
https://doi.org/10.1145/3666025.3699324

Bo Chen[1], Hongpeng Guo[1], Mingyuan Wu[1], Zhe Yang[1], Zhisheng Yan[2], Klara Nahrstedt[1]
[1]University of Illinois at Urbana-Champaign, [2]George Mason University

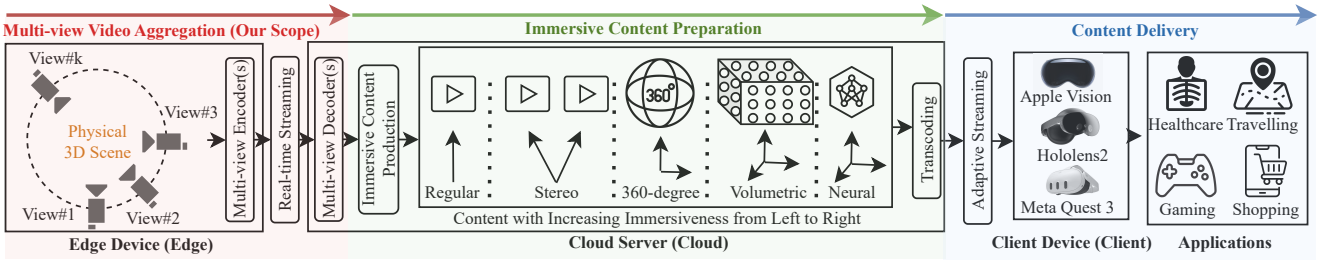


Figure 2: Immersive content service delivers immersive content from a multi-camera capture system to customers.

(E2) Media-specific solutions transform multi-view video into specific and compact contents before transmission, such as 360-degree videos [29, 33, 64] and volumetric videos [31, 49, 53, 99]. However, they are dedicated to delivering one content representation (*e.g.,* 360-degree video), which does not generalize to others (*e.g.,* volumetric and neural video).

(E3) Selective streaming solutions selectively stream multi-view content (*e.g.,* selected views [92, 94, 98] or selected image regions per view [28]) based on content redundancy. Unfortunately, advanced content such as neural videos [50, 59, 63, 82] must be prepared with a large number of high-resolution views to attain satisfactory quality. Removing multi-view content based on redundancy reduces data available for training and may compromise their quality.

(E4) Neural video codecs (NVC) recently show more promising coding efficiency (*i.e.,* the capability to reduce data size while maintaining a certain level of quality) than TSVCs. Still, neural single-view video codecs (NSVC) struggle to scale to multiple views. Despite the efforts that extend neural video codecs to stereo videos [17], there are no existing solutions that handle videos with three or more views.

(E5) Traditional multi-view video codecs (TMVC) are the closest effort to ours. TMVCs extend TSVCs by exploiting the redundancy across views in compression [41, 77, 81]. Surprisingly, our preliminary study with realistic multi-view videos [32] using the state-of-the-art (SOTA) TMVC [77] and TSVCs [76, 84] shows that *despite the computation spent on removing inter-view redundancy in TMVC, it does not effectively translate into improved coding efficiency.* Specifically, although the SOTA TMVC outperforms TSVCs with better coding efficiency, its frame rate is two orders of magnitude slower. If compared with similar frame rates, TSVCs are much more coding efficient. The problem lies in the inter-picture prediction (IPP) used by TMVCs [77, 81] that is designed for removing temporal redundancy in regular videos [47, 76, 84]. IPP fails to handle problems that arise with multi-view videos: (1) *underutilized dependency* (*i.e.,* IPP allows one view to depend on at most another view in coding, which is insufficient for multi-view videos) and (2) *content distortion* (*i.e.,* IPP inherently assumes the same content maintains the same shape in different pictures, which may be distorted in different views). These problems compromise the coding efficiency.

Our approach. In this paper, we introduce ImmerScope, a multi-view video aggregation framework at the edge with a *neural multi-view video codec* (NMVC). The NMVC consists of the neural encoder and the neural decoder built with neural networks, similar to existing NSVCs [2, 13, 15, 18, 21, 57, 65, 91]. The neural encoder compresses multi-view video at the edge and the neural decoder reconstructs the multi-view video in the cloud. The NMVC is end-to-end trained to maximize the coding efficiency. ImmerScope tackles the limitations of TMVCs with two insights. (1) *Highly-utilized dependency via neuron connections*: In the NMVC, the input and output are densely connected via neuron connections such that any pixel in the raw multi-view video may contribute to any pixel in the reconstructed multi-view video. As such, the video in any view may leverage information in other views jointly to be reconstructed, which highly utilizes the inter-view dependency. (2) *Distortion awareness via end-to-end training*: The end-to-end training minimizes the discrepancy between raw and reconstructed multi-view video and the bitrate via gradient descent [9, 46]. Gradient descent allows NMVC to correctly learn the mapping between content pixels in different views and better leverage inter-view dependency in the presence of distortion (*i.e.,* distortion awareness). By effectively utilizing inter-view dependency, ImmerScope attains better coding efficiency than TSVCs and NSVCs with a reasonable frame rate. Besides, as ImmerScope reconstructs the whole multi-view video in the cloud, it generically supports the production of various immersive content representations.

Challenges. ImmerScope tackles three main challenges.

(C1 §3.1) How to exploit inter-view dependency with distributed cameras? In the MCS, cameras are distributed to capture a scene from different angles and locations. To leverage the inter-view dependency in NMVC, we must allow raw images captured on different cameras to be processed by the neural encoder. One natural approach is hardwiring or wirelessly connecting all cameras to an edge device that runs the encoder. However, the computation overhead involved in encoding multiple views is intensive with neural codecs while the computation capability of the edge device is limited, which may result in a low frame rate. Besides, hardwiring is cumbersome (for distantly placed cameras and when reconfiguring the positions and number of cameras) and wirelessly transmitting raw multi-view data is bandwidth-intensive.

The key intuition is that *distributed source coding and end-to-end training allow NMVC to shift the computation and bandwidth overheads from the encoder to the decoder.* Specifically, distributed source coding performs separate encoding of multiple correlated sources (*i.e.,* multi-view videos) and joint decoding. As such, each edge device only processes one camera view, which scales computation overhead well to more views, and does not need communication between cameras. Meanwhile, the decoder exploits inter-view dependency on the cloud where the computation resource is more abundant, which less affects the frame rate. The bandwidth overheads are alleviated as the data of all views is located on the same machine. Most importantly, distributed source coding can theoretically

achieve the optimal rate of joint encoding and decoding [72, 87], which can be approximated via end-to-end optimization in NMVC. Based on this insight, we design the *edge-offloaded neural multi-view codec* where the distributed encoders process each view separately while the centralized decoder processes all views jointly.

(C2 §3.2) How to perform online training with bandwidth efficiency? For distortion awareness, a natural way is to collect multi-view video under a specific scene and train the codec with collected data before streaming starts. Unfortunately, the collection and training process would drastically delay the streaming. Thus, a more viable solution is *online training* that trains the codec in parallel to the streaming process, which does not delay streaming. Since the edge has a limited computation capability, we perform online training on the cloud. The problem is that online training must be supervised with the raw video data. However, the raw video data is located at the edge, whose transmission to the cloud poses a significant uplink bandwidth cost (*e.g.,* 20× the bandwidth usage for streaming the video).

The key finding is that *the end-to-end training result is particularly resilient to the temporal loss of data.* Specifically, we can remove 99% of the frames in training with a minimal impact on the video quality (*i.e.,* 0.2dB in Peak Signal-to-Noise Ratio (PSNR) [34]), without affecting the bitrate. In contrast, spatially removing the content causes drastic performance degradation. Therefore, we propose an *adaptive training data streamer* that uniformly and temporally samples training data according to a sampling interval, which is adapted to not violate a user-defined bandwidth increase.

(C3 §3.3) How to accelerate online training? Training a neural codec from scratch until convergence is a time-consuming process, which takes several days [2] on commodity GPUs and is made worse by more camera views. However, the online training duration (*e.g.,* 2 to 3 hours for a movie) is limited. It is therefore challenging to attain satisfactory codec performance in online training.

The key insight is that *the online training overhead can be partially shifted to offline by leveraging single-view videos.* Specifically, intra-view dependency exists in not only multi-view videos (not available before streaming) but also single-view videos, which can be captured in a different scene. These single-view videos allow multi-view codecs to learn intra-view dependency via offline training and amortize online training overheads. Based on this insight, we design the *hybrid online training accelerator* that separates end-to-end training into offline training to embed intra-view dependency and online training to impart inter-view dependency in the codec.

Experimental settings and results. We conduct an extensive evaluation of ImmerScope driven by two multi-view video datasets covering a total of six realistic scenes. Compared to systems based on the SOTA TSVCs and NSVCs on the WildTrack [12] dataset, ImmerScope achieves 64% and 78% average bandwidth savings at the same PSNR and the structural similarity index (SSIM) [83], respectively. For systems using SOTA TMVCs, ImmerScope improves the quality in PSNR by 3.8dB with similar bandwidth usage. Moreover, ImmerScope achieves a real-time streaming rate of 50 fps.

In summary, we make three main contributions:

(1) We present the design and implementation of ImmerScope, an edge-based multi-view content aggregation framework with a multi-view neural codec.
(2) We resolve numerous design challenges in the design of ImmerScope with novel techniques including edge-offloaded multi-view neural codec, adaptive training data streamer, and hybrid online training accelerator.
(3) We extensively evaluate ImmerScope, demonstrating its superior bandwidth efficiency and real-time frame rates.

2 Background and Motivation

2.1 Background

Content representations. Popular immersive content representations include regular, stereo, 360-degree, volumetric (point cloud/polygon mesh), and neural representation [43, 59] videos), as shown in Figure 1 (right). The regular video is the most common one that does not allow viewers to change their viewpoint, creating a single VDoF. The stereo video presents different views of a 3D scene to two eyes, creating one VDoF. The 360-degree video models a 3D scene as a spherical surface, allowing viewers to look in all directions from one location, which provides two VDOFs (*i.e.,* moving the viewpoint vertically and horizontally). The volumetric video models a 3D scene with points or meshes and allows viewers to look in all directions and from any location, which provides six VDOFs. Unfortunately, 360-degree and volumetric videos inherently suffer from artifacts caused by parallax [4, 5] and the discrete nature of points or meshes [19, 26], degrading their quality. Neural representations [43, 59] describe a 3D scene using a continuous function with learned parameters, which provides six VDOFs like volumetric video while preserving the same quality as the regular video.

