Streaming Radiance Fields for 3D Video Synthesis

Lingzhi Li
Alibaba Group
lingzhili@pku.edu.cn

Zhen Shen
Alibaba Group
zackary.sz@alibaba-inc.com

Zhongshu Wang
Alibaba Group
zhongshu.wzs@alibaba-inc.com

Li Shen
Alibaba Group
lshen.lsh@gmail.com

Ping Tan
Alibaba Group
pingtan@sfu.ca

Abstract

We present an explicit-grid based method for efficiently reconstructing streaming radiance fields for novel view synthesis of real world dynamic scenes. Instead of training a single model that combines all the frames, we formulate the dynamic modeling problem with an incremental learning paradigm in which per-frame model difference is trained to complement the adaption of a base model on the current frame. By exploiting the simple yet effective strategy of tuning with narrow bands, the proposed method achieves a feasible framework for handling on-the-fly video sequences with high training efficiency. The storage overhead induced by using explicit grid can be significantly reduced by using model difference based compression. We also introduce a curriculum learning strategy to further accelerate model optimization for each frame. Experiments on challenging video sequences demonstrate that our approach is capable of achieving a training speed of 15 seconds per-frame with competitive rendering quality, which attains $1000\times$ speedup over the state-of-the-art implicit methods.

1 Introduction

3D video synthesis aims to realize free-viewpoint photo-realistic rendering for dynamic scenes, which are typically recorded via a set of cameras (with known poses) from multiple views. The topic has attracted much research effort because of its potential value in a wide range of applications in VR/AR. Traditional techniques by estimating surfaces [1], multi-sphere images [2] or depth [3, 4] via multi-view stereo [5, 6] are usually used for modeling and representing dynamic scenes. However, arbitrary geometry and complex appearance exhibiting in real-world scenarios pose challenges for leveraging a general methodology to pursue high-performance modeling.

Neural radiance fields (NeRF) [7, 8] have recently emerged as a new methodology for effectively reconstructing and rendering static scenes via neural networks. These methods learn a continuous mapping between 3D points (given view directions) and their corresponding radiance colors and opacity, to realize high-fidelity rendering results through the use of volumetric rendering techniques [9]. However, they often suffer from costly training and inference due to a tremendous amount of computations through neural networks. When extending the implicit formulation to a dynamic scene [10], time steps are embedded as additional input for training models across frames and achieving time-dependent rendering. Compared to training on static scenes, training overhead increases significantly with respect to sequence length, e.g., costing about 56 GPU days for training 300 frames, that would be prohibitive when handling long sequences. More importantly, such learning paradigm is restricted to offline modeling, thus unable to tackle online scenarios.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

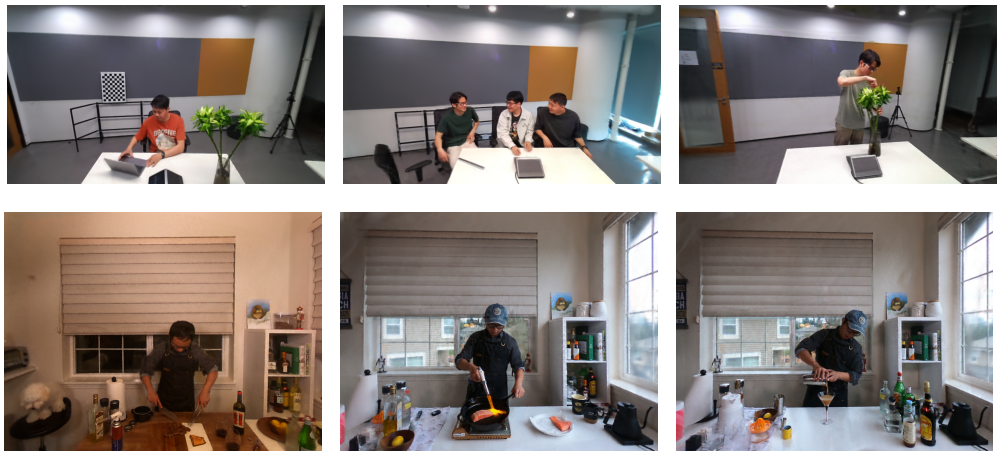


Figure 1: **Rendering results on test view.** *Top*: Our meet room dataset; *Bottom*: N3DV dataset. These novel view results are rendered at interactive speed ($\sim$10 FPS) by StreamRF.

In this work, we propose a novel method for reconstructing streaming radiance fields for novel view synthesis of 3D videos. We formalize the modeling of a dynamic scene in an incremental learning framework, enabling it to tackle video sequences on-the-fly. Inspired by the recent success achieved by Plenoxel [11] on training and inference efficiency, our method is built on explicit grid representations. Specifically, the overall training process is decomposed to the complete training of a basic voxel grid model at the first frame, with an online tuning mechanism for learning and storing model difference for each subsequent frame. A complete grid model given a time step i can be simply attained by adding the model difference to the previous model at $i-1$ (or accumulating all the model differences before i with the basic grid model). By exploiting the prior of exhibiting local continuity between adjacent frames, we present a narrow-band tuning strategy for capturing and training the model difference efficiently, associated with a compression strategy for significantly reducing memory cost that explicit grid methods struggle with. To further improve the training efficiency, we propose to learning pilot models to guide the optimization of full-scale models. The overall framework is simple yet efficient. Quantitative and qualitative experimental results demonstrate the effectiveness of our approach. The approach can achieve the training speed of 15 seconds and inference time of 120 ms per frame for rendering a 1k resolution image with competitive quality, achieving three orders of magnitude faster than the state-of-the-art dynamic neural radiance field methods (costing 56 GPU-days for training a sequence with 300 frames). Compared to the efficient explicit-grid baseline by per-frame training, it still can obtain two orders of magnitude acceleration with a much lower storage cost. The empirical analysis further show that using a proper incremental training mechanism would not degrade performance when dealing with relatively long sequences. code and dataset will be released at https://github.com/AlgoHunt/StreamRF

