FV-NeRV: Neural Compression for Free Viewpoint Videos

Anonymous Author(s)
Affiliation
Address
email

Abstract

The delivery of free viewpoint videos (FVVs) is gaining popularity because of their ability to provide freely switchable perspectives to remote users as immersive experiences. While smooth view switching is crucial for enhancing user's experiences, FVV delivery faces a significant challenge in balancing traffic and decoding latency. The typical approach sends limited viewpoints and synthesizes the remainings on the user, reducing traffic, but increasing decoding delays. Alternatively, sending more viewpoints reduces the delay, but requires more bandwidth for transmission. In this paper, we propose a novel FVV representation format, FV-NeRV (Free Viewpoint-Neural Representation for Videos), to address this dilemma in FVV delivery. FV-NeRV reduces both traffic and decoding delay even for content with a large number of virtual viewpoints by overfitting compact neural networks to all viewpoints and pruning and quantizing the trained model. Experiments using FVVs show that FV-NeRV achieves a comparable or even superior traffic reduction with faster decoding speed compared to existing FVV codecs and NeRV formats.

1 Introduction

Free-viewpoint video (FVV) [1, 2] is an emerging technique that allows freely switchable viewing experience even with the limited number of physical cameras. For this purpose, FVV generates video frames from any desired viewpoint utilizing a limited set of texture and depth frames of multiple cameras and their positions [3, 4]. This technique enables us to create a new type of immersive experience in the field of, for example, entertainment [5], digital archive, medical imaging [6].

Ensuring seamless view-switching between viewpoints is vital for enhancing user experiences; thus, users naturally demand as many viewpoints as possible. However, it is not necessarily the case that all viewpoints desired by users are pre-recorded, so there arises a need to synthesize and transmit frames from viewpoints other than those actually recorded, using the frames from the recorded perspectives. Although many existing solutions focus on the generation of high quality free viewpoints [3, 7] from the limited number of physical views, we are still faced with a dilemma regarding the practical use of FVV: balancing traffic and decoding speed.

Depending on who actually handles the rendering of frames for the necessary viewpoints, we can consider sender-side rendering and user-side rendering. In the sender-side rendering, a content sender with rich computational resources synthesizes the video frames from all the viewpoints on demand in advance. We can conceal decoding time from the users, but sending pre-rendered video frames of all the viewpoints increases traffic proportionally. In contrast, in user-side rendering, the sender encodes video frames only from the physical viewpoints into a bitstream, distributes them to users, and commits the users to synthesize desired viewpoints locally as they want. In this way, we can avoid a rapid increase in traffic with an increase in the number of viewpoints. However, a long decoding delay caused by complex rendering operations, such as 3D warping and hole filling, prevents real-time

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

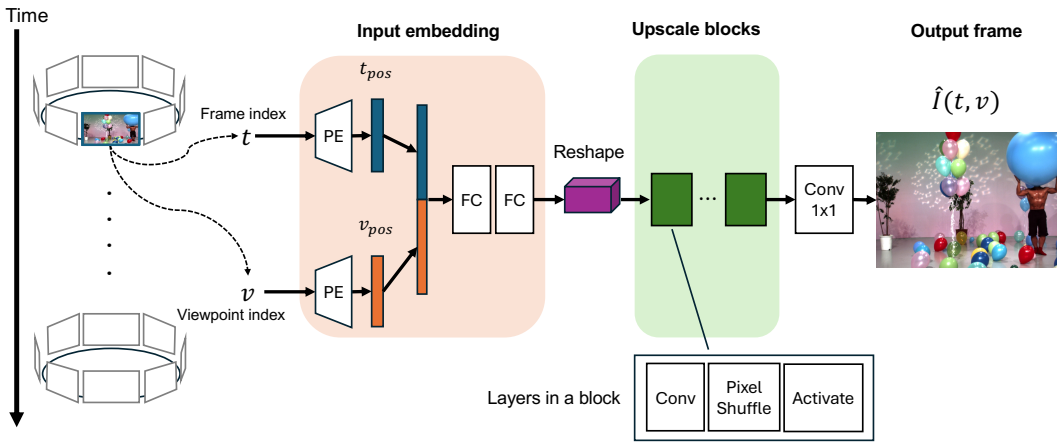


Figure 1: Overview of FV-NeRV

playback, that is, limited frame per second (fps), especially with the limited computational power on the user device.