Immersive content service. Immersive content service (Figure 2) involves two key design considerations (1) content preparation on the cloud and (2) real-time streaming from the edge. Here, we discuss the motivations behind them.

(1) Why the cloud? Multi-camera capture systems are installed at the edge to capture multi-view videos indoor [3, 32, 50], campus [12, 66], natural [10] scenes. Unfortunately, the edge does not have enough computing resources to produce immersive content and bandwidth resources to deliver them to multiple users. Hence, we seek to leverage the cloud for content production and delivery, which has more abundant computational and bandwidth resources than the edge.

(2) Why real-time streaming? Instead of real-time streaming, the other option is to first store captured videos on the hard drive at the edge and then upload them to the cloud later or asynchronously, this approach requires significant storage space at the edge for long-duration capturing, and delays content production and delivery.

Traditional single-view video codec (TSVC). TSVC (*e.g.,* H.262/MPEG-2 [41], H.264/AVC [84] and H.265/HEVC [76]) builds intra-view dependency to achieve high coding efficiency via *inter-picture prediction* (IPP). Specifically, TSVC divides multiple video frames into several groups of pictures (GOP), as shown in Figure 3 (the Base view). Each GOP contains one I frame, predictive frames (*i.e.,* P frames), and bi-directionally predicted frames (*i.e.,* B frames), by different colored blocks. The I frame is encoded and decoded without

Bo Chen[1], Hongpeng Guo[1], Mingyuan Wu[1], Zhe Yang[1], Zhisheng Yan[2], Klara Nahrstedt[1]
[1]University of Illinois at Urbana-Champaign, [2]George Mason University

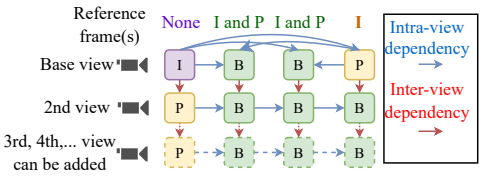


Figure 3: TMVCs establish intra-view and inter-view dependency to remove redundancy via P and B frames.

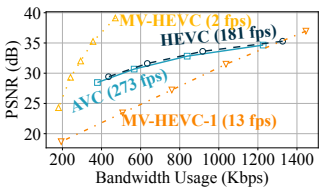


Figure 4: Comparing TMVCs and TSVCs.

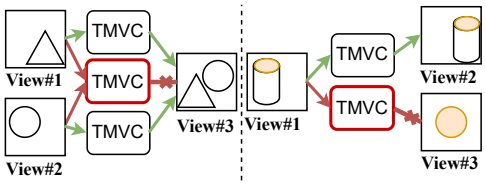


Figure 5: TMVCs suffer from underutilized dependency and content distortion problems due to inter-picture prediction.

relying on other frames. IPP instructs that P and B frames rely on one and two other frames as references, respectively. It builds the intra-view dependency (blue arrows) between the reference frame (s) to the to-be-processed frame, which means P and B frames only need to encode the differences between the content in them and the reference frame(s). Thus, IPP makes the P and B frames smaller than the I frames.

Traditional multi-view video codec (TMVC). TMVCs [41, 77, 81] improve the coding efficiency of TSVCs by leveraging view-level redundancy. The intuition is to use IPP to encode the differences between content in different views. Figure 3 illustrates how TMVC [41, 77, 81] builds view-level redundancy with IPP. The TMVC identifies one view among all views as the base view, which is similar to the role of the I frame in a GOP. The base view is encoded exactly like a single-view video. The second view encodes its first frame as the P frame that is predicted from another view (the base view in this example) and the other frames as the B frames that are predicted from frames in the same and another view. The red arrows represent intra-view dependency. Similar to the second view, the other views (third, fourth,...) can depend on one existing view to perform encoding.

2.2 Pitfall of TMVC

Preliminary study. We conduct a preliminary study to benchmark the performance of the SOTA TMVC (MV-HEVC) [77] using a realistic multi-camera setup. Specifically, We adopt the four-view video capturing the "Lobby" scene [32], with more details in §4.1. We compare MV-HEVC to two TSVCs (H.264/AVC [84] and H.265/HEVC [76]) with the default codec setting. All codecs share the same GOP structure as specified in the base configuration of MV-HEVC. SVCs ignore the GOP structure across different views. As shown in Figure 4, the MV-HEVC by the default configuration outperforms TSVCs in coding efficiency. However, its frame rate is two orders of magnitude smaller. For a fair comparison at a comparable frame rate, we adjust the *coding depth* of x265-mv. (Note that we attempted other configurations of the MV-HEVC, but they do not affect the frame rate as significantly as the coding depth.) A smaller value of coding depth means coarser granularity in IPP, which trades coding efficiency for computation speed. The MV-HEVC with an adjusted coding depth of 1, the lowest value, is denoted by MV-HEVC-1. MV-HEVC-1 achieves a frame rate of 13 fps, which is still one magnitude slower than TSVCs. Moreover, the coding efficiency of the MV-HEVC drastically decreases below that of TSVCs. In summary, despite the computation spent on removing inter-view redundancy in TMVC, it does not effectively translate into improved coding efficiency.

Why TMVC fails. TMVCs fail mainly because they rely on IPP, proposed to handle temporal redundancy in a regular video, which cannot handle the more challenging inter-view redundancy. Specifically, TMVCs have two limitations.

(1) Underutilized dependency. In a multi-view video, one view may overlap with two or more views (*e.g.,* view#3 overlaps with both view#1 and view#2 in Figure 5 (left)). IPP allows one view to depend on another view in coding. For instance, IPP allows us to leverage the dependency between view#3 and view#1 (or between view#3 and view#2) such that view#3 mainly needs to encode the difference (*i.e.,* the circle (or the triangle)). However, inter-view dependencies exist between both (1) view#3 and view#1 and (2) view#3 and view#2. A more efficient way is to encode view#3 based on both view#1 and view#2, which requires encoding minimal extra information. Restricted by P and B frames, IPP cannot utilize all these inter-view dependencies, causing underutilization and compromising coding efficiency. Moreover, the reference frames decided heuristically, may not capture the dependency correctly, which exacerbates this problem.

(2) Content distortion. IPP inherently assumes the same content maintains almost the same shape in different pictures (*e.g.,* views #1 and #2 in Figure 5 (right)). However, content may be distorted in views with different viewing positions and directions. However, it cannot effectively utilize the inter-view redundancy when there is a distortion (*e.g.,* views #1 and #3 in Figure 5 (right)). In this case, more additional bandwidth are used to describe the discrepancy in shapes. Nevertheless, this problem is challenging to solve with TMVCs [41, 77, 81] as the distortion varies depending on camera setups, which cannot be addressed without prior knowledge of the camera setup and manual fine-tuning of the codec parameters.

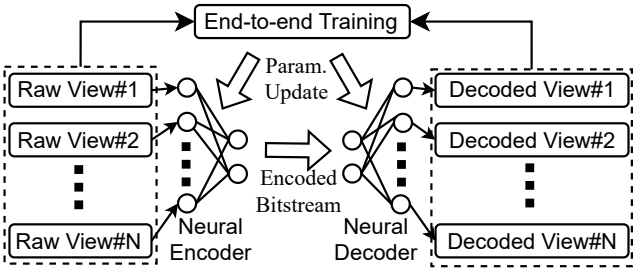


Figure 6: How neural codecs work.

2.3 Neural Multi-view Video Codec Basics

High-level workflow. The neural multi-view video codec (NMVC) extends existing NSVCs [2, 15, 18, 21, 57, 65, 91] by changing the input and output from single-view video frames to multi-view video

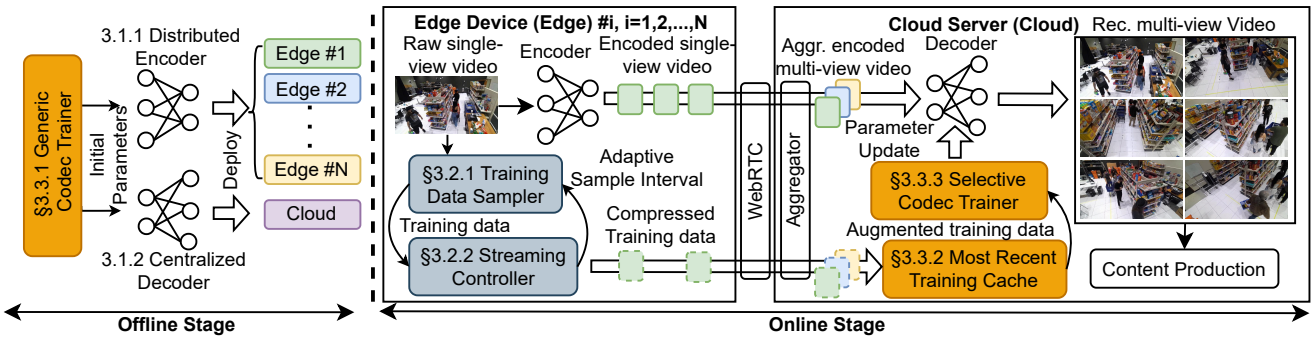


Figure 7: Workflow of ImmerScope consists of (1) an offline stage that trains and deploys the neural encoder and decoder and (2) an online stage that streams multi-view videos and training data to fine-tune the codec.

frames. As shown in Figure 6, NMVC processes the raw multi-view video into a bitstream via a neural network (*i.e.,* neural encoder) and reconstructs the decoded multi-view video from the bitstream via another neural network (*i.e.,* neural decoder). The end-to-end training (*e.g.,* SGD [9] and ADAM [46]) jointly optimizes the neural encoder and decoder by minimizing (1) the discrepancy between raw and decoded multiview videos and (2) the compressed data size, which improves coding efficiency.

Frame-level workflow of NSVC. In NSVCs [2, 57, 65], the neural encoder E compresses the raw frame x_t ($t = 1, 2, \ldots$ is the frame index) into a motion feature (MF) f_t^v and a residual feature (RF) f_t^r, by using the previously reconstructed frame $\hat{x}_{t-1}$ as a reference. The neural decoder D reconstructs the frame $\hat{x}_t$ based on the MF f_t^v and the RF f_t^r and the previously reconstructed frame $\hat{x}_{t-1}$. MFs leverage temporal redundancy while RF stores non-redundant information. Entropy coding converts MFs/RFs to bitstreams and back [39, 85], facilitating the network transmission or the storage of videos.