2 Problem Formulation

Neural Radiance fields (NeRF) [7] model a *static* scene implicitly by learning a continuous mapping from a 3D spatial position $\mathbf{x} \in \mathbb{R}^3$ specified in a viewing direction $\mathbf{d} \in \mathbb{R}^3$ (represented as Cartesian unit vectors) to RGB radiance color $\mathbf{c} \in \mathbb{R}^3$ and opacity $\sigma \in \mathbb{R}$. The mapping function is typically implemented by a multilayer perceptron (MLP) $f_m : (\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma)$, trained on a set of images with known camera pose. Given a ray $\mathbf{r}$ casting on the image plane through a pixel, by using a traditional volume rendering technique [12], the expected color $\hat{C}(\mathbf{r})$ of the pixel is rendered by integrating the opacity and the color of sampling points along the ray:

$$\widehat{C}(\mathbf{r}) = \sum_{i=0}^{N-1} T_i \left(1 - \exp(-\sigma_i \delta_i)\right) \mathbf{c}_i, \quad T_i = \exp\left(-\sum_{j=0}^{i-1} \sigma_j \delta_j\right), \tag{1}$$

where δ_i denotes the distance to adjacent points. These methods are subjected to slow training and rendering speed, since the ray casting involved in both training and rendering stage requires hundreds

of forward propagation (and backward propagation in training) through MLPs for the sampling points on each ray. For example, JaxNeRF [13], though optimized through parallelization, needs over 20 seconds to render an image with 800×800 resolution with over 10 hours of training.

When extending the mechanism to model a *dynamic* scene, one straightforward solution is to embed the time variable t into the representation, i.e., $f_t : (\mathbf{x}, \mathbf{d}, t) \mapsto (\mathbf{c}, \sigma)$, while training time would dramatically increase when dealing with a large number of frames. Training a video with original NeRF frameworks may cost over 600 GPU-days. N3DV [10] improves training efficiency by virtue of sophistical designs, however it still needs about 56 GPU-days for pursuing a model with reliable rendering quality. More importantly, these implicit formulations are naturally suitable for offline processing which feeds all the (key)frames in advance for training, and may not be easily generalized to online training scenarios.

Recent progress on explicit representations has demonstrated compelling improvement on training and render efficiency for modelling a *static* scene. The scene space is discretized and represented explicitly with a sparse voxel grid instead of learning representations implicitly with large MLPs. As a representative work, Plenoxel [11] optimizes voxel grids without neural networks. Each voxel in the grid consists of the scalar opacity σ and the coefficient vector $\mathbf{k}$ of spherical harmonics (SH) for respectively describing geometric and color information. The color at the voxel v is obtained by,

$$\mathbf{v}_c = c(\mathbf{d}, \mathbf{k}) = \xi \left(\sum_{l=0}^{l_{max}} \sum_{m=-l}^{l} \mathbf{k}_l^m \cdot Y_l^m(\mathbf{d}) \right), \tag{2}$$

where ξ is the sigmoid function and $\{Y_l^m\}$ denotes the set of SH basis. To render the expected color of the pixel through a ray, the opacity and the color of a sampling point $\mathbf{x}$ are calculated by performing trilinear interpolation among the voxels within its neighborhood $\mathcal{N}(\mathbf{x})$,

$$\sigma = U_{\mathbf{v} \in \mathcal{N}(\mathbf{x})} \left(\mathbf{v}_\sigma \right), \quad \mathbf{c} = U_{\mathbf{v} \in \mathcal{N}(\mathbf{x})} \left(\mathbf{v}_c \right). \tag{3}$$

Then the pixel color is produced by integrating the sampling points along the ray based on Eq. 1. The voxels with zero opacity (or truncated by a minimal threshold) are pruned after sparsification, and an occupancy mask $\mathbf{M}$ with binary values is used for indicating if the space is occupied. More details can refer to the paper of Plenoxels [11]. In theory, the mechanism could significantly reduce the overall computational complexity as it omits the requirement of numerous MLP-based costly computation during training and inference. It is able to train a static scene with good rendering quality within 15 minutes on single GPU and render a $1k$ resolution image in 100 ms. These efficiency gains would benefit developing a highly efficient framework for modeling and rendering a dynamic scene.

3 Streaming Radiance Fields for Dynamic Scenes

To realize sparse grids for the modeling of a dynamic scene, one intuitive way is to directly extend 3D spatial dimensions (denoted by $H \times W \times D$) with the 4-th dimension indicating the time steps t for frames. However, using explicit grids for a static scene have faced challenges on storage although progressive training and sparsifying (i.e, pruning off empty voxels) [11] are used to alleviate the issue of costly overhead. Simply expanding the grid into a space-time manner would make storage overhead increase linearly with t, which is infeasible for handling videos with long sequences. On the other hand, per-frame training individually without considering temporal correlation may be feasible for achieving a tradeoff between storage and efficiency, while it is still far from the objective of pursing a high-performing framework for 3D video synthesis.