In this paper, we propose **FV-NeRV**, a novel FVV format designed to effectively address the drawbacks of both the sender- and user-side rendering. This approach is heavily inspired by recent developments in frame-based implicit neural representation (INR) [8–16]. FV-NeRV operates on the principle of sender-side rendering while allowing all viewpoints to be transmitted to the user in an extremely lightweight manner, and this lightweight characteristic is achieved by leveraging INR techniques to overfit the transmitted content to a compact neural network. The technical contribution of FV-NeRV is that, unlike all frame-based INR derivatives, we make the compact neural network take viewpoint indices in addition to frame indices and overfit the video frames of the desired viewpoints to a single network at once. The network is capable of leveraging both multiview and temporal coherences during encoding, thereby effectively representing FVV across multiple viewpoints without a corresponding increase in model size, i.e., traffic. Moreover, the compact network benefits the user in terms of real-time FVV decoding by skipping compute-intensive rendering operations. Experiments on the FVV dataset show that the proposed FV-NeRV simultaneously outperforms existing FVV schemes in terms of traffic and decoding speed.

Related Work and Contributions

Recent INR architectures have been designed for image and video compression. Specifically, they send the overfitted weights of the INR architecture to the user side, and the user reconstructs video frames by feeding the coordinates and corresponding features. Existing studies have proposed the frame-wise INR architectures. Specifically, they feed the frame index and/or corresponding features to the INR architecture to generate a video frame. NeRV [8] is the first work of the INR architecture for frame-wise video reconstruction, and there are many extensions of NeRV architectures [9–18] for quality improvement.

Our FV-NeRV is the first study on the frame-wise INR architecture for low-delay and low-traffic FVV. Although some studies [19, 20] have designed the frame-wise INR architecture for multiview texture and depth videos, they considered user-side rendering and needed view synthesis operations for the reconstruction of the desired viewpoints. The proposed FV-NeRV is a novel approach for sender-side rendering, effectively averts traffic increase by overfitting single and compact neural networks to various viewpoints.

2 FV-NeRV

Fig. 1 shows the overview of the proposed FV-NeRV architecture. Let $\{I(t,v)\}_{t=1,v=1}^{T,V}$ be an FVV sequence consisting of an RGB video frame $I(t,v) \in \mathbb{R}^{H\times W\times 3}$ with T frames and V physical and virtual viewpoints. Here, $t, v \in [0,1]$ are normalized frame and viewpoint indices, and H and W

are the height and width of the video frame. The proposed FV-NeRV architecture can be defined by a mapping function f with learnable parameters $\boldsymbol{\theta}$ from the frame and viewpoint indices t, v to the corresponding video frame $I(t, v)$ as follows:

$$f : \mathbb{R}^2 \longrightarrow \mathbb{R}^{H \times W \times 3}. \tag{1}$$

The goal of the proposed FV-NeRV architecture is to obtain the indices-to-frame mapping for the video frame of each desired viewpoint. For this purpose, we obtain the optimal function $f(t, v; \boldsymbol{\theta}) \approx f$ via network training with learnable parameters $\boldsymbol{\theta}$ using the FVV sequence $\{I(t, v)\}_{t=1,v=1}^{T,V}$.

The trained parameters are then further compressed and sent to users as $\hat{\boldsymbol{\theta}}$. Once users receive the parameters, they can reconstruct the t-th video frames of the desired viewpoint v by feeding the corresponding indices to the FV-NeRV architecture $f(t, v; \hat{\boldsymbol{\theta}})$.

2.1 Model Architecture

FV-NeRV architecture $f(t, v; \boldsymbol{\theta})$ consists of a MLP and an upscaling module. The MLP part starts from positional embedding (PE) for both the frame and the viewpoint indices, which projects a single scaler onto a high-dimensional vector. The vector maintains the positional information of the index throughout the video sequence. As proposed in [8], we use a sinusoidal positional embedding with the basis b and level l as follows:

$$\begin{aligned} \mathbf{t}_{\text{pos}} &= \big(\sin(b^0 \pi t), \cos(b^0 \pi t), \cdots, \sin(b^{l-1} \pi t), \sin(b^{l-1} \pi t)\big) \in \mathbb{R}^{2l}, \\ \mathbf{v}_{\text{pos}} &= \big(\sin(b^0 \pi v), \cos(b^0 \pi v), \cdots, \sin(b^{l-1} \pi v), \sin(b^{l-1} \pi v)\big) \in \mathbb{R}^{2l}. \end{aligned} \tag{2}$$

These embeddings are concatenated $[\mathbf{t}_{\text{pos}}, \mathbf{v}_{\text{pos}}]$ and passed to the successive fully-connected (FC) layers. Finally, the output is reshaped to a 2D feature map $\mathbf{m} \in \mathbb{R}^{h_0 \times w_0 \times c}$, where h_0, w_0, c are the height, width, and channel.