Camera assumptions. In this paper, we assume homogeneous cameras that have the same resolution and frame rate. We also assume the cameras are static, although we do not know their positions before streaming. The discussion of heterogeneous cameras is detailed at §5.

3 ImmerScope Design

Figure 7 illustrates an overview of ImmerScope, encompassing an offline stage and an online stage. In the offline stage, the generic codec trainer (§3.3.1) produces the initial parameters of the distributed encoder (§3.1.1), and the centralized decoder (§3.1.2). The encoder is copied and deployed on different edge devices. The decoder is deployed on the cloud server. In the online stage, each encoder encodes a unique view of the video and streams the encoded video data to the cloud using the WebRTC framework [40]. Meanwhile, the training data sampler (§3.2.1) samples training data from the raw video. The sampling controller (§3.2.2) compresses the sampled training data and adapts the sample interval. The compressed training data is streamed to the cloud along with the encoded data. Then, the aggregator synchronizes videos from all edge devices. The aggregated encoded multi-view video is decoded into the reconstructed multi-view video, which will be used for content production. At the same time, the most recent training cache (§3.3.2) augments the training data for the selective codec trainer (§3.3.3), which updates the decoder's parameters during streaming.

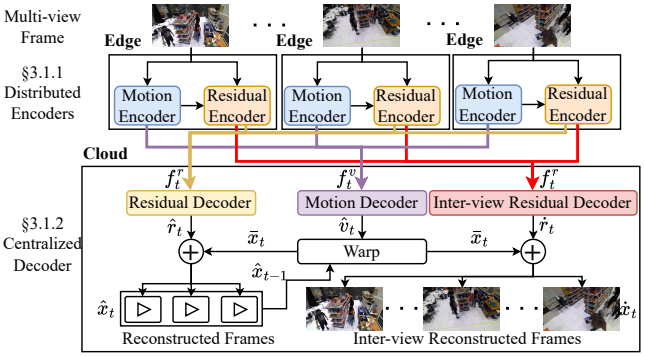


Figure 8: Edge-friendly neural multi-view video codec.

3.1 Edge-friendly Neural Multi-view Codec

The edge-friendly multi-view neural codec involves distributed encoders that process raw videos in different views independently (§3.1.1) and a centralized decoder that reconstructs videos encoded by all encoders on the cloud (§3.1.2).

3.1.1 Distributed Encoder. A natural extension of existing single-view neural codecs' frame-level workflow (§2.3) to support multi-view videos is to process multiple frames using this workflow in parallel. To easily leverage inter-view dependency, we can jointly encode all camera views with a centralized encoder at the edge and jointly decode all views with a centralized decoder at the cloud.

Challenge. While a centralized decoder is affordable for the cloud with abundant computation resources, a centralized encoder is challenging for the edge with limited computation resources. First, a centralized encoder requires the edge to be connected to all cameras, which can be cumbersome or bandwidth-consuming. Second, neural codecs are computation-intensive, even when processing a single view. Processing all views on one edge device creates a significant computation overhead, which may lead to GPU memory overflow or a low encoding frame rate.

Solution. Our intuition is that *distributed source coding and end-to-end optimization allow the multi-view neural codec to shift the computation overhead from the encoder to the decoder.* Distributed source coding is a technique that performs separate encoding of

Bo Chen[1], Hongpeng Guo[1], Mingyuan Wu[1], Zhe Yang[1], Zhisheng Yan[2], Klara Nahrstedt[1]
[1]University of Illinois at Urbana-Champaign, [2]George Mason University

multiple correlated sources (*i.e.,* multiple cameras) and joint decoding. Separate encoding circumvents the drawbacks of a centralized encoder. Moreover, it theoretically achieves the optimal rate of joint encoding and decoding in lossless (*i.e.,* the Slepian-Wolf theorem [72]) and lossy (*i.e.,* the Wyner-Ziv theorem [87]) coding. The theoretical optimal of distributed source coding may be approximated with end-to-end training.

Based on this intuition, we design the *edge-friendly multi-view neural codec* in an asymmetric way where encoders run separately on distributed edges and one centralized decoder performs joint decoding on the cloud, as illustrated in Figure 8. Although we need one edge device for each camera, this design enables easy scaling (*i.e.,* the system adapts to more views by adding the same edge device and camera without the need to upgrade the edge device). The distributed encoders perform encoding independently on separate camera sources with a motion encoder and a residual encoder, following single-view neural codecs [2, 57, 65]. The motion encoder first generates the MF from the current frame and the previously reconstructed frame (omitted in Figure 8 for clarity). MFs are used by the decoder to predict the current frame (*i.e.,* predicted frame) based on a previously reconstructed frame, leveraging the temporal redundancy. Then, the residual encoder generates the RF based on the current frame, the MF, and the previously reconstructed frame, which compensates for the discrepancy between the predicted and raw frames. We formulate the distributed encoder for the i-th view as shown.

$$f^v_{t,i}, f^r_{t,i}, \hat{x}_{t,i} = E_\theta(x_{t,i}, \hat{x}_{t-1,i}), \tag{1}$$

where $f^v_{t,i}$ and $f^r_{t,i}$ represent the motion and residual feature for the i-th view at timestamp t, respectively. $x_{t,i}$ and $\hat{x}_{t,i}$ represent the raw and reconstructed image for the i-th view at timestamp t, respectively. θ denotes the parameter of the encoder. Note that the reconstructed frame $\hat{x}_{t,i}$ does not leverage inter-view dependency, which is created and consumed by the encoder following the convention of inter-picture prediction as most codecs (*e.g.,* H.264 [84] and SSF [2]). The centralized decoder leverages inter-view dependency to generate the *inter-view reconstructed frame* for users to view, which will be discussed next.

3.1.2 Centralized Decoder. The centralized decoder concurrently aggregates N MFs and RFs via concatenation, denoted by f^v_t and f^r_t, respectively, where N is the number of camera views. Then, as shown in Figure 8, the centralized decoder derives the *intra-view reconstructed frames* in three steps that do not rely on inter-view dependency, following single-view neural codecs [2, 57, 65]:

(1) *Decoding*: The motion decoder and residual decoder decodes aggregated MFs (f^v_t) and RFs (f^r_t), deriving reconstructed motion ($\hat{v}_t$) and residual ($\hat{r}_t$).
(2) *Motion compensation*: The warp module utilizes the decoded motion ($\hat{v}_t$) and previous reconstructed frames ($\hat{x}_{t-1}$) to obtain a predicted frame ($\bar{x}_t$).
(3) *Reconstruction*: The addition module adds the predicted frame ($\bar{x}_t$) and the decoded residual ($\hat{r}_t$) to derive the reconstructed frame $\hat{x}_t$, which is the frame to be viewed.

Challenge. Leveraging inter-view dependency in the decoder is challenging. Straightforward adaptations of the neural codec would either result in non-convergence of training or delayed decoding.

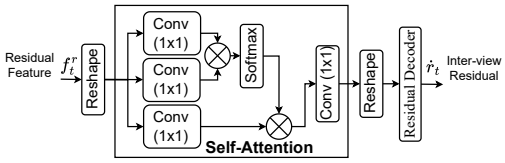


Figure 9: Design of the inter-view residual decoder.

First, a natural solution is to enhance the motion and residual decoders (*e.g.,* with the self-attention module [80], which has demonstrated notable effectiveness in correlating features along specific dimensions). We create a modified neural codec by adding the self-attention module in the motion and residual decoders and train the whole modified neural codec end-to-end. However, the training of the neural codec cannot converge and performs worse than the unmodified neural codec. The crux of the issue lies in that the neural codec requires the reference frame and the predicted frame to be consistent on the encoders and the decoder. The consistent reference frame and predicted frame ensure the motion compensation step and the reconstruction step are correctly performed, respectively. The self-attention module causes inconsistent reconstructed motions and residuals, which leads to inconsistent predicted and reference frames and compromises the coding efficiency. **Second**, the other design is to apply deep learning models [67, 80] to enhance intra-view reconstructed frames. However, this design would delay the decoding process as the enhancement must wait until the intra-view reconstructed frames are derived.

Solution. We design the *inter-view residual decoder* that leverages inter-view dependency to produce the *inter-view reconstructed residual* from RFs, which runs in parallel to the original residual decoder. The key insights are that (1) such a design ensures the convergence of the training by keeping reconstructed and predicted frames consistent and (2) the additional computation is mainly to derive residuals, which can be performed in parallel to the motion and residual decoders without incurring too much delay in decoding. More importantly, this design allows residuals to be better reconstructed by utilizing inter-view dependency, which improves video reconstruction quality.

For implementation, the inter-view residual decoder generates inter-view residual by pre-processing RFs with the self-attention module and then applying the residual decoder, as illustrated in Figure 9. The self-attention module has a single attention layer for efficiency, which contains 8 attention heads and a head dimension of 64. The residual decoder is a duplication of the original residual decoder. We apply proper reshaping in this decoder to ensure (1) features in the dimension of view, height, and width are correlated and (2) the inter-view residual has the same shape as the original residual. Finally, the inter-view residual is added to the predicted frame to derive inter-view reconstructed frames. We abstract the centralized decoder as shown.

$$\hat{x}_t, \dot{x}_t = D_\phi(f^v_t, f^r_t, \hat{x}_{t-1}), \tag{2}$$

where $f^v_{t,i}$ and $f^r_{t,i}$ denote the motion and residual feature for the i-th view at timestamp t, respectively. $\dot{x}_{t,i}$ and $\hat{x}_{t,i}$ represent the inter-view reconstructed and the reconstructed image for the i-th view at timestamp t, respectively. ϕ is the parameter of the decoder.

3.2 Adaptive Training Data Streamer

We sample data from the raw video (§3.2.1) and adapt the sampling rate according to a user-defined *bandwidth impact* (§3.2.2). The bandwidth impact is defined as the ratio of the bandwidth consumption of additional data for training to the bandwidth for video streaming.

3.2.1 Training Data Sampler. To train the neural codec, we must stream the raw video from the edge devices to the cloud.

Challenge. However, sending raw frames presents a significant bandwidth overhead (*e.g.,* 20× the streaming bandwidth usage), which slows down training and streaming. Although we can downsample the data for training, we inevitably affect the training performance.