In this regard, we propose a novel method, named StreamRF, to realize a streaming radiance field for effectively rendering dynamic scenes. Our approach is built on the benefits of utilizing an explicit representation via sparse voxel grids. Particularly, by leveraging some intrinsic priors exhibiting in videos, we develop an incremental training framework which enables a faster training convergence and lower storage over per-frame grid modelling.

Videos in forward-facing scenarios are typically composed of time-invariant components (e.g., static background) and time-variant components (e.g., moving objects) with small variation that exhibits local continuity between adjacent frames. By exploiting the fact, we decompose the problem of sequential modeling into two sub-problems. We first fully train a base grid model given the first frame, denoted as $\mathbf{V}^0 = \{\mathbf{V}_c^0, \mathbf{V}_\sigma^0\}$ which stores the corresponding color features (the coefficients of

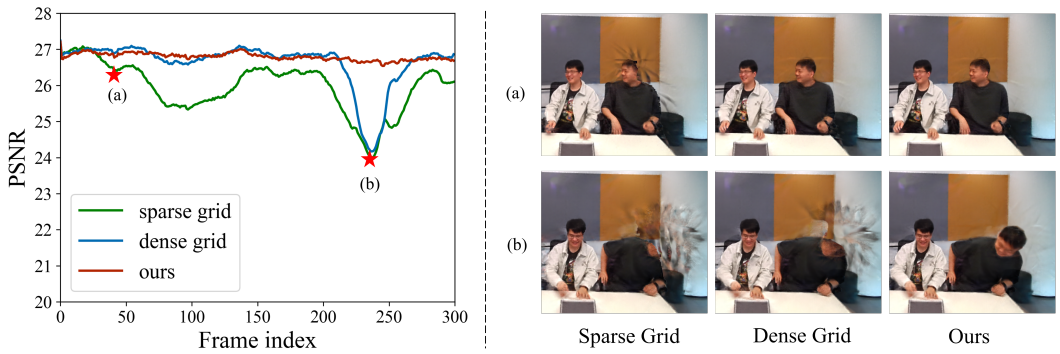


Figure 2: *Left*: PSNR comparison between tuning with (1) original sparse grid (2) dense grid and (3) our narrow band finetune in a 300 frames sequence; *Right*: visual comparison of above methods on (a) $40th$ frame (b) $240th$ frame. All results are trained with the same initial model.

spherical harmonics) and opacity values. Then the model is adapted to succeeding frames through the use of an effective tuning strategy. Per-frame tuning is expected to perform only on the changes between adjacent frames. The grid model in the current frame is produced by adding the model difference, which is optimized and stored during training phase, to the grid model in the previous frame. Formally, assume the model difference between the $i-1$-th and i-th frames is denoted as $\delta_i = \Delta(\mathbf{V}^i, \mathbf{V}^{i-1})$. In the inference phase, the grid model at the time step i can be obtained by,

$$\mathbf{V}^i = \mathbf{V}^{i-1} + \delta_i = \mathbf{V}^0 + \sum_{j=1}^{i} \delta_j. \tag{4}$$

Such an incremental formulation indeed presents an online training paradigm, enabling the proposed framework to handle video sequences on-the-fly.

3.1 Tuning with Narrow Bands

A problem arises: how to effectively capture the model difference between adjacent frames $i-1$ and i. Given the sparse grid model $\mathbf{V}_{i-1}$, any object motion at the time step i may result in changes on the opacity and colors of a part of voxels. It is hence likely to happen that informative points occur in the originally empty space as well as some voxels originally occupied becomes empty. If we directly tune the sparse grid, only the non-empty voxels could be changed or pruned in a one-way manner while unable to add informative voxels. It is problematic when handling long sequences (shown by the ablation study on the sparse grid in Fig. 2). To address the issue, we propose a tuning strategy based on narrow bands. We first retrieve and activate the narrow band for the frame i, i.e., the region of interest in the grid $\mathbf{R}^i$, based on the occupancy mask $\mathbf{M}^{i-1}$ (with binary values) which indicates non-empty voxels at the frame $i-1$,

$$\mathbf{R}^i = \mathcal{F}_{di}(\mathbf{M}^{i-1}, \rho_d) \oplus \mathcal{F}_{er}(\mathbf{M}^{i-1}, \rho_e), \tag{5}$$

where $\mathcal{F}_{di}$ and $\mathcal{F}_{er}$ correspond to dilation and erosion operations, and ρ_d and ρ_e denote the corresponding radius parameters. $\oplus$ denotes the element-wise xor operation on the masks to extract the additional region of the dilated area compared to the eroded one. The narrow band actually represents a thin area around the surface of scenes (or objects in a composite scene). We use it for two purposes:

- For every empty voxel that falls into the band, we restore them to the grid and initialize the opacity and color features with zero values.
- Only the voxels within the band are tuned and the rest ones are not involved during the training stage for the frame i.

In other words, the tuning always performs on a relative small set of voxels in the frame i. We also prune empty voxels off to get a compact grid $\mathbf{V}_i$ with an occupancy mask $\mathbf{M}_i$ after tuning.