The upscaling module is made up of L upscaling blocks, implemented using NeRV blocks [8], which gradually enhance the resolution of the feature map. Specifically, the l-th block first performs 2D convolution to increase channels by $h_{l-1} \times w_{l-1} \times c \cdot s_l^2$, and then 2D pixel shuffle to increase resolution by $h_{l-1} \cdot s_l \times w_{l-1} \cdot s_l \times c$, where s_l is the scaling factor for the l-th block. The feature map will have a resolution of $h \cdot s_1 \ldots s_L \times w \cdot s_1 \ldots s_L \times c$ after L upscaling blocks, and finally, the header layer of the 1×1 2D convolution projects the feature map to the final output with the resolution of $H \times W \times 3$ pixels.

2.2 Loss Function

To train the proposed FV-NeRV architecture, we integrate mean absolute error (MAE) and structural similarity (SSIM) losses as the following:

$$l = \frac{1}{T}\frac{1}{V} \sum_{t=1,v=1}^{T,V} \{\alpha \cdot \text{MAE}(f(t, v; \boldsymbol{\theta}), I(t, v)) + (1 - \alpha) \cdot (1 - \text{SSIM}(f(t, v; \boldsymbol{\theta}), I(t, v))\}, \tag{3}$$

where α is hyper-parameter to balance MAE and SSIM losses. Note that MAE represents the pixel loss averaged across the whole frame. Here, the MAE loss helps minimize the pixel-level distortions, whereas the SSIM loss reduces the perceptual distortion, e.g., blockwise distortion, during the training.

2.3 Model Compression

We introduce model compression for the overfitted FV-NeRV model to further reduce transmission and storage costs. Like existing INR models, the proposed FV-NeRV follows the model pruning, weight quantization, and weight encoding.

2.3.1 Model Pruning

Given the overfitted FV-NeRV model, global unstructured pruning is used to reduce the model size. Let $\boldsymbol{\theta}_q$ be the q-percentile value of all parameters $\boldsymbol{\theta}$. FV-NeRV sets weights with magnitudes below

$\boldsymbol{\theta}_q$ to zero as follows:

$$\hat{\boldsymbol{\theta}} = \begin{cases} \boldsymbol{\theta} & \boldsymbol{\theta} \geq \boldsymbol{\theta}_q \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

After pruning the model, the pruned FV-NeRV parameters $\hat{\boldsymbol{\theta}}$ are fine-tuned using the same dataset.

2.3.2 Model Quantization and Encoding

The fine-tuned parameters are then uniformly quantized with respect to the given bit depth N_b, followed by the entropy coding. Given a parameter tensor $\mu \in \hat{\boldsymbol{\theta}}$, the quantized parameter tensor μ_q is given as:

$$\mu_q = \text{round}\left(\frac{\mu - \mu_{\min}}{2^{N_b}}\right) * s + \mu_{\min},\ s = \frac{\mu_{\max} - \mu_{\min}}{2^{N_b}}, \tag{5}$$

where $\text{round}(\cdot)$ is a rounding function to the nearest integer and $\mu_{\max}$ and $\mu_{\min}$ are the maximum and minimum values for the parameter tensor μ. The quantized tensor μ_q is finally coded into binaries using entropy coding. FV-NeRV uses Huffman coding for binarization. Since the distribution of the tensor parameters μ_q tends to zero, especially at small bit depths, the Huffman coding further reduces the size of the model.

3 Experiments

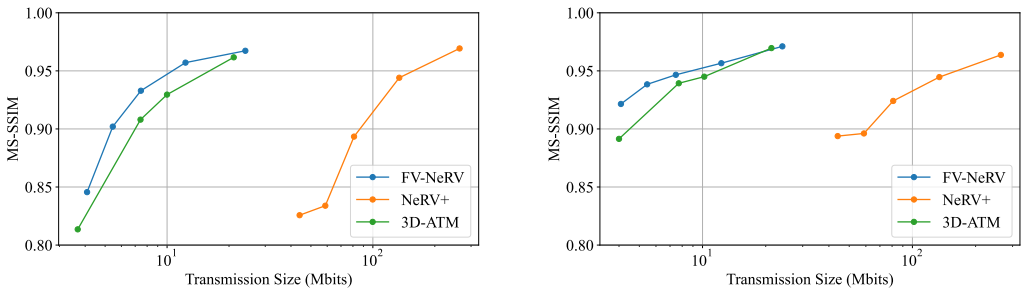
We evaluate the performance of our FV-NeRV with respect to the quality of decoded video sequences and decoding delay, using an FVV dataset obtained in the real world.