Solution. Our insight is that *training performance is particularly resilient to temporal data loss.* Specifically, removing multiple frames in training has a negligible impact on the training result. In contrast, when we spatially remove the same amount of pixels per frame, there is a more significant drop in the training result. We defer a more detailed analysis of data sampling to §4.5. Based on this observation, we design the *training data sampler* that uniformly filters out a portion of frames based on a sampling interval (*i.e.,* the number of frames between consecutively sampled frames). We further reduce the data size by sending the difference between raw frames and reconstructed frames (*i.e.,* the streamed video) and performing lossless compression via `zlib` [55]. We do not adopt lossy image codecs like JPEG [27] and BPG [7], which may affect training unpredictably.

3.2.2 Streaming Controller. The streaming controller adapts the sampling rate of the training data sampler to attain a target bandwidth usage. Toward this goal, a naive way is to compress every frame and determine which frame to send based on the size of the compressed video and training data.

Challenge. However, the compression overhead is not negligible (*e.g.,* 44 ms per frame), making it unrealistic to compress the training data in every frame.

Solution. A key observation is that the *bandwidth impact remains stable per frame* (*e.g.,* 0.1% change during streaming). Therefore, we can rely on a short streaming duration to reliably determine the impact, which saves the computation cost in compression. Based on this observation, the *streaming controller* derives the sampling interval in two steps. First, we initialize the sampling interval T_0 to an empirical value (*e.g.,* 100). Second, we calculate the average bandwidth impact R_i resulting from the sampling interval T_0, based on a few frames of streamed training data (*e.g.,* 10). Third, we approximate the new sampling interval via $T = \lfloor R_i / R_t \times T_0 \rfloor$, where R_t is the targeted bandwidth impact such as 1%.

3.3 Hybrid Online Training Accelerator

We offline train the codec (§3.3.1) before deployment, which is performed once for any scene. During streaming, we build a training cache and train the codec continuously.

3.3.1 Generic Codec Trainer. ImmerScope aims to be generic and does not assume any prior knowledge of a scene. A straightforward way to prepare the codec for deployment is to collect multi-view video under a specific scene and train the codec with collected data. Unfortunately, the collection and training process would drastically delay the deployment. To make matters worse, this cumbersome process has to be repeated for different scenes. Therefore, we resort to another approach that randomly initializes the codec and online trains it using the streamed data.

Challenge. However, online training minimally improves codec performance. The reason is that the training of neural codecs is time-consuming, which could take days on consumer-grade GPUs (*e.g.,* NVIDIA RTX 3090). Besides, the training overhead is more considerable and increases with the number of views. The online training duration (*e.g.,* a few hours for a movie) is insufficient to fully exploit the benefits of inter-view dependency.

Solution. The key insight is that *the online training overhead can be partially shifted to offline by leveraging single-view videos.* Specifically, the dependencies in multi-view videos can be decoupled into intra-view and inter-view dependencies. Intra-view dependency exists in both multi-view and single-view videos, even when captured in different scenes. While multi-view videos may not be available before deployment, single-view videos are widely accessible for training. By leveraging single-view videos and learning intra-view dependency offline, we only need to capture inter-view dependency in online training, whose overhead is drastically reduced. Therefore, we design the *generic codec trainer* that offline optimizes the codec to leverage intra-view dependency. To do so, we must adapt the inter-view residual module (§3.1.2), configured for multi-view input, to handle single-view input. We achieve this by changing the reshaping function before and after the self-attention module, such that it only correlates the height and width dimensions, excluding the view dimension. Even though the reshaping function is different in online training, the intra-view dependency learned offline generalizes well to any number of views. Offline training optimizes the loss function of the raw frame ($\dot{x}$) and the reconstructed frame (x), as shown.

$$\mathcal{L}_0 = \mathcal{D}(\hat{x}, x), \tag{3}$$

where x is sampled from single-view videos and $\hat{x}$ is jointly derived by the encoder (Equation 1) and decoder (Equation 2). $\mathcal{D}$ measures the distance between two images, which typically can be the Mean Square Error (MSE) or SSIM.

3.3.2 Most Recent Training Cache. Intuitively, a codec should be up-to-date (*i.e.,* trained on the latest data) to have the best performance. A straightforward way is to perform back-propagation and update parameters every time we receive new training data.

Challenge. However, the codec improvement is insignificant in online training. The reason is that the number of training iterations positively correlates with the number of frames of training data, which drastically reduces after sampling (*e.g.,* by more than 99%). Even though we can train the codec repeatedly on the latest data, the codec still performs poorly.

Solution. The problem is that the codec is overfitted to a small set of frames and loses generalizability to future unseen frames. Instead, the key intuition is to *balance timeliness and generalizability in training.* We design the *most-recent training cache* that stores a few most recent frames for training, which achieves timeliness by reusing the most recent data and makes the codec generalizable by including multiple frames. During online training at the timestamp τ, the training data x_τ will be uniformly drawn from a set S (*i.e.,*

$x_\tau \sim \text{Uniform}(S(\tau))$ where $S(\tau)$ is defined in Equation 4.

$$S(\tau) = \{x_t \mid x_t \text{ is among the } C \text{ most recently received frames before } \tau.\} \quad (4)$$

The cache size C is a constant. A larger value of C improves the training result, but it should be constrained to save memory usage.

3.3.3 Selective Codec Trainer. Intuitively, we can train the codec by jointly optimizing parameters in the encoder and the decoder using data in the training cache (§3.3.2).

Challenge. However, the encoders are deployed on the edge while the training happens on the cloud. Updating the parameters of the encoders, which requires continuous sending the updates, leads to significant communication costs. Besides, gradient calculation causes a high computation cost.

Solution. Our intuition is that *the training result is resilient to the reduction of trainable parameters*. Specifically, during codec training, we can freeze a few parameters while only updating others, which still demonstrates promising results in image and video neural codecs [13, 45]. More importantly, this insight presents an opportunity to address the aforementioned problems. We design the *selective codec trainer* that optimizes the parameters of the inter-view residual decoder while freezing other parameters, which greatly improves the coding efficiency and provides two benefits. First, these parameters reside on the decoder, eliminating the communication costs associated with parameter updates. Second, the decoder contains only a small subset of the codec's parameters, saving significant computational costs. Suppose the set of decoder parameters $\phi = \{\phi_u, \phi_f\}$, where ϕ_u represents the set of parameters to be updated (*i.e.,* the parameters of the inter-view residual decoder) and ϕ_f represents the set of parameters to freeze (*i.e.,* parameters not in the inter-view residual decoder). The loss function for online training quantifies the distance between the training image drawn from the optimization cache (x_τ) and the reconstructed image ($\dot{x}_\tau$), as shown.

$$\mathcal{L}_1 = \mathcal{D}(\dot{x}_\tau, x_\tau), \quad (5)$$

where $\dot{x}_\tau$ is jointly derived by the encoder (Equation 1) and decoder (Equation 2). The parameter update can be formalized in Equation 6.

$$\phi_u = \phi_u - \eta \nabla_{\phi_u} \mathcal{L}_1, \quad (6)$$

Algorithm 1 shows the hybrid codec training algorithm.

Algorithm 1 Hybrid Codec Training.

Require: Learning rate η.
Ensure: Trained encoder (θ^*) and decoder (ϕ^*) parameters
1: Offline train parameters of encoder (θ^*) and decoder (ϕ).
2: Initialize $\phi^* \leftarrow \phi$.
3: **for** streaming timestamp τ **do**
4: Sample data $x_\tau \sim \text{Uniform}(S(\tau))$ (Equation 4).
5: Split parameters $\{\phi_u, \phi_f\} \leftarrow \phi^*$
6: Update selected parameters ϕ_u^* (Equation 6).
7: Update parameters $\phi^* \leftarrow \{\phi_u^*, \phi_f\}$.

3.4 Implementation

Hardware. To test various multi-camera setups in a controlled and reproducible environment, the multi-camera captures are emulated with a Linux laptop driven by (1) realistic multi-view video streams and (2) the encoding performance benchmarked for each frame independently on the same laptop, which avoids the interference of the encoding processes of different views. The Linux laptop is equipped with an NVIDIA Geforce GTX 1080 Ti GPU and an Intel Core i9 CPU clocked at 2.90GHz, which is connected to the cloud server through a WiFi connection. Each multi-view capture is emulated as a separate thread that continuously reads the one camera view from the disk at a real-time rate of 30 fps, replays the video encoding latency, and transmits the corresponding bitstream (encoded and saved during benchmarking) to the cloud server using WebRTC (Web Real-Time Communication) [40], a low-latency streaming framework. Our cloud server is configured on a Linux desktop featuring dual NVIDIA Geforce RTX 3090 Ti GPUs and an AMD Ryzen 9 CPU running at 4.95GHz. One GPU is dedicated to video decoding, while the other is for training. The server connects to the campus network via a high-speed 1 Gbps cable.

Software. We synchronize all cameras to the same global clock using Network Time Protocol [60] before streaming, assign a timestamp to each frame when generated at the edge, and align the received frames at the server based on the timestamp. The asymmetric codec design (§3.1) is generic to single-view neural video codecs that follow an inter-picture prediction workflow [2, 57, 65, 91]. We adopt components of single-view video codecs (*i.e.,* the encoder, motion/residual decoder, and warp modules) from scale space flow (SSF) [2], the SOTA open-source neural codec to the best of our knowledge. To demonstrate our consistency across different tradeoffs of quality and bitrate, we choose four SSF models for our experiments. Specifically, we adopt the models trained by CompressAI [6], with the quality level $\lambda = 1, 2, 3, 4$ (higher λ better quality). We configure the distance metric in the loss function (Equation 5) to MSE in both offline and online training for its wide adoption. The offline training (§3.3.1) is performed following the training procedure of SSF with a batch size of 8. The targeted bandwidth impact is set to 1% which ensures a minimal impact on the bandwidth. For the most recent cache, we set it to 500 video frames. The online training (§3.3.3) adopts the batch size of one to reduce the online GPU memory usage. We utilize the ADAM [46] algorithm and set the learning rate to 0.0001, following prior works [2, 57]. To accelerate online training, we do not decay the learning rate, which is commonly adopted to train a model offline. In offline neural codec training, we augment the training data with random cropping and flipping [2, 57, 65]. However, we do not apply such random transforms in order to better generalize the learned inter-view dependency to unseen videos in the same multiview scene. Every time the codec model decodes a new GOP, it will be synchronized with the one being trained, which incurs minimal overheads.