The key idea of narrow band-based tuning is built upon leveraging the intrinsic priors: motions induced by moving objects typically show small changes and local continuity within neighboring

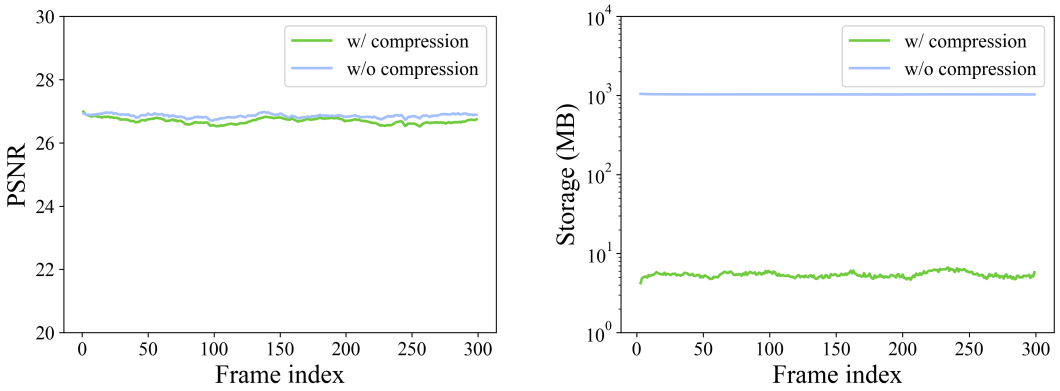


Figure 3: **Ablation study of diff-based compression**. We compare per frame PSNR (**left**) and storage size (**right**) with diff-based compression enabled and disabled. The storage space drops to 0.5% of the original with a negligible difference on PSNR (decrease 0.156 in average)

areas between adjacent frames. These assumptions have been successfully exploited in a wide range of video processing [14] and video understanding tasks [15, 16], e.g, traditional mean-shift method for tracking [17]. We demonstrate that the strategy can effectively benefit to training convergence and be capable of achieving the speed of ~ 15s per-frame training for achieving a competitive quality (shown in section 5.4). Compared to the state-of-the-art implicit method [10] with the training speed of 4.5 GPU-hours for each frame in average, our method is about three orders of magnitude faster on training and can achieve over $500\times$ speedup on rendering efficiency (over 60 s s VS 121 ms).

3.2 Difference-based Compression

Using explicit grid modelling may pose the issue of large storage cost. Taking a grid size ($1408 \times 1156 \times 128$ in the experimental setting) for example, the overall storage for a conventional sparse grid costs about 1015 MB after sparsity for one frame. By virtue of narrow band tuning, we only need to store the model difference δ_i with a significantly smaller size compared to the complete grid model. The change of storage cost only happens in the following two scenarios, i.e., when adding voxels into the grid (i.e., appearing in the empty space) and pruning voxels off from the grid (i.e., the previously occupied space becomes empty). In this regard, we introduce a difference-based compression strategy to further reduce the storage overhead. We can efficiently find out the change areas by using the occupancy masks $\mathbf{M}^{i-1}$ and $\mathbf{M}^i$ between the adjacent frames $i-1$ and i,

$$\begin{aligned} \mathbf{M}_a^i &= \mathbf{M}^i \odot (1 - \mathbf{M}^{i-1}), \\ \mathbf{M}_e^i &= \mathbf{M}^{i-1} \odot (1 - \mathbf{M}^i), \\ \mathbf{M}_r^i &= \mathbf{M}^{i-1} \odot \mathbf{M}^i \odot \ \mathbf{1}\{\gamma(\delta_i^c) > \epsilon\}, \end{aligned} \tag{6}$$

where $\mathbf{M}_a^i$, $\mathbf{M}_e^i$ and $\mathbf{M}_r^i$ respectively denote the masks of adding voxels, erasing voxels and remaining ones while the features at the voxels may change. $\gamma(\cdot)$ defines the function of getting the L1 norm along the axis of color features (SH coefficients) on the model difference, and ϵ denotes a threshold. $\mathbf{M}_r^i$ actually involves the voxels with relatively large modification. Instead of storing model difference δ_i directly, we store the three masks, the voxel representations (SH coefficients and opacity) belonging to the adding area $\mathbf{M}_a^i$ and the representation differences for the voxels within the remaining area $\mathbf{M}_r^i$. It can achieve a remarkable reduction rate (178 times) on storage that reduces **1015 MB** to **5.7 MB** for each frame in average.

4 Efficient Training with Pilot Model Guidance

To further improve the training efficiency of StreamRF, we exploit an optimization strategy inspired from curriculum learning to train from an easier problem. Formally, given the grid model $\mathbf{V}^{i-1}$, we adopt down-sampling operations to get a *pilot* model of a smaller size,

$$\widehat{\mathbf{V}}^{i-1} = \downarrow^S (\mathbf{V}^{i-1}). \tag{7}$$

When setting the S to $1/2$, the capacity of pilot model is roughly $1/8$ of the full grid. Training on a smaller model would reduce the computational complexity at every iteration (with the same batch size) and also ease optimization. The model difference on the pilot models $\widehat{\delta}_i = \Delta(\widehat{\mathbf{V}}^i, \widehat{\mathbf{V}}^{i-1})$ are implemented by using $\{\widehat{\mathbf{M}}_a^i, \widehat{\mathbf{M}}_e^i, \widehat{\mathbf{M}}_r^i\}$ as following Eq. 6. Then we can get a guidance mask $\widehat{\mathbf{G}}^i$ by merging the three mask via elementwise "or" operator and upsampling to the full scale,

$$\begin{aligned} \widehat{\mathbf{G}}^i &= \widehat{\mathbf{M}}_a^i \vee \widehat{\mathbf{M}}_e^i \vee \widehat{\mathbf{M}}_r^i, \\ \mathbf{G}^i &= \uparrow^S (\widehat{\mathbf{G}}^i). \end{aligned} \tag{8}$$

The guidance mask is used for regularizing the gradient update when tuning the full-scale model. The values of added and erased voxels are filled back to the full-scale grid. All the voxels outside the mask are frozen without optimization when training at the frame i.

