OD-NeRF: Efficient Training of On-the-Fly Dynamic Neural Radiance Fields

Zhiwen Yan Chen Li Gim Hee Lee
Department of Computer Science, National University of Singapore
{yan.zhiwen, lichen}@u.nus.edu gimhee.lee@nus.edu.sg

Abstract

Dynamic neural radiance fields (dynamic NeRFs) have achieved remarkable successes in synthesizing novel views for 3D dynamic scenes. Traditional approaches typically necessitate full video sequences for the training phase prior to the synthesis of new views, akin to replaying a recording of a dynamic 3D event. In contrast, on-the-fly training allows for the immediate processing and rendering of dynamic scenes without the need for pre-training on full sequences, offering a more flexible and time-efficient solution for dynamic scene rendering tasks. In this paper, we propose a highly efficient on-the-fly training algorithm for dynamic NeRFs, named OD-NeRF. To accelerate the training process, our method minimizes the training required for the model at each frame by using: 1) a NeRF model conditioned on multi-view projected colors, which exhibits superior generalization across multiple frames with minimal training ,and 2) a transition and update algorithm that leverages the occupancy grid from the last frame to sample efficiently at the current frame. Our algorithm can achieve an interactive training speed of 10FPS on synthetic dynamic scenes on-the-fly, and a 3×-9× training speed-up compared to the state-of-the-art on-the-fly NeRF on real-world dynamic scenes.

1. Introduction

Neural radiance fields (NeRF) [14] have recently emerged as a new 3D volumetric representation capable of synthesizing photo-realistic novel views from multi-view images. Many dynamic NeRFs are also proposed to reconstruct scenes with moving or deforming objects, using space-time 4D [10, 18] or explicit deformation [17, 20] representations. Nonetheless, a common limitation among these dynamic NeRF models is their focus on post-event reconstruction from full training video sequences, thereby restricting their use to replaying recorded dynamic event rather than enabling real-time rendering during the event itself.

StreamRF[8] diverges from these conventional joint training approaches by proposing an on-the-fly dynamic scene reconstruction algorithm; it trains the NeRF model concurrently with the ongoing capture of input frames. On-the-fly training of a dynamic NeRF fundamentally differs from the conventional post-event training approach. While post-event training aims to fit a 4D space-time radiance field using the complete training videos, on-the-fly training adapts the NeRF to represent the current radiance field, informed by both the previous reconstruction and the concurrently acquired video frames. As the dynamic event progresses, the model is updated with new frame information. Avoiding retraining from scratch for each frame, which would be prohibitively time-consuming, our approach focuses on exploiting the continuity inherent in the radiance fields of successive frames by using 1) a NeRF representation conditioned on multi-view projected colors for cross-frame generalizability, and 2) a transition and update to the occupancy grid used for efficient sampling.

Our proposed NeRF representation conditioned on multi-view projected colors is designed for better generalization across consecutive frames during on-the-fly training. Most of the existing dynamic NeRFs explicitly represent the motion using a temporal input dimension [10, 18] or a temporal deformation field [17, 20]. Due to the lack of direct supervision or known dynamics, these models with temporal input often extrapolate poorly to the unseen time and require extensive optimization for each new frame. In lieu of a temporal input, our model takes the projected color as input together with the spatial position. Since the projected color of a corresponding 3D point mostly remains constant in consecutive frames, the NeRF model generalizes better across frames even if the spatial position of the 3D point has change. Consequently, our proposed model has excellent temporal extrapolation capability and requires minimal optimization iterations per frame, leading to a considerable training acceleration.

Moreover, we introduce a method of transiting and updating the occupancy grid used for efficient point sampling. Since most of the 3D scenes are dominated with empty space, the occupancy grid has been used in static NeRFs [4, 15] to avoid redundant point sampling. However, this occupancy grid cannot be easily applied to on-the-fly train-

ing, as the occupancy at each incoming frame is always unknown. To tailor the occupancy grid to on-the-fly dynamic training, we treat it as a probability function reflecting the likelihood of the 3D voxels occupied by any entity. To model the motions in the scene, a transition function is applied to occupancy probability at the start of each frame optimization and later updated with new observations. The updated occupancy grid can then be used to sample points only in the occupied areas anytime during the on-the-fly training.

