NeRFCompressor: Enhancing Dynamic Scene Representation for Efficient 6-DoF Object Transportation

Jin Zhou*, Mufeng Zhu†, Yao Liu†, and Songqing Chen*
*George Mason University, Fairfax, VA, USA †Rutgers University, Piscataway, NJ, USA

***Abstract*—3D scene modeling is essential for immersive experiences in Virtual, Augmented, and Mixed Reality (VR/AR/MR) applications. Neural Radiance Fields (NeRF) have emerged as a strong alternative to traditional representations such as meshes and point clouds for 6-DoF rendering. However, maintaining high visual quality while enabling efficient transmission in dynamic environments remains a significant challenge. In this paper, we propose *NeRFCompressor* , a novel compression framework for dynamic scene representation using NeRF-like models. Building on tensor decomposition-based 3D reconstruction, *NeRFCompressor* improves transmission efficiency by leveraging existing video codecs to exploit both intra-scene and inter-scene redundancies. It maintains high QoE with minimal degradation in reconstruction quality. Experiments show that *NeRFCompressor* outperforms state-of-the-art methods in compressing both static and dynamic scene representations.**

I. Introduction

Recent advances have enabled new applications in volumetric video, where content captured from multiple RGB-D cameras is stitched and streamed to devices like VR headsets and glasses. Companies such as Apple, Microsoft, and Intel are exploring use cases across sports, entertainment, e-commerce, and education, with the market projected to reach 4.9 billion USD by 2026 [1].

NeRF [13] provides an effective method for generating realistic 3D views from sparse inputs, but the training process is resource-intensive. Afterwards, TensoRF [5] enhances training efficiency using voxel grids and tensor decomposition, though at the cost of increased model size. As many applications target resource-limited mobile and AR/VR devices, compression becomes critical. Prior work [4], [6], [9], [15] addresses these constraints, and recent methods [10], [14], [17] further balance visual quality with compact model representation.

In this paper, motivated by recent advances in NeRF modeling [5], [22], we propose *NeRFCompressor* —a compression framework for representing dynamic scenes using NeRF-like models. *NeRFCompressor* provides a simple yet effective solution for compressing dynamic representations, enabling efficient 6-DoF exploration. Built upon TensoRF [5], *NeRFCompressor* observes both intra-scene and inter-scene similarities in tensor-decomposed model components. As these components can be represented as grayscale images, we encode them as video frames and apply standard video codecs for compression. *NeRFCompressor* incorporates three key techniques: (1) model quantization into 8-bit integers, (2) intra-scene redundancy exploitation, and (3) inter-scene similarity compression. We evaluate *NeRFCompressor* on both static and dynamic scenes using the Synthetic-NeRF, LLFF, and public dynamic datasets. Results demonstrate that *NeRFCompressor* significantly reduces model size while maintaining reconstruction quality, outperforming state-of-the-art compression schemes and preserving high Quality of Experience (QoE).

The remainder of the paper is organized as follows. We present background on NeRF, and follow-up works in Section II. We describe the design of *NeRFCompressor* in Section III. Section IV presents the experimental results. We make concluding remarks in Section V.

II. Background and Related Work

A. Neural Radiance Field (NeRF)

Neural radiance field modeling has recently been shown as a powerful tool for 3D scene reconstruction. In NeRF [13], rays are traced from the camera position ($\mathbf{o} = (x_o, y_o, z_o)$) through each pixel in the rendered image. If the direction of a ray is $\mathbf{d}$, samples on this ray can be denoted as $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, and the color of the ray with N samples can be obtained as $\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i$, where $T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right)$ where T_i is the transmittance at sample i, σ_i and $\mathbf{c}_i$ are the density and color, and $\delta_i = t_i - t_{i-1}$ is the sample interval. While NeRF achieves high-fidelity rendering, it requires known camera poses and extensive training time. NeRF– [22] relaxes the camera pose dependency, while TensoRF [5] accelerates training using tensor decomposition. Other works like Mega-NeRF [19] and DVGO [16] further improve speed. Our method, *NeRFCompressor* , builds upon TensoRF [5]. Unlike NeRF's MLP-based mapping from coordinates to color/density, TensoRF decomposes the 4D volumetric field into compact axis-aligned vector and matrix components, separately modeling density and appearance. The density at a point is computed via combinations of learned 1D vectors and 2D matrices. For appearance, each voxel stores a multi-channel feature generated similarly, which is then combined with view direction via a lightweight MLP to produce RGB output. This low-rank tensor factorization reduces memory overhead while preserving fine details, enabling efficient training and fast rendering—ideal for both static and dynamic 3D scenes.