4 Evaluation

Our evaluation answers the following questions:

(1) What is ImmerScope's end-to-end performance? (§4.2)
(2) How does ImmerScope generalize? (§4.3)
(3) How do ImmerScope's designs save resources? (§4.4)
(4) What is the impact of our design decisions? (§4.5)

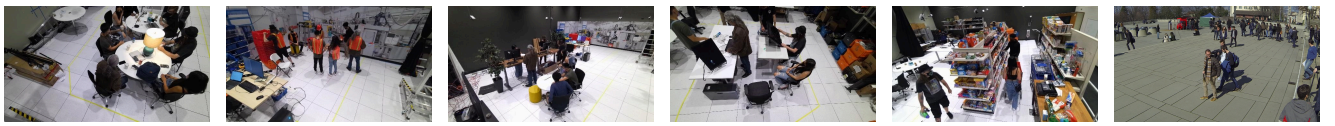


Figure 10: The name, number of views, and duration (minutes) of different scenes.

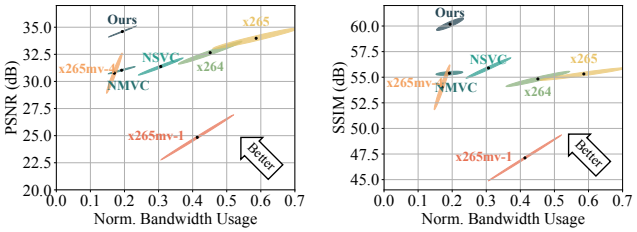


Figure 11: ImmerScope **saves BW by 64% in PSNR and 78% in SSIM than SOTA on WildTrack (high-res., 1 scene).**

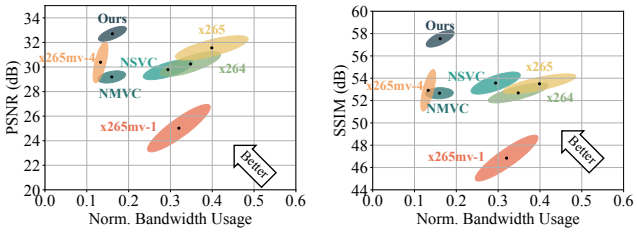


Figure 12: ImmerScope **saves BW by 63% in PSNR and 77% in SSIM than SOTA on MMPTRACK (low-res., 5 scenes).**

4.1 Methodology

Dataset. We aim to find multi-view video datasets with (1) multiple overlapped views, (2) frequent occurrences of moving objects, and (3) a reasonable duration for each scene to showcase the online learning benefits of ImmerScope. Based on these criteria, we choose two datasets. The first dataset is WildTrack [12], collected by seven GoPro cameras in front of the main building of ETH Zurich, Switzerland. Each view is of resolution 1920×1080 pixels, which lasts for roughly 35 minutes. The second dataset is Multi-camera Multiple People Tracking (MMPTRACK) dataset [32], which has around 6.5 hours of videos in total with a resolution of 360×640 pixels. All the videos are collected inside Microsoft indoor labs containing 5 scenes (*i.e.,* retail, lobby, office, industry, and cafe). Our datasets contain scenes with a diverse group of people, objects, and activities, varying numbers of views, and long durations, which are visualized in Figure 10. Although there are datasets captured with more cameras [50, 82], they are not included since their duration is too short (*i.e.,* ranging from tens to hundreds of seconds) to demonstrate the benefits with ImmerScope.

Baselines. We compare our approach to multi-view video streaming systems built with four categories of codecs.

- Traditional single-view codecs: H.264 [84] and H.265 [76] using the presets of `veryfast`, `medium`, or `veryslow`, which are denoted by x26n-p, where $n \in \{4, 5\}$ and $p \in \{\texttt{f}, \texttt{m}, \texttt{s}\}$.
- Traditional multi-view codecs: the SOTA multi-view codec MV-HEVC [77] at different coding unit depths, which are denoted by x265mv-{d}, where d is the coding unit depth. Other configurations are default.
- Neural single-view video codec (NSVC): the SOTA open-source neural video codec, scale space flow (SSF) [2].
- Neural multi-view video codec (NMVC): the codec with the design in §3.1 without online training. Existing works [17] focusing on two views are not applicable.

Following [2, 57], we disable B frames in all codecs and configure the same GOP size of 12 to ensure a fair comparison.

Metrics. We assess the performance of different systems in terms of video quality, bandwidth usage, and average bandwidth usage. We adopt two commonly used metrics for video quality, including peak-signal-to-noise ratio (PSNR) [34] and structural similarity index (SSIM) [83] calculated in the RGB colorspace like previous works [2, 65]. These quality metrics of the reconstructed video are computed using the raw video as the ground truth. They are averaged for each bitrate level and video, which is then used to make the oval plot. The reason for converting the SSIM metric to the unit of dB is to improve the visualization by making the oval spread more evenly on the plot. We normalize the bandwidth to avoid the impact of scenes and cameras in comparison, which is calculated by dividing the original value by the maximum value per dataset.

4.2 End-to-end Performance

Bandwidth efficiency. Figure 11 (left) compares the trade-offs of bandwidth and quality, measured in PSNR on the WildTrack dataset. ImmerScope significantly outperforms baselines with single-view video codecs (*i.e.,* x264, x265, and NSVC) with at least 64% savings in the average bandwidth usage. ImmerScope uses slightly more bandwidth (0.02 normalized bandwidth usage) than the SOTA multi-view video codec (x265mv-4) with significant quality improvement (3.8dB). A lightweight version of the multi-view video codec (x265mv-1) achieves the worst performance. Besides, ImmerScope outperforms NMVC by roughly 2dB, demonstrating the effectiveness of online learning. Figure 11 (right) shows more advantages of ImmerScope in SSIM on the MMPTRACK dataset, where ImmerScope demonstrates at least 78% average bandwidth savings than baselines with single-view video codecs. Compared to x265mv-4, ImmerScope achieves a much higher quality (4.9dB) with slightly more bandwidth usage (0.02 normalized bandwidth usage). Figure 12 (left) and Figure 12 (right) demonstrate the result on the MMPTrack dataset. The standard deviation of the result increases because of more scenes in this dataset. Nevertheless, ImmerScope outperforms single-view video codecs by 63% and 77% in terms of PSNR and SSIM, respectively. Meanwhile, ImmerScope drastically improves the quality of the SOTA multi-view video codec with minimal additional bandwidth usage. It's important to note that the quality level of our approach is adaptable by adjusting the parameter λ. While the average quality and bandwidth usage may not always be strictly better than other methods shown in Figure 11 and Figure 12,

Bo Chen[1], Hongpeng Guo[1], Mingyuan Wu[1], Zhe Yang[1], Zhisheng Yan[2], Klara Nahrstedt[1]
[1]University of Illinois at Urbana-Champaign, [2]George Mason University

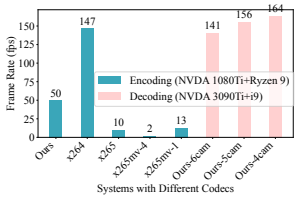


Figure 13: ImmerScope **attains real-time encoding and decoding frame rates.**

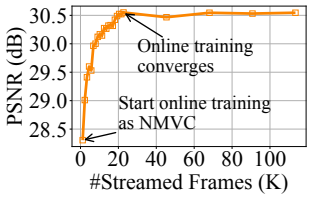


Figure 14: Quality improvement over time.

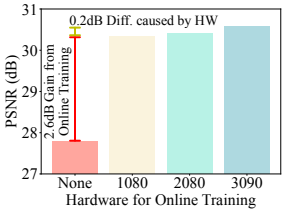


Figure 15: Quality gain is consistent on GPUs.

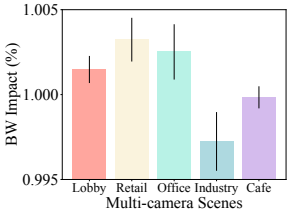


Figure 16: BW impact is consistently close to 1%.

we can fine-tune the quality level to outperform other approaches at any specific tradeoff between quality and bandwidth usage.

Frame rate. Figure 13 shows the average streaming rate of ImmerScope in encoding and decoding. ImmerScope achieves real-time encoding frame rates of 50 fps on the edge with NVIDIA GTX 1080Ti GPU where only x264 can outperform ImmerScope. As every edge always processes one view, its speed does not vary with the number of views. ImmerScope decodes at a speed of over 140 fps on the cloud, which is faster than encoding due to a more powerful GPU (NVIDIA RTX 3090Ti) than the edge and less complexity of decoding than encoding. The cloud processes all camera views centrally, whose speed is negatively affected by more views.

Training speed. Table 1 reports that offline training takes about 45 minutes on RTX 3090 per model, which is only performed once and further amortized by unlimited multi-view streaming scenes. It also shows the frame rate in online training for WildTrack [12] increases with the computation capability of NVIDIA Geforce GPUs (*i.e.,* NVIDIA Geforce RTX 3090 Ti, RTX 2080 Ti, and GTX 1080 Ti). The most powerful GPU attains the highest frame rate of 40 fps.

Table 1: The duration of offline processing and the rate of online optimization on different hardware.

Module	Offline Training	Online Training		
HW	3090	3090	2080	1080
Metrics	45 min	40 fps	26 fps	21 fps

Improvement over time. Figure 14 shows the video quality of ImmerScope ($\lambda = 1$) increases over time at the initial stage of online training. It plateaus after streaming around 20,000 frames (*i.e.,* 11 minutes and 20% of the total streaming duration) and reaches a quality gain of over 2dB. The achieved quality of roughly 30.5dB is a reasonable visual quality, which is only 1-2dB lower than the ideal quality (*i.e.,* the highest quality achieved when $\lambda = 1$ in Figure 23). Unfortunately, the ideal quality must be achieved with offline data collection using the same camera setup and training, which is impractical. This result demonstrates our approach swiftly adapts to a scene with good quality, considering the duration of a regular movie is 2-3 hours.

Improvement by online training. Figure 15 shows the video quality is at least improved by 2.6dB in PSNR with online training on different hardware when $\lambda = 1$. This observation highlights the significance of online training.

4.3 Generalizability

Hardware. Figure 15 reports how the quality improvement in PSNR ($\lambda = 1$) generalizes to GPUs on the cloud (*i.e.,* NVIDIA Geforce RTX 3090 Ti, RTX 2080 Ti, and GTX 1080 Ti) with fixed bitrate. The difference caused by GPUs is negligible (0.2dB) compared to the online training gain. By varying λ, we observe 2-3dB gains with online training and less than 0.5dB difference due to GPUs.