We evaluate our method on synthetic D-NeRF dataset [20], real-world MeetRoom [8], and DyNeRF [10] dataset. When trained and rendered on-the-fly, our method achieves substantial acceleration compared to the state-of-the-art algorithms while maintaining a comparable rendering quality. Particularly, our method can train 10 frames per second (FPS) on the synthetic D-NeRF dataset. We summarize **our contributions** as follows: 1) we propose a projected color-guided dynamic NeRF, facilitating efficient on-the-fly training with its temporal generalization ability. 2) We introduce a transition and update algorithm of occupancy grid designed for on-the-fly point sampling. 3) We achieve $10\times$ on-the-fly training and rendering acceleration in synthetic scenes and $3\times$-$9\times$ acceleration in real-world scenes compared to the state-of-the-art on-the-fly models.

2. Related Works

2.1. Accelerated Dynamic Neural Radiance Fields

In the area of Neural Radiance Fields (NeRFs), the initial achievements in representing static 3D scenes [14, 27, 33] have catalyzed many works [10, 17, 20, 31] directed towards adapting NeRFs for dynamic environments. Similar to the development of early static NeRF applications, these dynamic models typically require extensive training time for each scene, posing a significant challenge in efficiency.

Efforts to enhance training efficiency for dynamic NeRFs have often involved adapting strategies from their static counterparts. Innovations in this domain include the introduction of novel scene sampling and decomposition techniques in studies such as HyperReel[1] and NeRFPlayer[23]. These have notably improved rendering speeds, yet the training phase remains a time-intensive process. Further advancements have been achieved through the adoption of 4D Gaussian Splatting [29, 32], which offers enhanced rendering speeds. Nonetheless, the challenge of prolonged training durations persists. Other approaches, including TiNeuVox[6], k-planes[22], temporal interpolation[18], Im4D[12], and masked spatial-temporal hashing[26], leverage advanced 4D grids, hash grids or 6 planes to accelerate training. However, these methods are primarily designed for joint training scenarios and do not readily support flexible, on-the-fly training. Notably, StreamRF [8] stands as a singular approach, to the best of our knowledge, that accommodates on-the-fly training. This method utilizes the Plenoxel framework [33] to represent the initial scene frame, subsequently adapting this voxel grid to accommodate scene changes.

Our proposed methodology not only surpasses these models in per-frame training efficiency but also introduces the capability for on-the-fly training, making it particularly suitable for streaming applications.

2.2. Image Based Rendering

Instead of representing a 3D scene as a NeRF conditioned on spatial coordinates and viewing direction, some works rely on additional projected colors/features on the training views to improve generalization or robustness. IBRNet [28], MVSNeRF [3] and many following works [7, 13, 21] construct a cost volume of a dynamic scene based on the image features of the nearby views to learn a blending weights or the density and color output. These models are designed to be generalizable to unseen scenes and require prolonged pre-training on large datasets. LLFF [24, 25] aggregates features along the multi-view epipolar line to render view-dependent effect.

Recently, DynIBaR [11] applies this technique to dynamic NeRF by aggregating the projected image features across frames, after warpping a point using a motion trajectory learned from past and future frames. However, this method requires time-consuming optimization of the per-frame motion trajectory and cannot be applied to on-the-fly training as the future frames are not known. Similarly, Im4D [12] utilizes projected image features as color prediction guidance, but its geometry prediction still relies on temporal input only. This results in its slower training speed than our method in which both the geometry and color generalize across consecutive frames.

3. Problem Formulation

In this section, we first briefly describe the NeRF preliminaries necessary to understand our formulation. We then formally define the on-the-fly dynamic NeRF training.

NeRF Preliminaries. Static neural radiance fields [14] represent a 3D scene implicitly with a continuous volumetric function $F : (\mathbf{x}, \mathbf{d}) \mapsto (\sigma, \mathbf{c})$ that maps the spatial position $\mathbf{x} \in \mathbb{R}^3$ and viewing direction $\mathbf{d} \in \mathbb{R}^3$ to the volume density $\sigma \in \mathbb{R}$ and RGB color $\mathbf{c} \in \mathbb{R}^3$. To synthesize the pixel color $\hat{C}(\mathbf{r})$ on any 2D images, volume rendering is used to aggregate the color of N points with interval δ