B. Compression and Quantization in NeRF

To improve the efficiency of NeRF-like models, several recent works have investigated quantization and compression techniques. VQRF [10] introduces a vector-quantized radiance field framework that compresses DVGO [16] to 1MB and TensoRF [5] to an average of 3.6MB while maintaining a PSNR of 32.86 dB on the Synthetic-NeRF dataset [13]. Within their framework, voxel pruning, vector quantization via a codebook, and weight quantization are applied, with a particular focus on compressing the plane components of radiance fields. Rho et al. [14] proposed an alternative method based on wavelet transforms and learnable masking to enable efficient compression of grid-based radiance fields. Expanding NeRF deployment to resource-constrained devices such as AR/VR headsets and mobile platforms has also been a key focus. MobileNeRF [6] adopts polygon rasterization with Z-buffering to facilitate fast and parallel rendering, while Li et al. [9] propose RT-NeRF, an algorithm–hardware co-design targeting real-time rendering on AR/VR devices. These approaches address both computational and energy limitations of modern devices.

In our proposed framework *NeRFCompressor* , we integrate vector quantization techniques with video compression methods within the TensoRF structure to support both static and dynamic scenes. By leveraging low-bit quantization and efficient video codec pipelines, *NeRFCompressor* achieves compact model storage while preserving high rendering quality, providing a scalable and effective solution for bandwidth-sensitive and performance-constrained applications.

C. Neural Radiance Field for Dynamic Scenes

Several works have extended NeRF to dynamic scenes, where the scene content evolves over time. Li et al. [11] proposed DyNeRF, which models dynamic volumetric scenes by conditioning NeRF on time across multi-view video inputs. More recently, Fridovich-Keil et al. [7] proposed K-Planes, a method that represents dynamic scenes using six decomposed planes in a 4D (space + time) radiance field. K-Planes can be seen as an extension of TensoRF that introduces temporal modeling. Similarly, HexPlanes [3] provides a concurrent decomposition-based approach for dynamic scene encoding. While methods like K-Planes offer promising PSNR performance, we observe visual artifacts such as ghosting and cloudy textures in practical settings. To address this, *NeRFCompressor* adopts a per-scene training strategy: we treat each dynamic frame as an independent TensoRF model and then encode the component planes across frames using traditional 2D video codecs. This enables us to exploit both inter-frame and intra-model redundancies for compression, while avoiding temporal artifacts and ensuring robust per-frame reconstruction quality in dynamic 3D content.

III. Design of *NeRFCompressor*

The design of *NeRFCompressor* aims to efficiently compress dynamic scenes via the neural radiance fields. We draw the analogy between dynamic scenes and videos, emphasizing the connection that exists between consecutive scenes within the dynamic scene and the successive frames within a video. As existing video codecs exploit intra-frame and inter-frame similarities for compression, we propose *NeRFCompressor* to exploit similarities among decomposed components within a single scene (intra-scene compression) and similarities between consecutive scenes (inter-scene compression) for a more compact representation of dynamic scenes. Figure 1 illustrates the overall design of *NeRFCompressor* . We present the details of each component next.

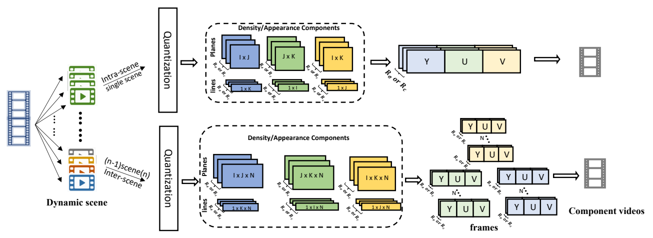

Fig. 1: **Overall design of *NeRFCompressor* .** The top part sketches our proposed idea on intra-scene compression that can be used for static scenes. The bottom part illustrates our proposed idea for inter-scene compression that can be used for dynamic scenes.