The design of the strategy comes from the insight that training at a small size converges much faster and the induced difference of pilot model indicates the voxels that need to be modified. Using this strategy can prevent unnecessary modification in the full-scale model and avoid the loss of high-frequency details during downsample-upsample process. We observe that using this strategy facilitates pursuing a higher fidelity model compared to directly training on the full-scale grid with the same training time. The qualitative results (in Fig. 4) reveals that the model via directly training on the full-scale grid may encounter flicker issues when rendering along time steps, i.e., sparkling noise may occur in the stable background area when training period is short. The issue could be addressed by lengthening the training schedule with a small learning rate. While training with the pilot model guidance can effectively prohibit flicker artifacts from happening, enabling a more efficient and stable optimization for the streaming paradigm.

5 Experiments

5.1 Datasets

Meet Room Dataset We build a multi-view capture system using 13 Azure Kinect cameras. We use twelve cameras for training and one for testing, with an external pulse signal to synchronize all camera shutter. Though this system is capable of depth capture, we do not use depth images in this research. The videos are recorded at a resolution of 1280×720 and a frame rate of 30 FPS. The total size of the capture system is 100 cm $\times$ 75 cm and the captured views are aligned much sparser than some common datasets such as LLFF and N3DV. We will release this dataset for research purposes.

Neural 3D Video (N3DV) Dataset [10] It contains 6 dynamic indoor scenes with varying illuminations, view-dependent and highly volumetric effects. Videos are captured at a resolution of 2704×2078 and a frame rate of 30 FPS. Following the setting in the original paper [10], the quantitative score is produced on the designated sequence ("flame_salmon") by downsampling it to 1352×1039 for a fair comparison, where the frames captured by 18 views are used for training and the rest one for testing.

5.2 Implementation Details

Initialization. We follow the standard training protocol of a forward-facing scene in [11] to obtain the base model at the first frame. We start by a $256 \times 256 \times 128$ grid which is upsampled and sparsified for every 38.4K iterations until the model reaches the final resolution of $1408 \times 1156 \times 128$. The total training takes 128K iterations with a batch size of 5K rays. We adopt the RMSProp [18] optimizer with a decay parameter of 0.95 for optimization. We adopt a slighter larger TV penalty to mitigate foggy issues, where λ_{TV} is set to 5×10^{-4} for opacity and 5×10^{-3} for color features.

Per Frame Tuning. The full-scale model and the pilot model are trained in parallel. For the experiments on the Meet Room dataset, we train the pilot model with 1000 iterations and then tune the full-scale model with 500 iterations. For the experiments on the N3DV dataset, we train the pilot model with 750 iterations and then tune the full model with 500 iterations. We use RMSprop to optimize both models with a batch size of 20k rays. To promote the pilot model to have a better convergence, we enlarge λ_{TV} by 10 times when training the pilot model.

Diff Based Compression. We set the SH coefficient threshold ϵ to 1.5/27 and 1/27 for Meet Room Dataset and N3DV dataset respectively by the consideration of coefficient dimension (i.e., 27). We

Table 1: **Comparison with related works on the N3DV and Meet Room datasets.** We report *per frame* PSNR, training and inference time and storage cost in average for a clear comparison. All the results are recorded with our 3090 GPU except the results of N3DV are referred to the numbers in the original paper. Compared to all the baselines, our method can achieve remarkable speedup on training time with competitive performance on the other metrics.

Methods	N3DV				Meet Room			
	PSNR (dB)	Train (minutes)	Inference (seconds)	Storage (MB)	PSNR (dB)	Train (minutes)	Inference (seconds)	Storage (MB)
JaxNeRF*[13]	28.53	485	67	14	27.11	473	40	14
LLFF*[20]	23.23	8	**0.004**	3192	22.88	3	**0.003**	1500
N3DV [10]	**29.58**	260	>67	**0.1**	-	-	-	-
Plenoxels*[11]	28.68	23	0.12	4106	**27.15**	14	0.1	1015
Ours	28.00	**0.25**	0.12	12.6/26.3	26.72	**0.17**	0.1	5.7/9.0

* Denote training from scratch per frame.

found using too small value would drastically increase saving size while the gain of rendering quality is marginal. In the experiments we use them by default to achieve a proper trade-off between storage cost and rendering quality. In order to further reduce storage cost, we convert all saving SH coefficients and opacity to float16 and merge $\mathbf{M}_a^i$ and $\mathbf{M}_e^i$ to one single mask to save more space in the implementation. Moreover, as the mask is mostly comprised of consecutive binary values, it can be compressed with a very high compression ratio (over 99%) simply with a standard compression library (*e.g.* ZLIB[19]). We pack all tensors together and compress them with ZLIB to get the final on-disk storage cost for each frame.

Pilot Model Guidance. We first downsample the grid model and apply the standard training pipeline (narrow band and difference-based compression) for pilot model training. The induced model difference is used in two purposes, including filling the values of added and erased voxels back to the full-scale model and tuning the full-scale model with the guidance mask.