3.1 Settings

Dataset: We use a free-view TV dataset provided by Nagoya University [21] and specifically 2 FVV sessions "Balloons" and "Kendo" in the dataset. This dataset provides RGB frame sequences and depth image sequences for these two sessions. For both, the frame resolution is 768×1024 and the total number of frames T is 300. For every sequence, we choose 2 physical viewpoints, synthesize 9 virtual viewpoints in between, and utilize these 11-frame sequences as our own dataset for the experiments. We use High-Efficiency Video Coding (HEVC) test model (HTM) software renderer [22] for intermediate view synthesizing. The dispersion of camera positions is $10\,\text{cm}$ for the selected viewpoints and $1\,\text{cm}$ for the synthesized viewpoints. When frame indices or view indices are required, normalized indices $\{v|v = 0, 0.1, \cdots, 1\}$ and $\{t|t = 0, 1/300, 2/300, \cdots, 1\}$ are used.

Baselines: In our experiment, we consider two scenarios for how an FVV sequence is encoded, decoded, and transmitted and select the corresponding baseline for each as the competitive method for our FV-NeRV.

1. **User-side rendering.** The sender encodes the texture and depth image sequences for the selected viewpoints in a format and sends it to a user. The user first reconstructs the encoded texture and depth image sequences and takes charge of synthesizing RGB frame sequences for any other viewpoints as they want. As a baseline for this scenario, we use 3D AVC test model (3D-ATM) [23], an existing FVV codec that produces a bit stream from given texture and depth frame sequences. 3D-ATM employs delta encoding in both the temporal and geometric domains, accounting for motion and view disparity compensation.
2. **Sender-side rendering.** The sender uses an image renderer to synthesize all the frame sequences for intermediate views in advance and then sends all encoded for each viewpoint to a user. For this scenario, we use NeRV+, in which we simply train as many NeRV networks as the viewpoints to be sent. For a total of V viewpoints, both actually recorded and synthesized, NeRV+ prepares dependent networks $\{f(t; \boldsymbol{\theta}_i) : \mathbb{R} \to \mathbb{R}^{H \times W \times 3} \mid i = 1, \cdots, V\}$ and overfits each network to each view as single-view video encoding. After obtaining all V models, the user executes feedforward operations with these models by sequentially inputting frame indices and decoding the frame sequences.



(a) Balloons (b) Kendo

Figure 2: Average MS-SSIM index of the proposed and baseline schemes as a function of the total data size using (a) "Balloons" sequence and (b) "Kendo" sequence [2].

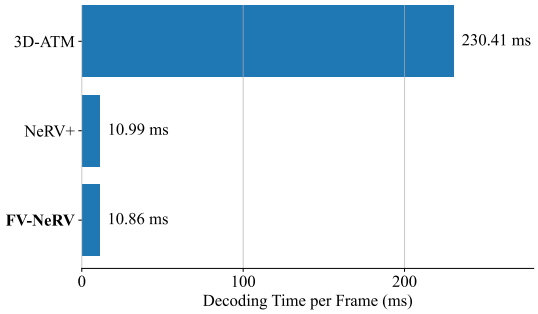


Figure 3: Average decoding time for each frame across "Balloons" and "Kendo" video sequences.

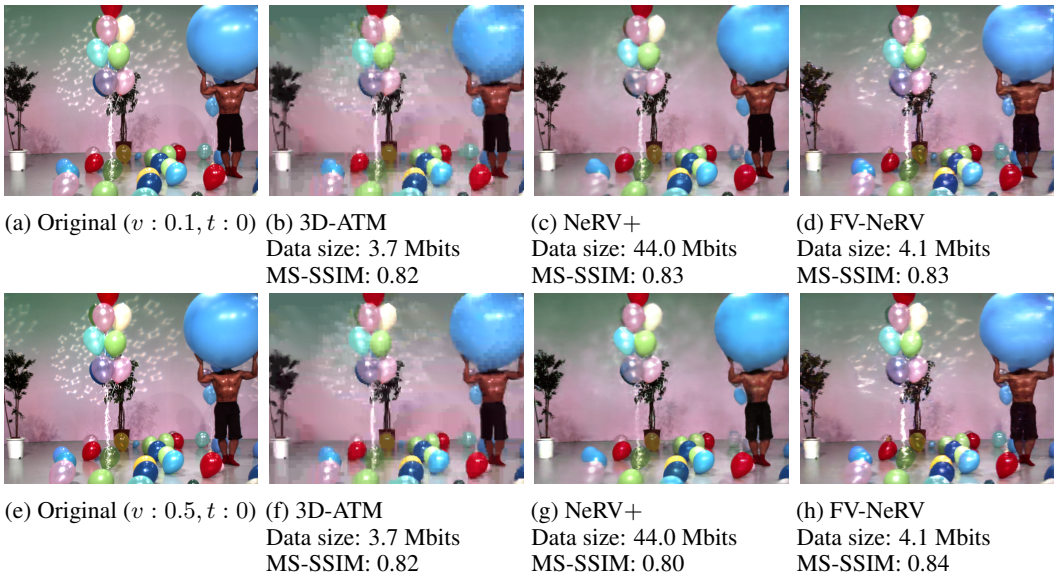
FV-NeRV and NeRV+ training: Models for the proposed FV-NeRV and NeRV+ are trained with the Adam optimizer with a learning rate of $5e-4$. We implement a cosine annealing learning rate schedule, incorporating a 10-epoch warm-up phase, with a batch size of 1 over a total of 50 training epochs. Following model pruning, we retrain the networks for an additional 50 epochs to fine-tune them. For the loss function in Eq. (3), α is set to 0.7.