A. Model Compression via Quantization

The first stage of *NeRFCompressor* is feature quantization, commonly used for reducing the size of machine learning models. In an example configuration, a TensoRF model contains 16 components in each density plane and line (i.e., $R_\sigma = 16$) and 48 components in each color plane and line (i.e., $R_c = 48$), with a total of 192 components in the trained model. To exploit model compression via quantization, we first normalize the data stored in the $3R_\sigma + 3R_c$ "planes" and "lines" as follows:

$$d_{norm} = (d - d_{min})/(d_{max} - d_{min}) \quad (1)$$

We then quantize the data to integers in $[0, 255]$. The quantization step reduces model size by converting 32-bit floating-point data to 8-bit integers. Furthermore, this also makes it possible for us to represent the "planes" and "lines" as grayscale images and video frames and leverage existing image/video codecs for efficient encoding in later stages. We store the d_{min} and d_{max} in Equation 1 so that the data can be re-scaled back to its original float32 representation with minor precision loss.

B. Intra-Scene Compression via Similarities Among Decomposed Neural Components

As we inspect the planes and lines in grayscale images, we observed visual similarities among images of the same dimension, e.g., between $M_\sigma^{(r_m,yz)}$ and $M_\sigma^{(r_n,yz)}$, where r_m and r_n index two different components belonging to the yz plane. This motivates us to compress the model representing a single static scene by exploiting visual similarities among decomposed neural components. To do so, we represent components as video frames and use existing 2D video codecs for compression. We refer to this scheme as "Intra-Scene Compression".

Existing video codecs (e.g., H.264 [23]) work well for video frames represented in Y, U, V – three channels. Given that we have three orthogonal planes (i.e., xy, yz, and xz) and three orthogonal lines (i.e., z, x, and y), we composite a video frame using each of the three planes/lines as one of the three channels (Y, U, and V). In this way, with a total of $3R_\sigma + 3R_c$ components, we obtain $R_\sigma + R_c$ frames representing planes and $R_\sigma + R_c$ frames (with the height of just 1), representing lines. A benefit of this video frame composition is that all components of the same plane are compressed as the same channel (e.g., Y-channel) of consecutive video frames. This allows the video codecs to exploit similarities across components of the same plane for efficient compression. The three orthogonal planes—xy ($I \times J$), yz ($J \times K$), and xz ($I \times K$)—have mismatched dimensions, making direct composition into a single video frame infeasible. To resolve this, we resize all planes to a unified resolution of $n \times n$, where $n = \max(I, J, K)$, and represent line components as $1 \times n$, applying zero-padding as needed for consistency. While typical video codecs use `YUV420` to exploit human sensitivity to luminance by subsampling color channels, this degrades precision for model data. Instead, we adopt the `YUV444` [2] format, preserving full resolution across all channels to maintain model fidelity during compression.

In detail, our proposed intra-scene compression works as follows: ① We first extract all the components from the model trained by TensoRF. ② We then follow the normalization and quantization steps in Section III-A to represent the model components as 8-bit grayscale images. For density components, there are $3R_\sigma$ components for planes and $3R_\sigma$ components for lines, and for appearance components, there are $3R_c$ components for planes and $3R_c$ components for lines. In step ③, *NeRFCompressor* composites $3R_\sigma$/$3R_c$ components into R_σ/R_c frames in `YUV444` format. Then, in step ④, *NeRFCompressor* uses the video codec to compress these video frames, e.g., using FFmpeg [18] with H.264 [23]. As shown in the figure, the final output includes four compressed videos representing the density planes, density lines, appearance planes, and appearance lines, respectively.

C. Inter-Scene Compression via Similarities Across Consecutive Scenes

To represent dynamic scenes, drawing inspiration from the concept of inter-frame compression of consecutive frames in traditional 2D video codecs, we exploit similarities across consecutive scenes to achieve inter-scene compression. Here, we assume that the dynamic scene contains a sequence of scenes of consecutive time instants; motions in the scene cause items to move over a range of positions and that the changes in position are continuous and relatively small in consecutive scenes. This inter-scene compression scheme complements the intra-scene compression discussed in the previous subsection.