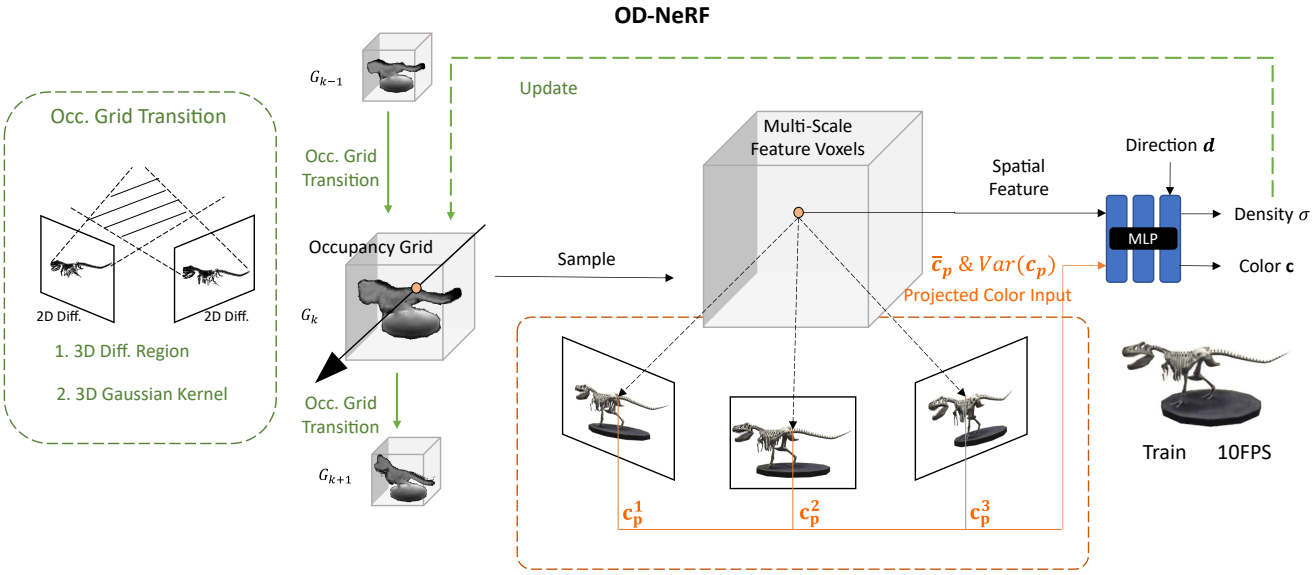

Figure 1. We introduce the on-the-fly training(left) of dynamic NeRFs and the OD-NeRF model(right). In on-the-fly training, the dynamic NeRF is trained based on the current and previous training frames to synthesize novel views for the current time step. Our OD-NeRF leverages the projected colors(orange arrow) for better cross-frame generalization and transition and update to the occupancy grid(green arrows) for efficient sampling.

along the ray $\mathbf{r}$ shooting from the pixel:

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

Dynamic NeRFs usually include an additional time dimension t in the input, through either a time-varying deformation field $D : (\mathbf{x}, t) \mapsto (\mathbf{x}')$ that maps the spatial coordinates $\mathbf{x}$ to its canonical space correspondence $\mathbf{x}'$ [17, 20], or directly expanding the 3D NeRF model to a 4D variant $F' : (\mathbf{x}, \mathbf{d}, t) \mapsto (\sigma, \mathbf{c})$ [10, 18]. During the training stage, all K frames of the training images $C_{0:K}(\mathbf{r})$ from all time $t_{0:K}$ are used to jointly minimize the rendering loss:

$$\mathcal{L} = \sum_{t=0}^{K}\sum_{\mathbf{r}\in\mathcal{R}} ||\hat{C}_t(\mathbf{r}) - C_t(\mathbf{r})||_2^2. \tag{2}$$