Scene. Figure 16 shows how ImmerScope generalizes to different scenes in MMPTRACK [32] by consistently maintaining an actual bandwidth impact (between 0.995% and 1.005%) close to the targeted value (1%). This is achieved by the streaming controller that adapts the temporal sampling interval accordingly, as shown in Figure 17. The reason that the temporal sampling interval negatively correlates with the bandwidth usage is that a higher bandwidth usage leads to a higher reconstructed video quality. As such, each frame difference contains less information (or entropy). Targeting a fixed bandwidth impact, a smaller sampling interval is therefore allowed to send more frames.

4.4 Resource Efficiency

Training data efficiency. Figure 18 reports that the adaptive training data streamer achieves a compression ratio (*i.e.,* the ratio of the size of compressed training data to the raw training data) of more than 85% compared to streaming the raw video as training data. There is an increase in the compression ratio when the bandwidth usage increases because the increased usage improves the video quality, which reduces the difference between the reconstructed and raw frames (*i.e.,* the entropy). As a result, entropy coding becomes more effective, boosting bandwidth saving.

Online training efficiency. The selective codec trainer saves bandwidth usage for updating neural encoders' parameters from 148 MB (for each encoder in every training iteration) to zero, as it eliminates the need to update the encoders. It reduces the compute resource (measured in the number of updated parameters in every training iteration) by 85% as we only optimize the inter-view residual decoder.

4.5 Rethinking Design Choices

Why temporal sampling? We reflect on whether we should adopt temporal sampling in the training data sampler (§3.2.1). Particularly, we compare two strategies to sample the training data with other configurations default: (1) *spatial sampling*, which uniformly samples 0.001% to 100% of pixels per frame, and (2) *temporal sampling*, which uniformly samples one frame from 1, 10, 100, and 1000 frames in raw video. ImmerScope easily adapts to a reduced number of frames. For a reduced number of pixels per frame, we need to reconstruct missing pixels to use the frames in training. To do so, we reconstruct the missing pixels with pixels at the same spatial and temporal location in the reconstructed frame received by the cloud. Figure 19(a) and Figure 19(b) show how the video quality changes

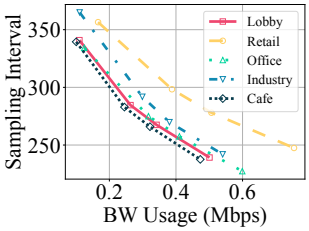


Figure 17: More BW use, more sampled frames.

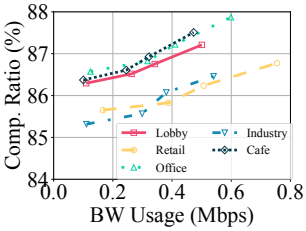
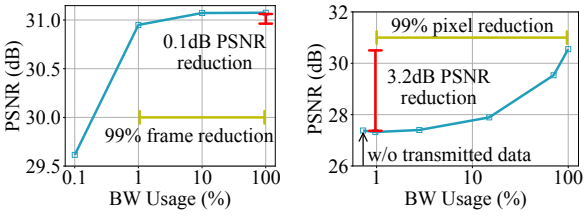


Figure 18: More BW use, higher compression ratio.



(a) Temporal sampling (b) Spatial sampling

Figure 19: Quality is resilient to temporal sampling.

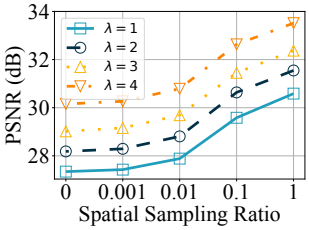


Figure 20: Spatial sampling degrades quality.

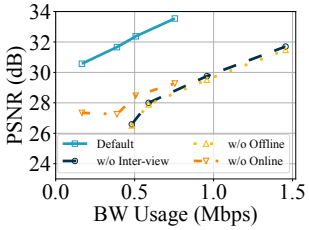


Figure 21: Impact of learning on ImmerScope.

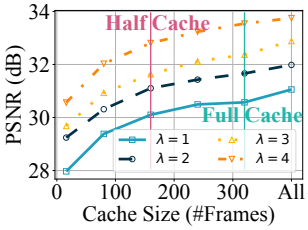


Figure 22: Video quality improves with cache size.

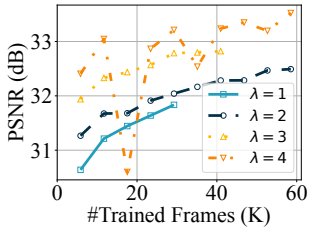


Figure 23: Prior knowledge boosts video quality.

when the bandwidth usage is altered by temporal and frame-level sampling, respectively. The video quality changes mildly when the bandwidth usage is reduced by temporal sampling but drastically with spatial sampling, which validates our design. For instance, when the bandwidth usage is reduced by 100× using temporal sampling, the quality degradation of the video is 0.1dB. However, if the same happens to spatial sampling, the quality degradation is 3.2dB, which is almost the same as online training without using transmitted data. This conclusion still holds for different values of λ, where temporal sampling degrades less than 0.2dB quality but spatial sampling reduces over 3dB quality. We conduct another experiment where the targeted bandwidth impact is fixed at 1% and the spatial sampling ratio is varied from 0 to 1. The sampling interval is automatically adapted to attain the targeted bandwidth impact. Figure 20 shows the reduction in the spatial sampling ratio results in a significant quality loss. When the sampling ratio is equal to or lower than 0.001, there is little difference (≤1dB) compared to using the reconstructed video for training. This result translates into one interesting finding: *given the same bandwidth consumption, sending multiple "partial" raw frames is no better than sending fewer "full" raw frames.*

Is learning important? Figure 21 compares the performance of the default ImmerScope, ImmerScope without inter-view dependency (§3.1.2), ImmerScope without offline training (§3.3.1), and ImmerScope without online training (§3.3.3). The results show that removing online training leads to a noticeable quality drop (1.8dB-4dB in PSNR), indicating that online learned inter-view correlations are critical for maintaining video quality. Additionally, the absence of inter-view dependency and offline training results in a 3.75× increase in bitrates compared to ImmerScope. Without inter-view dependency, online learning becomes less effective since the main benefits stem from learned inter-view relationships. The lack of offline training forces ImmerScope to learn both intra-view and inter-view dependencies online, which offers little improvement over approaches without online learning, highlighting the importance of offline training.

Is caching necessary? Figure 22 illustrates the impact of the most recent training cache with the cache size increasing from one GOP to all frames ("All"). The results show that the cache size of roughly 300 frames allows ImmerScope to achieve a comparable video quality (0.2-0.5dB less) as using all frames, which needs 25× the bandwidth used by ImmerScope for training. With a reduced cache size, there is a slight quality drop (*i.e.,* less than 0.75dB with size reduced by half). Even though the number of frames used in training is the same, the frames are temporarily closer to the current frame with a smaller cache size. Such a closeness makes the trained codec less generalized (or overfitted), which is the reason for the quality drop on average.

What if we have prior knowledge? Although we assume no prior knowledge of the scene before deployment, in certain cases, it is acceptable to offline fine-tune the system with videos recorded in a specific scene for better performance. To evaluate ImmerScope in this case, we divide the video in the "Retail" scene [32] into training and testing sets, containing 80% and 20% of the video, respectively. The fine-tuning proceeds similarly to online training except for the training data. To avoid an over-length training duration, the stopping (convergence) criteria is that the codec does not improve for three consecutive full passes of the training set. Figure 23 shows that fine-tuning with prior knowledge noticeably improves the codec by roughly 1dB for different λ. Fine-tuning takes a reasonable number of iterations (*i.e.,* 1.5-2× the number for online training) to converge. We could further trade computation for more quality gains by training over a longer duration with gradually reduced learning rates.

5 Discussion

Camera heterogeneity. ImmerScope can handle camera heterogeneity that does not alter the input format (*e.g.,* varying light conditions, camera positions, and intrinsic parameters) through

online learning, as the affected image data is transmitted to the server, allowing the codec to adapt accordingly. For light condition changes, another option is to compensate on the encoder side using hardware or software solutions, such as Auto-Exposure [71], White Balance [54], or on-camera denoising [101], depending on the processing capabilities of the camera or edge device. ImmerScope can address heterogeneity that affects input format (*e.g.,* resolution and frame rate) by naively upsampling or downsampling to a standard resolution and frame rate. For a more advanced approach, encoder architectures can be customized to accept different resolutions, and different decoders can be used to handle varying numbers of synchronized frames based on frame rate. If frame rates vary significantly across cameras, maintaining multiple decoders on the server may be impractical. In such cases, cameras with similar frame rates can be clustered and processed with a few decoders, while "outlier" cameras can be handled with off-the-shelf single-view codecs.

Customize for a specific end application. While ImmerScope is designed as a general-purpose framework for streaming multi-view videos from the edge to the cloud for immersive applications, it can be further improved by customizing it for specific applications. For example, the generic loss function (Equation 5) could be replaced with an application-specific loss function. One such application is the production of neural content representations [14, 43, 50, 59, 73, 82]. In this case, ImmerScope could be tailored by designing a loss function that maps decoder parameters to the rendering quality of the trained neural content. However, the design of such a loss function is beyond the scope of this paper.

Abrupt changes of camera states. The camera states (*e.g.,* positions, light conditions, and synchronization) may change during streaming. When a camera moves or lighting conditions shift, there may be a temporary drop in video quality as the new view dependencies no longer match the previously learned ones. However, video quality will gradually recover through online learning.

Handling random out-of-sync cameras is more challenging, as it requires adapting the decoder architecture in real-time to process a varying number of views without degrading quality. To address this, a progressive training procedure [11, 97] can be used to offline-train the decoder, enabling it to tolerate any loss of input views.

Impact of resolution requirements. The adoption of ImmerScope depends on resolution requirements. When resolution demands are low and bandwidth is not a constraint, users may opt for single-view codecs, which are faster and more energy-efficient. However, when higher resolution is needed and bandwidth becomes a bottleneck, ImmerScope must be used to maintain real-time streaming rates.

Hardware demand. One common challenge with neural video codecs is their reliance on GPUs. However, we demonstrate that even consumer-grade GPUs like the NVIDIA GTX 1080 Ti can run ImmerScope on the capture device at a real-time frame rate. Additionally, it is common for the cloud server to be equipped with high-end GPUs, which makes it less likely a computational bottleneck.