The high-level idea is as follows: suppose there are N consecutive scenes in the dynamic sequence of scenes, we first use TensoRF for training representations for individual scenes. We then composite the $N \times (3R_\sigma + 3R_c)$ trained components as $N \times (R_\sigma + R_c)$ frames (each in `YUV444` pixel format in Y, U, V – 3 channels) for video compression. We show the procedure of model quantization and `YUV444` frame composition in Figure 2. To improve the compression ratio, we design two training strategies for increasing the similarities among trained components of consecutive frames. The first approach always uses the trained components of the 1st scene to initialize the models (i.e., components) for the remaining scenes in the sequence. We expect this approach to work well in scenarios where the position changes are relatively small across the dynamic sequence. The second approach always uses the trained components of the previous scene (e.g., scene $s-1$) to initialize the models for scene s. With this approach, we expect the trained components of consecutive scenes to have increased similarity. For merging composited frames of a sequence of scenes into a single video for compression, we concatenate the frames in naïve manner while preserving the sequence within the dynamic scene. Finally, the concatenated frames are compressed as a video using traditional video codecs.

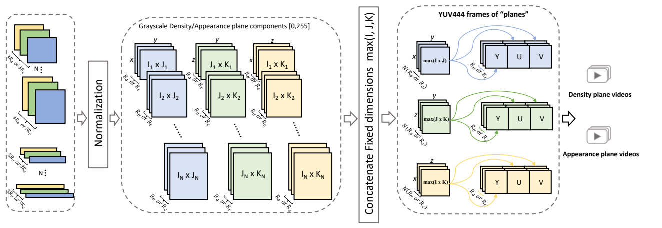

Fig. 2: **Inter-scene Quantization.** It shows the quantization steps for inter-scene compression.

In this paper, we choose H.264 for its commendable performance in scenarios involving transmission and processing. However, we note that our *NeRFCompressor* approach is compatible with other video codecs as well. Naïvely concatenating feature frames can result in poor compression efficiency, and inefficient utilization of storage and computing resources. Instead, we propose two strategies to improve compression efficiency by optimizing the arrangement of feature frames. The first approach is called "*NeRFCompressor*$_{SF}$", *NeRFCompressor* with Sequential features Fusion. For each feature component plane, "*NeRFCompressor*$_{SF}$" segments scene features into three groups based on the total number of scenes. Each group will be one of the three channels (YUV) in the video. For example, each scene contains R_σ features for each density component plane, so R_σ frames are generated for each scene. Overall, the total number of frames for each feature video is $(R_c \times N/3)$ or $(R_\sigma \times N/3)$. Lastly, "*NeRFCompressor*$_{SF}$" generates three density component videos and three appearance component videos. The format of frames structure is as follows:

$$\mathcal{V}_{R_c} : \{\mathcal{F}^{(1)}_{R_c}; \mathcal{F}^{(2)}_{R_c}; \cdots; \mathcal{F}^{(N/3)}_{R_c}\} \ldots \{\mathcal{F}^{(2N/3)}_{R_c}; \mathcal{F}^{(2N/3+1)}_{R_c}; \cdots; \mathcal{F}^{(N)}_{R_c}\}$$
$$\mathcal{V}_{R_\sigma} : \{\mathcal{F}^{(1)}_{R_\sigma}; \mathcal{F}^{(2)}_{R_\sigma}; \cdots; \mathcal{F}^{(N/3)}_{R_\sigma}\} \ldots \{\mathcal{F}^{(2N/3)}_{R_\sigma}; \mathcal{F}^{(2N/3+1)}_{R_\sigma}; \cdots; \mathcal{F}^{(N)}_{R_\sigma}\}$$

where $\mathcal{V}_{R_c}$ and $\mathcal{V}_{R_\sigma}$ represent density and component videos. $\mathcal{F}^{(1)}_{R_c}$ represents R_c frames of one of the appearance component planes from scene 1. *NeRFCompressor*$_{SF}$ can achieve a high compression rate with lossless re-rendering quality. We also design another method for compositing and merging frames into component videos to maximize feature similarities and express a more efficient compression rate. This method utilizes the spatial position for frame composition, called "*NeRFCompressor*$_{SA}$"(*NeRFCompressor* with Spatial Alignment). *NeRFCompressor*$_{SA}$ considered the features between the scenes with similar spatial orientations. Thus, we choose to merge the features from the same position of each scene into a sequence and composite them into frames. Because the camera positions remain consistent across each scene, the features generated from our second strategy of each scene manifest spatial similarity. Hence, the composition method used in *NeRFCompressor*$_{SA}$ would utilize the similarity between scenes to achieve inter-scene compression and avoid codec constraints. We describe "*NeRFCompressor*$_{SA}$" as:

$$\begin{aligned}\mathcal{V}_{R_c} &: \{\mathcal{F}^{(1)}_{[1]};\mathcal{F}^{(2)}_{[1]};\cdots;\mathcal{F}^{(N)}_{[1]}\}\ldots\{\mathcal{F}^{(1)}_{[R_c]};\mathcal{F}^{(2)}_{[R_c]};\cdots;\mathcal{F}^{(N)}_{[R_c]}\} \\ \mathcal{V}_{R_\sigma} &: \{\mathcal{F}^{(1)}_{[1]};\mathcal{F}^{(2)}_{[1]};\cdots;\mathcal{F}^{(N)}_{[1]}\}\ldots\{\mathcal{F}^{(1)}_{[R_\sigma]};\mathcal{F}^{(2)}_{[R_\sigma]};\cdots;\mathcal{F}^{(N)}_{[R_\sigma]}\},\end{aligned}$$ where

$\mathcal{V}_{R_c}$ represents the appearance component videos, and $\mathcal{V}_{R_\sigma}$ represents the density component videos. $\mathcal{F}^{(1)}_{R_c}$ stands for the frame at position R_c of scene 1 of the appearance component plane. For example, after extracting feature images from the same position, merge them into a sequence and composite them into frames with other sequences. Finally, R_c density component videos and R_σ appearance component videos are generated while each video contains N frames. This method can leverage the higher similarity between scenes but will result in a greater loss of re-rendering qualities than *NeRFCompressor*$_{SF}$. Consequently, there is a trade-off between *NeRFCompressor*$_{SF}$ and *NeRFCompressor*$_{SA}$. While the total number of scenes exceeds the codec limitation, we can choose *NeRFCompressor*$_{SA}$ to composite component feature frames. Otherwise, when higher re-rendering quality is the prerequisite, *NeRFCompressor*$_{SF}$ can be the prior method.

IV. Performance Evaluation

For evaluation, we use both static and dynamic scene datasets. We demonstrate the performance of *NeRFCompressor* by evaluating and comparing the PSNR of rendered views and the model size of *NeRFCompressor* with state-of-the-art methods. For static scenes, besides NeRF [13] and TensoRF [5], we also compare with VQ-TensoRF by Li et al. [10]. For dynamic scenes, we note that existing NeRF-based dynamic scene representations suffer from visual impairments such as ghosting effects and cloudy artifacts. The evaluation metrics used in the experiments focus on visual quality and compressed size. To evaluate the re-rendering performance, we apply *NeRFCompressor* to the trained TensoRF models for both static and dynamic scenes and compute the PSNR [8] of rendered views with ground-truth views rendered from the original 3D models.

TABLE I: **PSNR(dB) results of *NeRFCompressor* .**In this table, "*NeRFCompressor_xxx*" represents the quantized model with varying numbers of components.

	lego	chair	hotdog	drum
TensoRF (VM-192)	36.46	35.76	37.41	26.01
TensoRF-jpeg10	34.70	33.89	35.69	24.33
TensoRF-jpeg60	28.51	28.11	30.42	19.84
*NeRFCompressor*_192	36.59	35.70	37.65	25.95
*NeRFCompressor*_120	36.47	35.61	37.51	25.73
*NeRFCompressor*_84	36.17	35.48	37.20	25.55
*NeRFCompressor*_48	35.72	35.09	36.95	25.47

A. Experimental settings and Datasets

The model is implemented with PyTorch, and we used FFmpeg for video codec-based compression. We compared our proposed model with baselines in two static datasets and two dynamic datasets. For static datasets, we use NeRF synthetic dataset [13] and LLFF forward-facing dataset [12]. For dynamic datasets, we adopt the public synthetic dynamic dataset [25], including Lego, Pig, Worker, and Amily.

The baseline model we used is TensoRF VM-192, where $R_\sigma = 16$ and $R_c = 48$, i.e., $3R_\sigma + 3R_c = 192$. We applied our proposed model to the baseline model. Within our experiments, we also implemented our proposed quantization methods on VM-120, VM-84, and VM-48 for static scenes. We compared the rendering performance and model size with baseline models, including NeRF, TensoRF, and VQ-TensoRF. All the experiments are performed on a single NIVIDA P100 GPU with 16 GB memory. The experiment settings are similar as these of TensoRF. We trained the model with 30k iterations, while the batch size of each iteration is 4096. The initial learning rate and MLP-related parameters are 0.02 and 0.001. We used PSNR, SSIM [21], and LPIPS [24] to measure the reconstruction quality, and the model size to measure compression efficiency.