On-the-fly Dynamic NeRF. Although StreamRF[8] incorporates principles akin to on-the-fly training, the definition of this concept is not formally articulated within their work. In this section, we aim to provide a clear definition of on-the-fly training for dynamic NeRF. In contrast to the typical post-event joint training employed by the majority of dynamic NeRFs, on-the-fly training updates and renders the NeRF model concurrently with the acquisition of new training frames. Given the images up to the current time step $C_{0:k}$ and the radiance field estimated up to the last time step $F_{0:k-1}$, the goal of on-the-fly dynamic NeRF training is to find the radiance field function F_k at time t_k that minimizes the rendering loss at the current time step:

$$\begin{aligned} F_k(\mathbf{x}, \mathbf{d}) &= \underset{F_k}{\operatorname{argmax}}\, P(F_k \mid C_{0:k}, F_{0:k-1}) \\ &= \underset{F_k}{\operatorname{argmin}} \sum_{\mathbf{r}\in\mathcal{R}} ||\hat{C}_k(\mathbf{r}) - C_k(\mathbf{r})||_2^2. \end{aligned} \tag{3}$$

However, estimating the radiance field F_k at the current time step conditioned on all the previous images $C_{0:k}$ and radiance field $F_{0:k-1}$ is not scalable as time t_k increases. To mitigate this growth in complexity, we apply the first order Markov assumption to simplify the probability model by assuming conditional independence $F_k \perp\!\!\!\perp \{C_{0:k-1}, F_{0:k-2}\} \mid \{C_k, F_{k-1}\}$. This simplifies the estimation of the current radiance field as:

$$F_k = \underset{F_k}{\operatorname{argmax}}\, P(F_k \mid C_k, F_{k-1}). \tag{4}$$

Taking the radiance fields $F_{0:K}$ as hidden states and images $C_{0:K}$ as observations, we can formulate the on-the-fly training as the process of estimating the hidden states in a Hidden Markov model (HMM). The emission function $P(C_k \mid F_k)$ is the process of volumetric rendering that renders 2D images from 3D radiance fields. The transition function $P(F_k \mid F_{k-1})$ is the radiance field deformation or motion between two consecutive time steps.

In practice, this maximum likelihood estimation can be achieved by optimizing the last radiance field model F_{k-1} with the current training images C_k. The updated radiance

field is then rendered and evaluated on the test views at this time step.

Remarks. Note that we limit the focus of this work to dynamic scenes captured by multi-view forward-facing cameras based on realistic considerations. Although the reconstruction of dynamic scenes from a monocular camera is less demanding on the hardware, it requires the photographer to keep on moving the camera [16]. This is cumbersome in prolonged streaming scenarios. 360-degree inward-facing cameras can be used to reconstruct from all angles. Nonetheless, this often requires dozens of cameras and a much bigger space [19]. It is difficult for most streamers to acquire such professional setups. Consequently, we focus on scenes captured by static multi-view forward-facing cameras that are most aligned with the setups used in the current streaming industry. Some qualitative results with surrounding cameras are included in the supplementary.

4. Method

For highly efficient on-the-fly training, the model should require fewer number of optimization iterations at each frame and less training time for each iteration. To achieve these two objectives, we propose: 1) a dynamic NeRF guided by projected colors, and 2) an occupancy grid transition and update strategy(Fig. 1).

4.1. Dynamic NeRF Guided by Projected Colors

To accelerate training processes, it is crucial to minimize the number of required optimization iterations. Conventionally, dynamic NeRF models incorporating a temporal dimension often fail to extrapolate effectively to unseen time input, leading to poor generalization and the necessity for extensive iterative training upon receiving new frames.

In contrast, our approach seeks to replace the temporal input with multi-view projected colors, based on the observation that the projected color of a 3D point typically remains consistent across consecutive frames. This design allows the model to interpolate in the color dimension rather than to extrapolate in the temporal dimension. This framework inherently fosters a robust generalization capability of the model to new frames. Consequently, the model requires little to no training upon receiving new frames before rendering for this time step.

Delving into the model design, we replace the time input t in the space-time dynamic NeRF model $F(\mathbf{x}, \mathbf{d}, t)$ to the multi-view projected color mean $\bar{\mathbf{c}}_p(\mathbf{x}, k)$ and variance $\mathrm{Var}(\mathbf{c}_p(\mathbf{x}, k))$ at frame k to become:

$$F_k : (\mathbf{x}, \mathbf{d}, \bar{\mathbf{c}}_p(\mathbf{x}, k), \mathrm{Var}(\mathbf{c}_p(\mathbf{x}, k))) \mapsto (\sigma, \mathbf{c}). \quad (5)$$