Rate adaptation. Rate adaptation with ImmerScope poses a unique challenge due to the inherent characteristic of most neural video codecs [2, 56, 57, 90, 91], where the bitrate of a specific codec model is fixed. If we switch models to achieve variable bitrates, we allocate the limited online training duration across various models, compromising the performance gain per model. While addressing this problem is beyond our scope, we propose two solutions. One is to cautiously select the number of bitrate levels that maximize coding efficiency considering the network condition and streaming duration. The other is to change the neural codec backbone to one that has multiple bitrate levels [65], which enables continuous optimization one model while supporting multiple bitrates.

6 Related Works

Video codec. Video coding standards, such as MPEG-2 [47], H.264 [84], and H.265 [76], are designed to support single-view videos. Later, multi-view extensions [16, 77, 81] are added to traditional video codecs (*e.g.,* H.264 and H.265) to provide better coding efficiency in multi-view videos by introducing inter-view dependency. However, these works suffer from problems of underutilized dependency and content distortion, which hampers their coding efficiency (§2.2). Another line of research focuses on end-to-end learned neural video codecs [2, 30, 57, 65, 93], which demonstrate better coding efficiency than traditional codecs. To extend neural video codecs to multi-view videos, there are efforts on neural stereo image compression [22, 52] and, more recently, neural stereo video compression [17]. In contrast, this paper focuses on a more challenging scenario with more than two views.

Video streaming enhancement. Enhancing the quality of video streaming has been the focus of numerous systems, typically classified into sender-based and receiver-based depending on where the enhancement is applied. The sender-based strategy analyzes playback statistics from clients and delivers videos of suitable bitrates to each client from a central server [8, 25, 42, 51]. One typical receiver-based strategy guides clients to download videos with appropriate bitrates from the server based on predicted bandwidth or buffer level [35–38, 58, 74, 89, 96, 100, 102]. Another receiver-based strategy, video super-resolution, is employed to enhance video streaming quality by applying super-resolution models to increase the resolution of downloaded segments [44, 95, 99] without incurring additional bandwidth usage. ImmerScope belongs to the receiver-based category. To the best of our knowledge, ImmerScope is the first multi-stream video aggregation approach based on neural multi-view video codecs.

7 Conclusion

We present ImmerScope, a multi-view video aggregation framework at the edge with a multi-view neural video codec. We overcome the underutilized dependency and content distortion problems of traditional multi-view video codecs with the edge-friendly neural multi-view codec design, the adaptive training data streamer, and the hybrid online training accelerator. Evaluation results showcase ImmerScope's bandwidth efficiency and real-time frame rates.

Acknowledgement

We thank our shepherd and the anonymous reviewers for their valuable feedback, which greatly improves this paper. This work was supported by NSF under the award numbers NSF OAC-1835834, NSF IIS-2140645, NSF CNS-2106592, NSF CCF-2217144, NSF CNS-1900875, NSF IIS-2140620, and NSF OAC-2144764.

References

[1] Adobe. 2012. Adobe's Real Time Messaging Protocol. Available at: https://rtmp.veriskope.com/pdf/rtmp_specification_1.0.pdf. (2012). Accessed on 2023.

[2] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. 2020. Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8503–8512.

[3] Xavier Alameda-Pineda, Jacopo Staiano, Ramanathan Subramanian, Ligia Batrinca, Elisa Ricci, Bruno Lepri, Oswald Lanz, and Nicu Sebe. 2015. Salsa: A novel dataset for multimodal group behavior analysis. *IEEE transactions on pattern analysis and machine intelligence* 38, 8 (2015), 1707–1720.

[4] Miko Atokari, Marko Viitanen, Alexandre Mercat, Emil Kattainen, and Jarno Vanne. 2019. Parallax-tolerant 360 live video stitcher. In *2019 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 1–4.

[5] Roberto G de A Azevedo, Neil Birkbeck, Francesca De Simone, Ivan Janatra, Balu Adsumilli, and Pascal Frossard. 2019. Visual distortions in 360° videos. *IEEE Transactions on Circuits and Systems for Video Technology* 30, 8 (2019), 2524–2537.

[6] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. 2020. CompressAI: a PyTorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029* (2020).

[7] Fabrice Bellard. 2018. BPG Image Format. (2018). https://bellard.org/bpg/

[8] Abdelhak Bentaleb, Ali C Begen, and Roger Zimmermann. 2016. SDNDASH: Improving QoE of HTTP adaptive streaming using software defined networking. In *Proceedings of the 24th ACM international conference on Multimedia*. 1296–1305.

[9] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. *Proceedings of COMPSTAT'2010* (2010), 177–186.

[10] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. 2020. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 86–1.

[11] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2019. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791* (2019).

[12] Tatjana Chavdarova, Pierre Baqué, Stéphane Bouquet, Andrii Maksai, Cijo Jose, Timur Bagautdinov, Louis Lettry, Pascal Fua, Luc Van Gool, and François Fleuret. 2018. Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5030–5039.

[13] Bo Chen, Mingyuan Wu, Hongpeng Guo, Zhisheng Yan, and Klara Nahrstedt. 2024. Vesper: Learning to Manage Uncertainty in Video Streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference*. 166–177.

[14] Bo Chen, Zhisheng Yan, Bo Han, and Klara Nahrstedt. 2024. {NeRFHub}: A Context-Aware NeRF Serving Framework for Mobile Immersive Applications. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*. 85–98.

[15] Bo Chen, Zhisheng Yan, Yinjie Zhang, Zhe Yang, and Klara Nahrstedt. 2024. {LiFteR}: Unleash Learned Codecs in Video Streaming with Loose Frame Referencing. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 533–548.

[16] Ying Chen, Ye-Kui Wang, Kemal Ugur, Miska M Hannuksela, Jani Lainema, and Moncef Gabbouj. 2008. The emerging MVC standard for 3D video services. *EURASIP Journal on Advances in Signal Processing* 2009 (2008), 1–13.

[17] Zhenghao Chen, Guo Lu, Zhihao Hu, Shan Liu, Wei Jiang, and Dong Xu. 2022. LSVC: A learning-based stereo video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6073–6082.

[18] Yihua Cheng, Ziyi Zhang, Hanchen Li, Anton Arapin, Yue Zhang, Qizheng Zhang, Yuhan Liu, Kuntai Du, Xu Zhang, Francis Y Yan, et al. 2024. {GRACE}:{Loss-Resilient}{Real-Time} Video through Neural Codecs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 509–531.

[19] Alexander Clemm, Maria Torres Vega, Hemanth Kumar Ravuri, Tim Wauters, and Filip De Turck. 2020. Toward Truly Immersive Holographic-type Communication: Challenges and Solutions. *IEEE Communications Magazine* 58, 1 (2020), 93–99. https://doi.org/10.1109/MCOM.001.1900272

[20] Abril Corona-Figueroa, Jonathan Frawley, Sam Bond-Taylor, Sarath Bethapudi, Hubert PH Shum, and Chris G Willcocks. 2022. Mednerf: Medical neural radiance fields for reconstructing 3d-aware ct-projections from a single x-ray. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 3843–3848. https://arxiv.org/abs/2202.01020

[21] Mallesham Dasari, Kumara Kahatapitiya, Samir R Das, Aruna Balasubramanian, and Dimitris Samaras. 2022. Swift: Adaptive Video Streaming with Layered Neural Codecs. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 103–118.

[22] Xin Deng, Wenzhe Yang, Ren Yang, Mai Xu, Enpeng Liu, Qianhan Feng, and Radu Timofte. 2021. Deep homography for efficient stereo image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1492–1501.

[23] FCC. 2023. Measuring Broadband Raw Data Releases - Fixed. (2023). https://www.fcc.gov/oet/mba/raw-data-releases

[24] Internet Engineering Task Force. 1998. Real Time Streaming Protocol (RTSP). Available at: https://datatracker.ietf.org/doc/html/rfc2326. (1998). Accessed on 2023.

[25] Aditya Ganjam, Faisal Siddiqui, Jibin Zhan, Xi Liu, Ion Stoica, Junchen Jiang, Vyas Sekar, and Hui Zhang. 2015. C3: Internet-scale control plane for video quality optimization. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*. 131–144.

[26] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. 2022. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379* (2022). https://arxiv.org/abs/2210.00379

[27] The Independent JPEG Group. 2014. libjpeg. (2014). https://github.com/LuaDist/libjpeg

[28] Yongjie Guan, Xueyu Hou, Nan Wu, Bo Han, and Tao Han. 2023. MetaStream: Live Volumetric Content Capture, Creation, Delivery, and Rendering in Real Time. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.

[29] Yu Guan, Chengyuan Zheng, Xinggong Zhang, Zongming Guo, and Junchen Jiang. 2019. Pano: Optimizing 360 video streaming with a better understanding of quality perception. In *Proceedings of the ACM Special Interest Group on Data Communication*. 394–407.

[30] Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. 2019. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7033–7042.

[31] Bo Han, Yu Liu, and Feng Qian. 2020. ViVo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th annual international conference on mobile computing and networking*. 1–13.

[32] Xiaotian Han, Quanzeng You, Chunyu Wang, Zhizheng Zhang, Peng Chu, Houdong Hu, Jiang Wang, and Zicheng Liu. 2021. MMPTRACK: Large-scale Densely Annotated Multi-camera Multiple People Tracking Benchmark. (2021). arXiv:cs.CV/2111.15157

[33] Jian He, Mubashir Adnan Qureshi, Lili Qiu, Jin Li, Feng Li, and Lei Han. 2018. Rubiks: Practical 360-Degree Streaming for Smartphones. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 482–494.

[34] Alain Hore and Djemel Ziou. 2010. Image quality metrics: PSNR vs. SSIM. In *2010 20th international conference on pattern recognition*. IEEE, 2366–2369.

[35] Tianchi Huang, Rui-Xiao Zhang, Chao Zhou, and Lifeng Sun. 2018. QARC: Video quality aware rate control for real-time video streaming based on deep reinforcement learning. In *Proceedings of the 26th ACM international conference on Multimedia*. 1208–1216.