Hardware and Software: All experiments are performed on a computer with Intel(R) Core(TM) i9-10850K CPU@3.60GHz, 128 GB memory, and NVIDIA RTX 3080 with 10 GB memory. The networks of NeRV+ and our FV-NeRV are implemented with Pytorch (2.2.0).

Model size control: We control the data size of the proposed and baseline schemes to evaluate their performance under varying compression levels. For 3D-ATM, we control different quantization parameters (QPs) for changing the size of the encoded bitstream. Here, an identical QP is used for the texture and depth video frames. Specifically, we use 33, 40, 43, and 50 QP for compression.

For NeRV+ and FV-NeRV, we control the size of the model by changing the number of network parameters and the degree of pruning and quantization. For both architectures, there are 5 upscale blocks, with up-scale factor 4, 2, 2, 2, 2 respectively for both "Balloons" and "Kendo" video sequences. In addition, we use $b = 1.25$ and $l = 40$ for input embedding in Eq. (2). For pruning and model quantization, we set $q = 40\%$ pruning ratio and $N_b = 8$ bit weight quantization according to the ablation study in [8].

Metrics: As the video quality metric, we use Multi-Scale Structural Similarity (MS-SSIM) [24] . MS-SSIM is computed for each pair of frames, yet it remains valuable for assessing the overall video quality aligned with human perception. The value ranges from 0 to 1 and a higher value close to 1 indicates a higher perceptual similarity between the original and decoded video frames. The decoding delay is measured by the average processing time required for a single frame reconstruction in the proposed FV-NeRV and NeRV+ and single-frame rendering in 3D-ATM.



(a) Original ($v : 0.1, t : 0$) (b) 3D-ATM
Data size: 3.7 Mbits
MS-SSIM: 0.82
(c) NeRV+
Data size: 44.0 Mbits
MS-SSIM: 0.83
(d) FV-NeRV
Data size: 4.1 Mbits
MS-SSIM: 0.83

(e) Original ($v : 0.5, t : 0$) (f) 3D-ATM
Data size: 3.7 Mbits
MS-SSIM: 0.82
(g) NeRV+
Data size: 44.0 Mbits
MS-SSIM: 0.80
(h) FV-NeRV
Data size: 4.1 Mbits
MS-SSIM: 0.84

Figure 4: Snapshot of "Balloons" in the baselines and the proposed FV-NeRV schemes.

3.2 Performance Evaluation

Figs. 2 (a) and (b) show the MS-SSIM index as a function of the size of the data in the proposed and baseline schemes using "Balloons" and "Kendo" video sequences, respectively. It shows that the proposed FV-NeRV achieves a better MS-SSIM index compared to the existing 3D-ATM, especially in narrow-band environments. Even though 3D-ATM is the user-side rendering and sends texture and depth frames of adjacent viewpoints, that of the data size is larger than that of the proposed FV-NeRV. NeRV+, which is the sender-side rendering, proportionally increases the size of the data as the number of physical and virtual viewpoints increases, since it sends the overfitted parameters of all viewpoints to the receiver.

Fig. 3 shows the average decoding delay of the proposed and baseline schemes in the video sequences of "Balloons" and "Kendo". Here, the data size of the proposed FV-NeRV and existing 3D-ATM is approximately 4.0 Mbits, while that of the NeRV+ is approximately 44.0 Mbits. The proposed FV-NeRV achieves the lowest decoding delay for video frame reco, comparable to the NeRV+ scheme, with a cheme with significantly small data size. 3D-ATM The 3D-ATM scheme needs to perform view synthesis to reconstruct video frames of the desired viewpoints, and such view synthesis operations cause more than 20 times longer decoding delays proposed FV-NeRV.

Figs. 4 show the snapshots of the original and reconstructed "Balloons" frames of the desired viewpoints in the proposed and baseline schemes. Here, the frame index t is 0, and the viewpoint indices v are 0.1 and 0.5, respectively. Furthermore, the data size of the proposed and the baselines is the same as in Fig. 3.

The snapshots in Figs. 4 (a) through (h) show that the proposed FV-NeRV reconstructs clean video frames regardless of the viewpoint positions. The reconstructed video frames in the 3D-ATM scheme are completely noisy due to large distortions in the texture and depth video frames. Although NeRV+ can reduce noise, it does not reconstruct white lights near the center balloons.