B. Evaluation Results

We present a comprehensive evaluation of our proposed *NeRFCompressor* in this section. We focus on the efficacy of our techniques designed to compress both static and dynamic scene scenarios, encompassing both intra-scene and inter-scene complexities, ensuring efficient compression while maintaining high Quality of Experience (QoE).

1) **NeRFCompressor *with Model Quantization and Lossless Intra-Scene Compression*:** We evaluate *NeRFCompressor* 's performance on static scenes using quantization and lossless compression, with the video codec's quantization parameter (qp) set to 0 for lossless encoding. Table V reports PSNR results across different scenes using various numbers of components (192, 120, 84, 48). As expected, reducing the number of components slightly degrades reconstruction quality.

We also compare against JPEG compression applied to grayscale components of the original VM-192 model. Lower JPEG qp values (e.g., jpeg10) preserve more detail but result in larger file sizes. Notably, *NeRFCompressor* consistently outperforms all JPEG baselines across all scenes—even at lower bitrates. For example, the "lego" model compressed with *NeRFCompressor* _48 achieves 35.72dB with only 4.75MB, while JPEG at qp=60 yields only 28.51dB despite similar size. These results highlight *NeRFCompressor* 's superior rate-distortion performance. More comparisons are provided in the supplementary material.

2) **NeRFCompressor** ***with Lossy Intra-Scene Compression:*** In this section, we further show the feasibility and effectiveness of using lossy video compression for TensoRF model compression. Here, we use the quantization parameter (qp) in video codecs that regulate the degree of spatial detail preservation. When qp is 0, all details are retained. As qp increases, more details are lost, leading to decreased bit rates, but at the cost of loss of quality. The qp value we chose is based on $\{5n|n \in \mathbb{N}, 0 < n \leq 8\}$. The presented results show the efficacy of applying intra-scene lossy compression on the static scene model. The model size was reduced to a minimum of 1.3MB from the original 68.8MB after compression, a significant reduction. Table II reports the PSNR and model size results compared to state-of-the-art works. Our compression results were slightly better than the original due to the different computing resources. The datasets used in this experiment include both synthetic NeRF [13] and forward-facing [12] datasets. We applied our quantization method to the TensoRF VM-192 model(i.e., $3R_\sigma + 3R_c = 192$) for fair comparison in this table. Compared to VQ-TensoRF [10], *NeRFCompressor* -20 consistently achieves higher PSNR with comparable or smaller representation sizes. For *NeRFCompressor* -30, the representation sizes of *NeRFCompressor* -30 is the smallest, and the PSNR results are better than VQ-TensoRF baseline in two scenes.

3) **NeRFCompressor** ***with Inter-Scene Compression for Dynamic Scenes:*** In this section, we present our results for compressing dynamic scenes using *NeRFCompressor* . Within this experiment, we use two dynamic scene datasets we created. We first compare two alternative training strategies described in Section III-C. Here, we represent the first strategy, using the 1st scene to initialize model components for all remaining scenes as "*1*scene$n$", where n represents the n-th scene. We represent the second strategy as (n−*1*)scene$n$, where the $(n-1)$-th scene is used to initialize the model components for the n-th scene. The experimental results of these two training strategies after applying inter-scene compression indicate the difference in individual model sizes and PSNR results after using trained model components from different scenes as initialization. When using scene 1 to initialize scene n, the model sizes of scenes remain approximately constant. However, when we use the $(n-1)$-th scene for initialization, the model size for scene n increases. The results show that for both training strategies, the PSNR results for subsequent scenes outperform the 1st scene. Therefore, for the entire dynamic scenes, when using the components of the previous scene's model, we prefer to retain its original component matrix to reduce the loss of inter-scene information. As discussed in III-C, we use two ways to merge the component matrices.

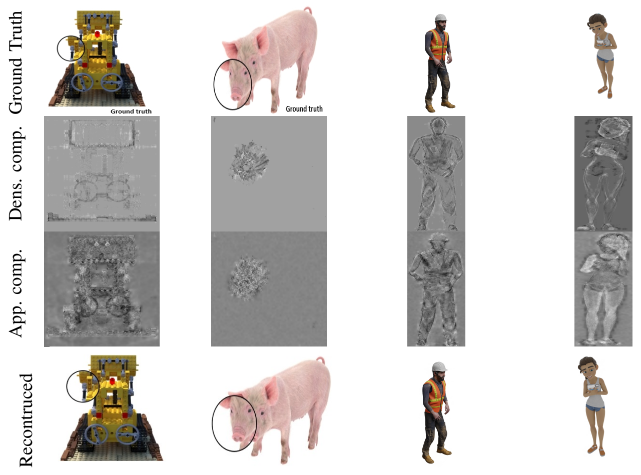

Fig. 3: Visualization of samples in dynamic scenes.