[36] Tianchi Huang, Chao Zhou, Rui-Xiao Zhang, Chenglei Wu, and Lifeng Sun. 2023. Buffer awareness neural adaptive video streaming for avoiding extra buffer consumption. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[37] Tianchi Huang, Chao Zhou, Rui-Xiao Zhang, Chenglei Wu, Xin Yao, and Lifeng Sun. 2019. Comyco: Quality-aware adaptive video streaming via imitation learning. In *Proceedings of the 27th ACM international conference on multimedia*. 429–437.

[38] Tianchi Huang, Chao Zhou, Rui-Xiao Zhang, Chenglei Wu, Xin Yao, and Lifeng Sun. 2020. Stick: A harmonious fusion of buffer-based and learning-based approach for adaptive streaming. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1967–1976.

[39] David A Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE* 40, 9 (1952), 1098–1101.

[40] IETF/W3C. 2015. WebRTC. (2015). https://webrtc.org/

[41] ITU-T and ISO/IEC JTC 1. 1994. Generic coding of moving pictures and associated audio informationVPart 2: Video. *ITU-T Recommendation H.262 and ISO/IEC 13818-2 (MPEG-2 Video)* (1994).

[42] Junchen Jiang, Shijie Sun, Vyas Sekar, and Hui Zhang. 2017. Pytheas: Enabling data-driven quality of experience optimization using group-based exploration-exploitation. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*. 393–406.

[43] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics* 42, 4 (2023), 1–14.

[44] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. 2020. Neural-enhanced live streaming: Improving live video ingest via online learning. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 107–125.

[45] Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. 2018. Semi-amortized variational autoencoders. In *International Conference on*

Machine Learning. PMLR, 2678–2687.

[46] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[47] Didier J Le Gall. 1992. The MPEG video compression algorithm. *Signal Processing: Image Communication* 4, 2 (1992), 129–140.

[48] Kyungjin Lee, Juheon Yi, and Youngki Lee. 2023. Farfetchfusion: Towards fully mobile live 3d telepresence platform. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.

[49] Kyungjin Lee, Juheon Yi, Youngki Lee, Sunghyun Choi, and Young Min Kim. 2020. GROOT: a real-time streaming system of high-fidelity volumetric videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.

[50] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. 2022. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5521–5531.

[51] Xianshang Lin, Yunfei Ma, Junshao Zhang, Yao Cui, Jing Li, Shi Bai, Ziyue Zhang, Dennis Cai, Hongqiang Harry Liu, and Ming Zhang. 2022. GSO-simulcast: global stream orchestration in simulcast video conferencing systems. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 826–839.

[52] Jerry Liu, Shenlong Wang, and Raquel Urtasun. 2019. Dsic: Deep stereo image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3136–3145.

[53] Yu Liu, Bo Han, Feng Qian, Arvind Narayanan, and Zhi-Li Zhang. 2022. Vues: practical mobile volumetric video streaming through multiview transcoding. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 514–527.

[54] Yung-Cheng Liu, Wen-Hsin Chan, and Ye-Quang Chen. 1995. Automatic white balance for digital still camera. *IEEE Transactions on Consumer Electronics* 41, 3 (1995), 460–466.

[55] Jean loup Gailly and Mark Adler. 1995. zlib: A Massively Spiffy Yet Delicately Unobtrusive Compression Library. https://zlib.net. (1995). https://zlib.net Version 1.2.11.

[56] Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao. 2020. Content adaptive and error propagation aware deep video compression. In *European Conference on Computer Vision*. Springer, 456–472.

[57] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. 2019. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11006–11015.

[58] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 197–210.

[59] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

[60] David L Mills. 1992. Network Time Protocol (Version 3) Specification, Implementation and Analysis. *RFC 1305* (1992). https://datatracker.ietf.org/doc/html/rfc1305

[61] NVIDIA. 2023. Neural Radiance Fields: View Synthesis and 3D Reconstruction for 3D Online Shopping Experiences. https://www.nvidia.com/en-us/on-demand/session/gtcspring23-s51547/. (2023). Accessed: 2024-06-29.

[62] Albert Parra Pozo, Michael Toksvig, Terry Filiba Schrager, Joyce Hsu, Uday Mathur, Alexander Sorkine-Hornung, Rick Szeliski, and Brian Cabral. 2019. An integrated 6DoF video camera and system design. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–16.

[63] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10318–10327.

[64] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. 2018. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 99–114.

[65] Oren Rippel, Alexander G Anderson, Kedar Tatwawadi, Sanjay Nair, Craig Lytle, and Lubomir Bourdev. 2021. Elf-vc: Efficient learned flexible-rate video coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14479–14488.

[66] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. 2016. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*. Springer, 17–35.

[67] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 234–241.

[68] MICHAEL RUBLOFF. 2023. Google announces new Google Maps experience featuring Neural Radiance Fields (NeRFs). (2023). https://neuralradiancefields.io/google-announces-new-google-maps-experience-featuring-neural-radiance-fields/

[69] Michael Rubloff. 2024. Apple Pursuing Radiance Fields in Apple Maps. *Radiance Fields* (Jan 2024). https://radiancefields.com/apple-pursuing-radiance-fields-in-apple-maps

[70] F. Sener, D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, and A. Yao. [n. d.]. Assembly101: A Large-Scale Multi-View Video Dataset for Understanding Procedural Activities. *CVPR 2022* ([n. d.]).

[71] Suji Shimizu, Toshiharu Kondo, Takashi Kohashi, M Tsurata, and Teruyoshi Komuro. 1992. A new algorithm for exposure control based on fuzzy logic for video cameras. *IEEE Transactions on Consumer Electronics* 38, 3 (1992), 617–623.

[72] David Slepian and Jack Wolf. 1973. Noiseless coding of correlated information sources. *IEEE Transactions on information Theory* 19, 4 (1973), 471–480.

[73] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. 2023. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics* 29, 5 (2023), 2732–2742.

[74] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2020. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1698–1711.

[75] Thomas Stockhammer. 2011. Dynamic adaptive streaming over HTTP–: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 133–144.

[76] 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.

[77] Gerhard Tech, Ying Chen, Karsten Müller, Jens-Rainer Ohm, Anthony Vetro, and Ye-Kui Wang. 2015. Overview of the multiview and 3D extensions of high efficiency video coding. *IEEE Transactions on Circuits and Systems for Video Technology* 26, 1 (2015), 35–49.

[78] TaoXi Technology. 2021. 3D Modeling Techniques for Products Based on Neural Rendering. Available at: https://www.alibabacloud.com/blog/3d-modeling-techniques-for-products-based-on-neural-rendering_598327. (2021). Accessed on 2023.

[79] Jim Thacker. 2023. Use NeRFs in Unreal Engine with Luma AI's new plugin. Available at: https://www.cgchannel.com/2023/04/use-nerfs-in-unreal-engine-with-luma-ais-new-plugin/. (2023). Accessed on 2023.

[80] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[81] Anthony Vetro, Thomas Wiegand, and Gary J. Sullivan. 2011. Overview of the Stereo and Multiview Video Coding Extensions of the H.264/MPEG-4 AVC Standard. *Proc. IEEE* 99, 4 (2011), 626–642. https://doi.org/10.1109/JPROC.2010.2098830

[82] Liao Wang, Qiang Hu, Qihan He, Ziyu Wang, Jingyi Yu, Tinne Tuytelaars, Lan Xu, and Minye Wu. 2023. Neural Residual Radiance Fields for Streamably Free-Viewpoint Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[83] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. 2003. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Vol. 2. Ieee, 1398–1402.

[84] 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.

[85] Ian H Witten, Radford M Neal, and John G Cleary. 1987. Arithmetic coding for data compression. *Commun. ACM* 30, 6 (1987), 520–540.

[86] Jelmer M Wolterink, Jesse C Zwienenberg, and Christoph Brune. 2022. Implicit neural representations for deformable image registration. In *International Conference on Medical Imaging with Deep Learning*. PMLR, 1349–1359. https://proceedings.mlr.press/v172/wolterink22a.html

[87] Aaron Wyner and Jacob Ziv. 1976. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on information Theory* 22, 1 (1976), 1–10.

[88] Hongchi Xia, Zhi-Hao Lin, Wei-Chiu Ma, and Shenlong Wang. 2024. Video2Game: Real-time Interactive Realistic and Browser-Compatible Environment from a Single Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4578–4588.

[89] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Alexander Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming.. In *NSDI*, Vol. 20. 495–511.

[90] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. 2020. Learning for video compression with hierarchical quality and recurrent enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6628–6637.

[91] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. 2020. Learning for video compression with recurrent auto-encoder and recurrent probability model. *IEEE Journal of Selected Topics in Signal Processing* 15, 2 (2020), 388–401.

[92] Zhenyu Yang, Wanmin Wu, Klara Nahrstedt, Gregorij Kurillo, and Ruzena Bajcsy. 2010. Enabling multi-party 3d tele-immersive environments with viewcast. *ACM Transactions on Multimedia Computing, Communications, and Applications*

(TOMM) 6, 2 (2010), 1–30.

[93] Shuochao Yao, Jinyang Li, Dongxin Liu, Tianshi Wang, Shengzhong Liu, Huajie Shao, and Tarek Abdelzaher. 2020. Deep compressive offloading: speeding up neural network inference by trading edge computation for network latency. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 476–488.

[94] Abid Yaqoob, Ting Bi, and Gabriel-Miro Muntean. 2020. A Priority-aware DASH-based multi-view video streaming scheme over multiple channels. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 297–303.

[95] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural adaptive content-aware internet video delivery. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*. 645–661.

[96] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 325–338.

[97] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. 2018. Slimmable neural networks. *arXiv preprint arXiv:1812.08928* (2018).

[98] Dooyeol Yun and Kwangsue Chung. 2017. DASH-based multi-view video streaming system. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 8 (2017), 1974–1980.

[99] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. 2022. YuZu:Neural-Enhanced Volumetric Video Streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 137–154.

[100] Rui-Xiao Zhang, Tianchi Huang, Ming Ma, Haitian Pang, Xin Yao, Chenglei Wu, and Lifeng Sun. 2019. Enhancing the crowdsourced live streaming: a deep reinforcement learning approach. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 55–60.

[101] Rui-Xiao Zhang, Chaoyang Li, Chenglei Wu, Tianchi Huang, and Lifeng Sun. 2023. Owl: A pre-and post-processing framework for video analytics in low-light surroundings. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[102] Yuanxing Zhang, Pengyu Zhao, Kaigui Bian, Yunxin Liu, Lingyang Song, and Xiaoming Li. 2019. DRL360: 360-degree video streaming with deep reinforcement learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 1252–1260.