4 Conclusion

In this paper, we proposed FV-NeRV, a novel FVV representation that addresses the challenge of balancing traffic and decoding latency in FVV delivery. By overfitting compact neural networks to all viewpoints and reducing the model through pruning and quantization, FV-NeRV simultaneously reduces both traffic and decoding delay. Experiments demonstrated that FV-NeRV outperforms existing FVV codecs and NeRV, offering a more efficient solution for smooth view-switching.

References

[1] S. Guo, J. Hu, K. Zhou, J. Wang, L. Song, R. Xie, and W. Zhang, "Real-time free viewpoint video synthesis system based on dibr and a depth estimation network," *IEEE Transactions on Multimedia*, vol. 26, pp. 6701–6716, 2024.

[2] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo, "Free-viewpoint TV," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 67–76, 2011.

[3] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," in *Stereoscopic Displays and Virtual Reality Systems*, vol. 5291, may 2004, pp. 93–105.

[4] J. Artois, M. Courteaux, G. Wallendael, and P. Lambert, "OpenDIBR: Open real-time depth-image-based renderer of light field videos for VR," *Multimedia Tools and Applications*, vol. 83, pp. 25 797–25 815, 08 2023.

[5] I. Amar, A. Elsana, and J. El-Sana, "Live free-view video for soccer games," in *Proceedings of the 28th International ACM Conference on 3D Web Technology*, 2023.

[6] D. Kitaguchi, K. Kumano, R. Takatsuki, C. Xie, S. Hashimoto, Y. Akashi, I. Kitahara, and T. Oda, "Free-viewpoint video in open surgery: Development of surgical arena 360," *Journal of Surgical Education*, vol. 81, no. 3, pp. 326–329, 2024.

[7] O. Stankiewicz, G. Lafruit, and M. Domański, "Chapter 1 - multiview video: Acquisition, processing, compression, and virtual view rendering," in *Academic Press Library in Signal Processing, Volume 6*. Academic Press, 2018, pp. 3–74.

[8] H. Chen, B. He, H. Wang, Y. Ren, S.-N. Lim, and A. Shrivastava, "NeRV: Neural representations for videos," in *NeurIPS*, 2021.

[9] H. M. Kwan, G. Gao, F. Zhang, A. Gower, and D. Bull, "HiNeRV: Video compression with hierarchical encoding-based neural representation," in *NeurIPS*, 2023.

[10] J. C. Lee, D. Rho, J. H. Ko, and E. Park, "FFNeRV: Flow-guided frame-wise neural representations for videos," in *Proceedings of the ACM International Conference on Multimedia*, 2023, p. 7859–7870.

[11] B. He, X. Yang, H. Wang, Z. Wu, H. Chen, S. Huang, Y. Ren, S.-N. Lim, and A. Shrivastava, "Towards scalable neural representation for diverse videos," in *CVPR*, 2023.

[12] Y. Xu, X. Feng, F. Qin, R. Ge, Y. Peng, and C. Wang, "VQ-NeRV: A vector quantized neural representation for videos," *arXiv e-prints*, Mar. 2024.

[13] S. R. Maiya, S. Girish, M. Ehrlich, H. Wang, K. Lee, P. Poirson, P. Wu, C. Wang, and A. Shrivastava, "Nirvana: Neural implicit representations of videos with adaptive networks and autoregressive patch-wise modeling," in *CVPR*, jun 2023, pp. 14 378–14 387.

[14] R. Xue, J. Li, T. Chen, D. Ding, X. Cao, and Z. Ma, "NeRI: Implicit neural representation of lidar point cloud using range image sequence," in *ICASSP*, 2024, pp. 8020–8024.

[15] Y. Bai, C. Dong, C. Wang, and C. Yuan, "PS-NeRV: Patch-wise stylized neural representations for videos," in *IEEE International Conference on Image Processing (ICIP)*, 2023, pp. 41–45.

[16] H. Yan, Z. Ke, X. Zhou, T. Qiu, X. Shi, and D. Jiang, "DS-NeRV: Implicit neural video representation with decomposed static and dynamic codes," in *CVPR*, 2024, pp. 23 019–23 029.

[17] C. Gomes, R. Azevedo, and C. Schroers, "Video compression with entropy-constrained neural representations," in *CVPR*, 2023, pp. 18 497–18 506.

[18] H. Chen, M. Gwilliam, S.-N. Lim, and A. Shrivastava, "HNeRV: Neural representations for videos," in *CVPR*, 2023.

[19] H. M. Kwan, F. Zhang, A. Gower, and D. Bull, "Immersive video compression using implicit neural representations," *arXiv preprint arXiv:2402.01596*, 2024.