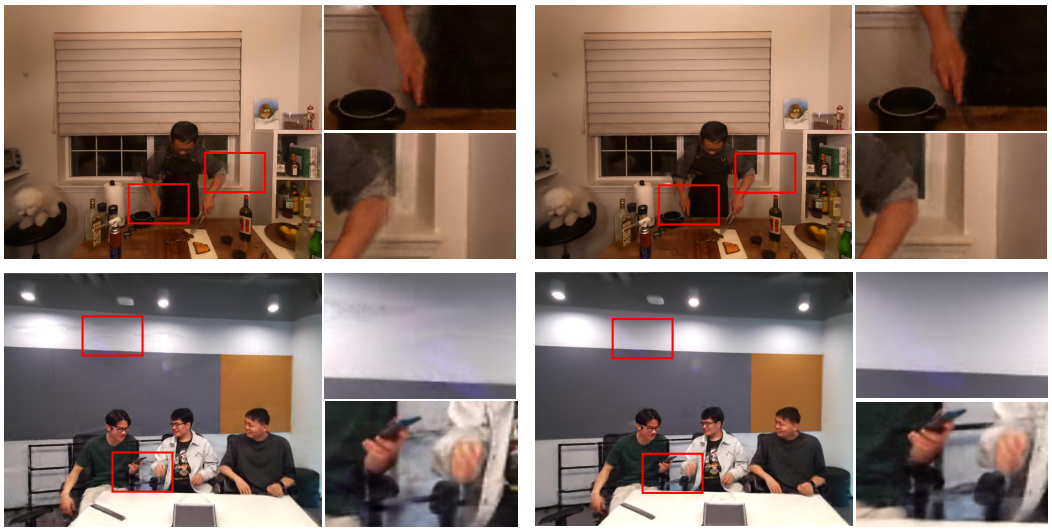
5.3 Experimental Comparison

We first quantitatively compare the method with some representative works including N3DV which achieves the state-of-the-art rendering quality. For the rest baselines, i.e., JaxNeRF [13], LLFF [20] and Plenoxels [11] which are originally designed and trained on static scenes, we train them in a per-frame independent manner, following the default setting as in their original papers. We use peak signal-to-noise ratio (PSNR), training and inference time and storage cost as metrics for a comprehensive evaluation, and report the mean scores per-frame. We refer to the scores reported in N3DV [10] for a fair comparison and estimate the rendering time of the method according to its network design. As it used a larger MLP than the one in NeRF [7], its inference time is supposed to be longer than NeRF. All the baselines are evaluated every ten frames as in [10].

As shown in the Table 1 , the implicit methods JaxNeRF and N3DV present fairly high computation cost, requiring over 8 GPU hours and 4 GPU hours for per-frame training, respectively. As for testing speed, the LLFF achieve the fastest record. Because it only needs blending the multi-plane image to render a novel view. However, it fails to produce a satisfactory result. We achieve the optimization speed of costing 15 seconds only, outperforming all the baseline methods in terms of training efficiency. Meanwhile our method can achieve a comparable PSNR to the previous SOTA. For the meet room dataset, we use the same sequence as in Fig 2 for testing and following the same test protocol as above. Because our dataset contains fast motion and camera spacing is larger than N3DV dataset, we could see a clear PSNR drop on all four methods. Nevertheless, our method still achieves a comparable performance with per frame training counterpart, around 0.4dB drop in PSNR compared to JaxNeRF and Plenoxels, while obtaining the acceleration over $2,700\times$ and $80\times$, respectively. More experimental results and analysis can be referred to supplementary materials.

5.4 Ablation Studies

Narrow Band Finetune. We compare the narrow band with the following two counterparts, both quantitative and visually: (1) directly adopt the original sparse voxels grid, (2) switch the original sparse voxels grid to its dense counterpart. It can be seen in Fig.2. The sparse grid failed immediately



w/o Pilot Model Guidance w/ Pilot Model Guidance

Figure 4: **Ablation study of pilot model guidance:** we compare **Left:** StreamRF trained Without pilot model guidance and **Right:** with pilot model guidance in both N3DV dataset (top) and meet room dataset (bottom). These results reflect that pilot model can help reduce artifacts in both static background and dynamic foreground.

(Fig.2 (a)), though the right person only move at a small distance. This artifact is because the subject has moved out of the trainable area. A dense voxel grid allows every voxel to be trainable. It still fails when the right person has a large movement(Fig.2 (b)). This artifact could be caused by the rapidly increasing number of active voxels after switching from sparse to dense (5% vs. 100% of all voxels). Both PSNR curve and rendering result show our method achieves comparable PSNR with a dense grid and can handle large movement in a dynamic scene. Moreover, in this ablation study, our method trains around 40% faster and requires 63% less memory consumption than dense finetune.

Diff-based Compression. We validate the performance and effectiveness of diff-based compression in Fig. 3. In this ablation, we first train our StreamRF on an entire sequence. The trained StreamRF is then saved with diff-based compression enabled and disabled. We then compare the PSNR and saving size between the two test view sequences. We present the per frame PSNR to show whether diff-based compression impairs the performance in the long run.

The quantitative result shows that our compression method has negligible impact after a very long propagation. Besides deducting overall storage size from 1015 MB to an average of 5.7 MB for each frame, we can observe that performance of our method does not downgrade with time and the incremental storage for each frame is roughly stable along time steps. More details of model size reduction can be referred to supplementary materials.

Pilot Model Guidance. We validate our pilot model guidance visually in Fig. 4. For base setting (right), we train the pilot model with 1,000 iterations and the full model with 500 iterations. For the ablation setting (left), we directly train the full model with 1,000 iterations to make per frame training time equivalent in both settings. As shown in Fig. 4, pilot model guidance help reduce various kinds of artifacts: (top) vanishing hands and blurry elbow. (bottom) sparkle noise in a white wall, the fuzzy area between two people. These results verified our intention that we could use an easy-to-optimized small model to provide a more optimal initialization to the full model.