To illustrate in detail, we use the two scenes from the "D-lego" experiment as an example. The size of the parameters for scene1 and 2scene1 is around 68.8 MB before applying intra-scene quantization. Table III presents average PSNR, SSIM, LPIPS, and model sizes across 60 dynamic scenes under different quantization settings. "No compression" reports the aggregated quality and total model size before quantization. We observe that our lossy compression maintains consistently high visual quality, with minimal degradation across increasing quantization levels. The compressed model size is reduced to approximately 2MB per scene on average.

To merge multi-scene models efficiently, we evaluated two strategies: naively concatenating components across scenes and padding and merging corresponding vectors and planes within the density and appearance components. The best approach reduced the total model size to 137.6MB while preserving more detail and delivering higher PSNR, making it the preferred method in our dynamic scene experiments. We then applied intra-scene quantization (Section IV-B2) on top of this merged model for further compression. Figure 3 illustrates qualitative results for representative dynamic scenes with sampled density and appearance features. These visual and quantitative results demonstrate that *NeRFCompressor* delivers strong compression efficiency while preserving visual fidelity in complex, dynamic scenes.

V. Conclusion

In this paper, we presented *NeRFCompressor* , an efficient compression framework for NeRF-based 3D scene representations targeting emerging VR, AR, and MR applications. By combining model quantization with intra- and inter-scene compression, *NeRFCompressor* significantly reduces the size of both static and dynamic scene representations while preserving high visual quality. Our experimental results demonstrate that *NeRFCompressor* achieves strong reconstruction performance with minimal computational and bandwidth overhead, making it well-suited for real-time, immersive applications requiring 6-DoF interaction. These findings highlight *NeRFCompressor*

TABLE II: **PSNR Comparison results.** This table shows the PSNR comparison results between our approach with different models.NeRFCompressor -n represents $qp = n$

	Synthetic-NeRF					LLFF				
	PSNR	SSIM	$LPIPS_{Vgg}$	$LPIPS_{Alex}$	Size	PSNR	SSIM	$LPIPS_{Vgg}$	$LPIPS_{Alex}$	Size
NeRF [13]	31.01	0.947	0.081	-	5MB	26.50	0.811	0.250	-	5MB
TensoRF [5]	33.09	0.963	0.048	0.027	67.56MB	26.70	0.836	0.202	0.115	179.84MB
VQ-TensoRF [10]	32.86	0.960	0.058	0.032	3.56MB	26.46	0.824	0.224	0.129	8.72MB
NeRFCompressor	33.13	0.963	0.047	0.027	9.53MB	26.74	0.839	0.201	0.119	21.71MB
NeRFCompressor -20	32.97	0.961	0.051	0.029	3.18MB	26.69	0.836	0.209	0.124	6.00MB
NeRFCompressor -30	32.35	0.958	0.060	0.036	2.28MB	26.24	0.819	0.235	0.171	2.69MB

TABLE III: **Experimental results of inter-scene compression for dynamic 60 scenes.**