[20] C. Zhu, G. Lu, B. He, R. Xie, and L. Song, "Implicit-explicit integrated representations for multi-view video compression," *arXiv preprint arXiv:2311.17350*, 2023.

[21] T. Fujii, "Nagoya university sequences," 2013. [Online]. Available: http://www.fujii.nuee.nagoya-u.ac.jp/multiview-data/

[22] F. Bossen, D. Flynn, and K. Suhring, "HM software manual," 2014. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/HTM-13.0/

[23] A. Vetro, T. Ebrahimi, and V. Baroncini, "3D video subjective quality assessment test plan," Doc. JCT3V-F1011, Nov. 2013.

[24] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, vol. 2, 2003, pp. 1398–1402.

A Appendix

A.1 Video Representations in Dense Viewpoints and Different Video Sequences

We have prepared some results of video frame reconstruction as shown in Fig. 5 and 6 under the XS model size. In particular, in Fig. 6, we show the reconstructed video frames when the distance between the viewpoints is 1 cm. Although the camera arrangement is dense, the proposed FV-NeRV can reconstruct clean video frames of arbitrary viewpoints from the single compact model. In 3D-ATM, block noise occurs in the whole frame due to block-wise lossy operations for compression. The proposed FV-NeRV prevents block noise because the proposed upsampling blocks reconstruct the whole video frame at once. However, detailed visual information, such as the bamboo sword in each player, disappears when the size of the model is small.

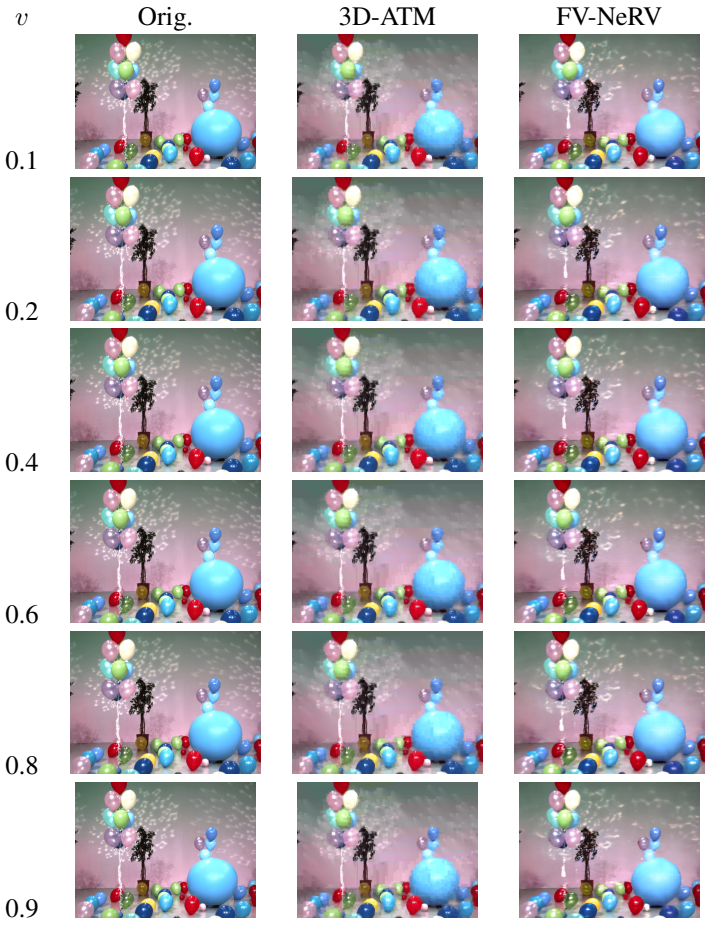


Figure 5: Reconstructed "Balloons" video frame at $t = 0.5$ of the XS-sized proposed and baseline schemes for different viewpoints v.

A.2 Training settings for NeRVs

Table 1 lists the parameter settings in the proposed FV-NeRV during training, pruning, and quantization.

Table 1: Parameter settings for our FV-NeRV.

	Name	Notation	Value
Positional Embedding	Basis	b	1.25
	Level	l	40
Training parameters	Batch size	B	1
	Epoch	-	50
	Epoch (fine-tune)	-	50
	Optimizer	-	Adam
	Learning rate	-	5e-4
	Learning rate scheduling	-	Cosine Annealing
	Loss function | weight factor	α	0.7
Model compression	Model pruning threshold	q	40%
	Model quantization depth	N_b	8

A.3 Detail Network Architectures for NeRVs

Table 2 shows the detailed network architecture of FV-NeRV. c_0 varies between different model sizes. SiLU stands for the activation function of the Sigmoid Linear Unit and is defined as:

$$\text{SiLU}(x) = x * \sigma(x),$$

where $\sigma(x)$ is the logistic sigmoid.