6 Related Work

Static Novel View Synthesis Novel view synthesis from a set of input images with known camera poses has been studied intensively in the field of computer vision and computer graphics. Earlier works like Lumigraph [21, 22] and Light-Field [23, 24] focus on interpolating the sampled light rays for realistic rendering. It has been proved [25, 26] that the sampling density heavily depends on the

underlying geometry of the captured scene. Thus, many works strive to recover an approximate [27, 28] or precise geometry model [29] to facilitate novel view rendering. Among those works, multi-plane image (MPI) [20, 30–32] is a promising approximate geometry model that enables realistic rendering of generic scenes.

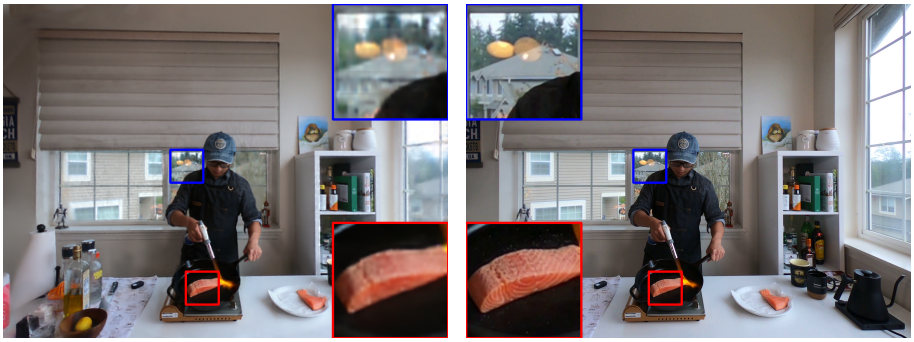
More recently, NeRF [7] represents the radiance field with a multi-layer perceptron (MLP) network and combines it with differentiable volume rendering for view interpolation. This approach achieved tremendous success and there are many variants to push its limits towards every aspect. The works [8, 33–35] focus on improving rendering quality, while the works [36–38] strive to accelerate rendering speed, and other works [11, 39–42] propose to replace implicit function by an explicit or hybird counterpart to boost training convergence.

Dynamic Novel View Synthesis Novel view synthesis has also been heavily studied with dynamic scenes. Early effort of generating 3D video can trace back to the 90s [4, 43], which computes explicit depth maps to facilitate novel view interpolation. This approach has been combined with more sophisticated capturing hardware to generate streamable free-viewpoint video in [1]. Matusik and Pfister [44] further integrate camera arrays and 3D displays to build an end-to-end 3D TV system.

More recent works learn a neural network to capture radiance fields of dynamic scenes. A family of methods [45–51] realize view synthesis by training on monocular videos captured by a moving camera, i.e., viewpoints change with respect to time steps. The works in [48–50] decouple the overall radiance field reconstruction to the learning of a static canonical template and a deformation field warping to the template. Park et al. [51] lifts canonical templates into hyper-spaces to induce flexibility on modelling variations in topology. Several methods model dynamics through the use of scene flows [45, 47] or depth prior [46], therefore they highly reply on the regularization of extra supervision (e.g., optical flow, depth or foreground-background segmentation estimated by well-trained models). Broxton et al. [2] and Attal et al. [52] use multi-camera stereo rig and extend MPI to MSI to support view interpolation with a larger field of view. Bansal et al. [53] use multiple mobile phones to capture a dynamic scene and compose the static and dynamic components with neural networks for 4D space-time visualization. The closest work to our setting is Neural 3D Video (N3DV)[10], which captures dynamic scenes with synchronous video sequences from multiple cameras. It directly extends NeRF with an additional timecode input, and accelerates computation with a hybrid importance sampling and hierarchical training. N3DV can produce strong visual results given enough optimization, while it is very time consuming, requiring over per frame 5 GPU hours in training and 10+ seconds in rendering, making it impractical for real-world applications.

On the other hand, some approaches adapt the advances in neural rendering for the modeling of specific objects, e.g., human bodies [54, 55] by integrating with domain-specific priors, e.g., skeleton or SMPL mesh [56]. In contrast, our method imposes no prior in objects and is capable of representing more general scenes with complex deformation.

7 Limitation & Future work



(a) Ours (b) Ground Truth

Figure 5: Our method does not reconstruct details and translucent surface very well.

In Fig. 5 we can still observe artifacts like translucent foreground and texture sticking in the rendering images which may "inherit" from using an explicit representation. We expect to improve the rendering quality by extending the framework with explicit-implicit hybrid representations while remaining the benefit on training efficiency. Moreover, assigning adaptive bandwidth to different parts by employing high-level understanding on scenes via extra supervisions, accelerating to support real-time training, or leveraging multiple key-frames would be valuable directions to improve the framework when applying it in real-word scenarios with extremely long sequences.

8 Conclusion

We propose a novel streaming radiance field method for effectively reconstructing and rendering dynamic scenes with explicit grid representations. The modelling of dynamic scenes is formalized as an incremental learning paradigm which allows the method for handling on-the-fly video sequences without need of recording scenes in advance. By virtue of narrow band tuning, our approach can achieve a fast training convergence. The storage cost induced by using explicit grids can be obviously reduced by performing difference-base compression. We also present an efficient training mechanism with pilot model guidance to further improve model optimization. Experiments demonstrate that our approach is capable of training a high-performing model for dynamic scenes with the speed of 15s for tuning every frame, achieving $1000\times$ speedup over the state-of-the-art implicit dynamic methods.