	qp	PSNR	SSIM	$LPIPS_{Alex}$	$LPIPS_{Vgg}$	Size	PSNR	SSIM	$LPIPS_{Alex}$	$LPIPS_{Vgg}$	Size
		Lego					Pig				
No compression	-	37.6305	0.9923	0.0058	0.0149	4.03GB	42.0324	0.9753	0.0285	0.0294	3.97GB
$NeRFCompressor_{SF}$	10	37.4332	0.9858	0.0063	0.0156	399.63MB	41.5424	0.9688	0.0289	0.0333	567.39MB
	20	36.4289	0.9681	0.0068	0.0167	250.67MB	40.6810	0.9643	0.0296	0.0397	356.36MB
	30	33.4040	0.9730	0.0164	0.0357	138.09MB	38.5194	0.9591	0.0390	0.0314	189.34MB
$NeRFCompressor_{SA}$	10	37.4283	0.9858	0.0063	0.0156	535.01MB	41.5437	0.9688	0.0289	0.0333	472.56MB
	20	36.3091	0.9668	0.0069	0.0169	258.17MB	41.4458	0.9681	0.0292	0.0343	320.16MB
	30	32.6267	0.9694	0.0195	0.0415	121.75MB	40.2093	0.9619	0.0297	0.0433	152.77MB
		Worker					Amily				
No compression	-	43.2173	0.9964	0.0041	0.0014	4.60GB	44.6270	0.9901	0.0090	0.0023	4.83GB
$NeRFCompressor_{SF}$	10	43.1522	0.9914	0.0047	0.0018	412.82MB	44.2341	0.9879	0.0095	0.0026	436.20MB
	20	42.7651	0.9883	0.0051	0.0022	303.73MB	43.5625	0.9814	0.0103	0.0029	366.72MB
	30	38.4683	0.9699	0.0055	0.0039	159.51MB	40.5872	0.9603	0.0224	0.0084	202.08MB
$NeRFCompressor_{SA}$	10	43.1123	0.9912	0.0049	0.0019	451.09MB	43.9977	0.9899	0.0097	0.0027	418.93MB
	20	42.0008	0.9849	0.0052	0.0026	303.11MB	43.0009	0.9801	0.0142	0.0032	352.48MB
	30	36.9459	0.9561	0.0063	0.0043	121.39MB	39.9886	0.9532	0.0278	0.0088	188.65MB

's potential for practical deployment in resource-constrained multimedia systems.

References

[1] Volumetric video market. https://www.marketsandmarkets.com/Market-Reports/volumetric-video-market-259585041.html, 2021.

[2] S.-H. Bae, J. Kim, M. Kim, S. Cho, and J. S. Choi. Assessments of subjective video quality on hevc-encoded 4k-uhd video for beyond-hdtv broadcasting services. *IEEE Transactions on Broadcasting*, 2013.

[3] A. Cao and J. Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 130–141, 2023.

[4] J. Cao, H. Wang, P. Chemerys, V. Shakhrai, J. Hu, Y. Fu, D. Makoviichuk, S. Tulyakov, and J. Ren. Real-time neural light field on mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8328–8337, June 2023.

[5] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022.

[6] Z. Chen, T. Funkhouser, P. Hedman, and A. Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2023.

[7] S. Fridovich-Keil, G. Meanti, F. Warburg, B. Recht, and A. Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance, 2023.

[8] A. Hore and D. Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, 2010.

[9] C. Li, S. Li, Y. Zhao, W. Zhu, and Y. Lin. Rt-nerf: Real-time on-device neural radiance fields towards immersive ar/vr rendering. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2022.

[10] L. Li, Z. Shen, Z. Wang, L. Shen, and L. Bo. Compressing volumetric radiance fields to 1 mb. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[11] T. Li, M. Slavcheva, M. Zollhoefer, S. Green, C. Lassner, C. Kim, T. Schmidt, S. Lovegrove, M. Goesele, R. Newcombe, and Z. Lv. Neural 3d video synthesis from multi-view video. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[12] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines, 2019.

[13] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[14] D. Rho, B. Lee, S. Nam, J. C. Lee, J. H. Ko, and E. Park. Masked wavelet representation for compact neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2023.

[15] S. Rojas, J. Zarzar, J. C. Perez, A. Sanakoyeu, A. Thabet, A. Pumarola, and B. Ghanem. Re-rend: Real-time rendering of nerfs across devices. *arXiv preprint arXiv:2303.08717*, 2023.

[16] C. Sun, M. Sun, and H.-T. Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5459–5469, June 2022.

[17] J. Tang, X. Chen, J. Wang, and G. Zeng. Compressible-composable nerf via rank-residual decomposition. *Advances in Neural Information Processing Systems*, 2022.

[18] S. Tomar. Converting video formats with ffmpeg. *Linux journal*, 2006(146):10, 2006.

[19] H. Turki, D. Ramanan, and M. Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12922–12931, June 2022.

[20] G. K. Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 2004.

[22] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu. NeRF−−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[23] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[24] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

[25] M. Zhu, Y.-C. Sun, N. Li, J. Zhou, S. Chen, C.-H. Hsu, and Y. Liu. Dynamic 6-dof volumetric video generation: Software toolkit and dataset. In *2024 IEEE 26th International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6, 2024.