Table 2: Network architecture of FV-NeRV. Here, c_0 varies between different model sizes.

Group	**Layer**	**Output Shape**
Input		$[B, 2]$
Positional Embedding		$[B, 160]$
FC Layers	Linear	$[B, 512]$
	SiLU	-
	Linear	$[B, 12 \times 16 \times c_0]$
	SiLU	-
Reshape		$[B, c_0, 12, 16]$
Upsample Blocks	NeRV block 1	$[B, c_0, 48, 64]$
	NeRV block 2	$[B, 96, 96, 128]$
	NeRV block 3	$[B, 96, 192, 256]$
	NeRV block 4	$[B, 96, 384, 512]$
	NeRV block 5	$[B, 96, 768, 1024]$
Conv head		$[B, 3, 768, 1024]$
ith NeRV block	2D Convolution	$[B, c_{i-1} * s^2, h_{i-1}, w_{i-1}]$
	Pixel shuffling	$[B, c_i, h_i, w_i]$
	SiLU	-

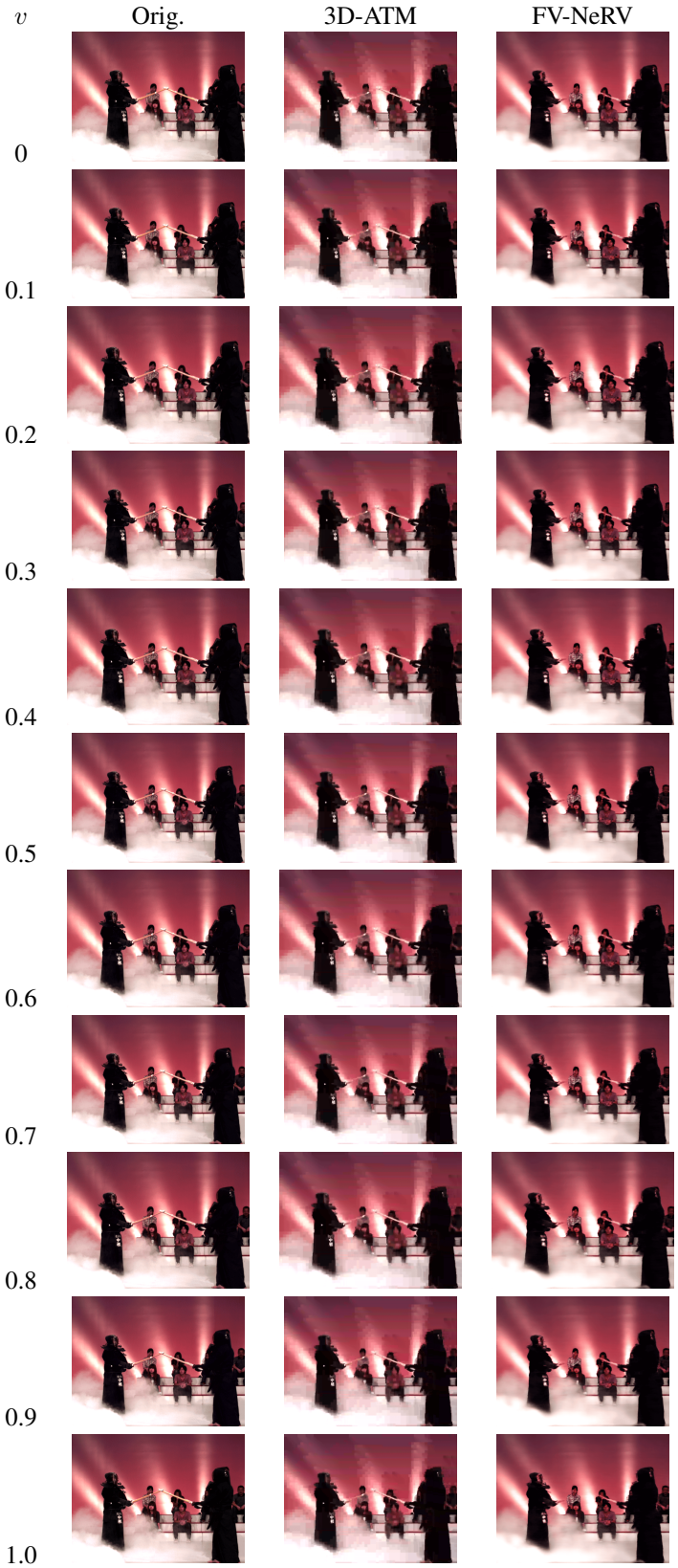


Figure 6: Reconstructed "Kendo" video frame at $t = 0.5$ of the XS-sized proposed and baseline schemes for different viewpoints v.