References

[1] Alvaro Collet, Ming Chuang, Patrick Sweeney, Don Gillett, Dennis Evseev, David J. Calabrese, Hugues Hoppe, Adam G. Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *international conference on computer graphics and interactive techniques*, 2015.

[2] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew DuVall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. 39(4):86:1–86:15, 2020.

[3] Yizhong Zhang, Jiaolong Yang, Zhen Liu, Ruicheng Wang, Guojun Chen, Xin Tong, and Baining Guo. Virtualcube: An immersive 3d video communication system. *CoRR*, abs/2112.06730, 2021.

[4] C. Lawrence Zitnick, Sing Bing Kang, Matthew T. Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *international conference on computer graphics and interactive techniques*, 2004.

[5] Heiko Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[6] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. *european conference on computer vision*, 2018.

[7] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[8] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arxiv CS.CV 2010.07492*, 2020.

[9] Ky Giang Nguyen and Dietmar Saupe. Rapid high quality compression of volume data for visualization. In *Computer Graphics Forum*, volume 20, pages 49–57. Wiley Online Library, 2001.

[10] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3d video synthesis. *arXiv: Computer Vision and Pattern Recognition*, 2021.

[11] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131*, 2021.

[12] James T. Kajiya and Brian Von Herzen. Ray tracing volume densities. *international conference on computer graphics and interactive techniques*, 1984.

[13] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020.

[14] Emanuele Trucco and Konstantinos Plakas. Video tracking: a concise survey. *IEEE Journal of oceanic engineering*, 31(2):520–529, 2006.

[15] Miki Haseyama, Takahiro Ogawa, and Nobuyuki Yagi. [survey paper] a review of video retrieval based on image and video semantic understanding. *ITE Transactions on Media Technology and Applications*, 1(1):2–9, 2013.

[16] Gal Lavee, Ehud Rivlin, and Michael Rudzsky. Understanding video events: A survey of methods for automatic interpretation of semantic occurrences in video. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(5):489–504, 2009.

[17] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 2, pages 142–149. IEEE, 2000.

[18] Geoffrey Hinton. RMSProp.

[19] Jean loup Gailly and Mark Adler. zlib, 2017.

[20] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics*, 2019.

[21] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael Cohen. The lumigraph. *international conference on computer graphics and interactive techniques*, 1996.

[22] Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael Cohen. Unstructured lumigraph rendering. *international conference on computer graphics and interactive techniques*, 2001.

[23] Marc Levoy and Pat Hanrahan. Light field rendering. In *ACM SIGGRAPH*, 1996.

[24] Heung-Yeung Shum and Li-Wei He. Rendering with concentric mosaics. In *ACM SIGGRAPH*, 1999.

[25] Jinxiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In *ACM SIGGRAPH*, 2000.

[26] Heung-Yeung Shum and Sing Bing Kang. Review of image-based rendering techniques. *visual communications and image processing*, 2000.

[27] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *ACM SIGGRAPH*, 2000.

[28] Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. Casual 3d photography. *ACM Transactions on Graphics*, 2017.

[29] Paul Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *ACM SIGGRAPH*, 1996.

[30] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. *Computer Vision and Pattern Recognition*, 2020.

[31] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. *Computer Vision and Pattern Recognition*, 2019.

[32] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew DuVall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics*, 2020.

[33] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *arXiv: Computer Vision and Pattern Recognition*, 2021.

[34] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022.

[35] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[36] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021.

[37] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien P. C. Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *ICCV*, 2021.

[38] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, 2021.

[39] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022.

[40] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. 2022.

[41] Anpei Chen, Zexiang Xu, Andreas Geiger, , Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields, 2022.

[42] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020.

[43] Takeo Kanade, Peter Rander, and P. J. Narayanan. Virtualized reality: constructing virtual worlds from real scenes. *IEEE MultiMedia*, 1997.

[44] Wojciech Matusik and Hanspeter Pfister. 3d tv: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *international conference on computer graphics and interactive techniques*, 2004.

[45] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv: Computer Vision and Pattern Recognition*, 2020.

[46] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. *Computer Vision and Pattern Recognition*, 2021.

[47] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. *arXiv: Computer Vision and Pattern Recognition*, 2021.

[48] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021.

[49] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*, 2021.

[50] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. 2021.

[51] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), 2021.

[52] Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. Matryodshka: Real-time 6dof video view synthesis using multi-sphere images. In *European Conference on Computer Vision*, pages 441–459. Springer, 2020.

[53] Aayush Bansal, Minh Vo, Yaser Sheikh, Deva Ramanan, and Srinivasa G. Narasimhan. 4d visualization of dynamic events from unconstrained multi-view videos. *Computer Vision and Pattern Recognition*, 2020.

[54] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *ICCV*, 2021.

[55] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM Trans. Graph.(ACM SIGGRAPH Asia)*, 2021.

[56] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We show and discuss the limitation on visual quality in the experimental section.
   (c) Did you discuss any potential negative societal impacts of your work? [No] There is no obvious potential negative societal impact for this work as the task 3D video synthesis is a general extension of traditional videos.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] There are no theoretical results in this paper.
   (b) Did you include complete proofs of all theoretical results? [N/A] There are no theoretical proofs in this paper.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide the code for reproducing.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use the public datasets presented in existing works.
   (b) Did you mention the license of the assets? [Yes] CC-BY-NC 4.0 for N3DV dataset
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our research code and a video for better showing qualitative results.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We use publicly datasets presented in existing works. We also use the data captured by our own.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] The data does not contain personally identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]