The projected colors $\mathbf{c}_p(\mathbf{x}, k)$ are defined as the pixel colors of the training images $C_{k,\text{cam}}$ of all cameras $\mathcal{M}$ through 3D to 2D projection with camera projection matrix $\mathbf{P}_{\text{cam}}$:

$$\mathbf{c}_p(\mathbf{x}, k) = \{C_{k,\text{cam}}(\mathbf{P}_{\text{cam}} \cdot \mathbf{x}) \mid \text{cam} \in \mathcal{M}\}. \quad (6)$$

Figure 2. Comparative performance of traditional NeRF models versus our proposed method when extrapolating to the next frame without additional training. The purple color highlights areas present in the ground truth but missing in the rendered image, while the green color shows regions rendered but absent in the ground truth. The traditional models (left) exhibit a time lag and unclear details. Our method (right) demonstrates superior generalization with high fidelity and no observable lag.

To empirically validate our model's enhanced frame-to-frame generalization, we demonstrate a simple experiment as shown in Fig. 2. Both models are trained sequentially from frame 0 to frame $k-1$ and then tested on frame k, without any further training for this new frame. The results contrast the performance of our model with that of traditional NeRF models dependent on time inputs. Where the traditional model renders an image lagging behind the ground truth indicative of its poor temporal extrapolation, our model delivers accurate rendering based on its inherent generalization ability.

This distinction also reflects our model's capability from another perspective. Where existing NeRF models rely on extensive optimization with the new frames to utilize their information, our model is smartly designed to be directly conditioned on them through projection. This allows our model to harness the scene information contained in the new frames without much training.

4.2. Occupancy Grid Transition and Update

Occupancy grids are often used in static NeRFs to reduce the number of points sampled by caching whether a voxel is occupied. Formally, the occupancy grid is a 3D voxel grid $G = \{\max(\sigma(\mathbf{x})) \mid \forall \mathbf{x} \in \mathbb{V}_{\text{cur}}\}^3$, where $\max(\sigma(\mathbf{x}))$ represents the maximum volume density of all points in the respective voxel $\mathbb{V}_{\text{cur}}$. When sampling points on a ray, only points within the voxel above a certain volume density threshold are kept. It is obvious that this cannot be directly applied to dynamic scenes since the volume density of the 3D space changes over time. One way of applying this approach in dynamic scene reconstruction is to main-

tain a space-time 4D occupancy grid [18]. Unfortunately, this does not improve the sampling efficiency when training on-the-fly as the 3D occupancy at t_{k-1} cannot be used for sampling at t_k. When training the new frame k, the occupancy grid at the current time t_k is the same as being initialized from scratch.

Addressing the challenge of adapting occupancy grids for dynamic scenes, we propose a novel transition function that bridges the temporal gap between successive frames. We consider the occupancy grid at any time t_k as a 3D probability function $G_k = \{P(\max(\sigma_k(\mathbf{x})) > 0) \mid \forall \mathbf{x} \in \mathbb{V}_{\text{cur}}\}^3$, representing the chance of any point present in the voxel $mathbbV_{\text{cur}}$ with positive volume density at the current time step. Since the occupancy grid G_k is constantly updated throughout the per-frame training, we use G_k^j to denote the occupancy grid after j iterations of optimization where $j \in [0, J]$. At the start of each new frame optimization, we apply a transition function to this occupancy grid for the possible motions of the objects in the 3D space. The transition function comprises two components, catering to both minor and large motions within the scene.

Transition for Minor Motion The first component employs a 3D Gaussian kernel S that convolves with the previous occupancy grid G_{k-1}^J. For scenarios involving minimal movement, the spatial coordinates of a point at frame k are expected to be in close proximity to its location at frame $k-1$. In such cases, the convolution of the prior occupancy probabilities with a Gaussian kernel serves as a reliable estimator for the forthcoming frame's occupancy likelihood.

Transition for Large Motion The second component of the transition function is specifically designed to account for substantial movements. When an object undergoes significant motion between consecutive frames, there's often a notable change in the projected color of its previously and newly occupied area on the training images. By back-projecting the areas with significant pixel color variations $\Delta C_{k,\text{cam}}$ between frames $C_{k-1,\text{cam}}$ and $C_{k,\text{cam}}$ from the training views cam $\in \mathcal{M}$, we construct an irregular frustum $\Delta G_{k,\text{cam}}$ for each view, which encompasses the potential 3D region of movement:

$$\Delta G_{k,\text{cam}}(\mathbf{x}) = \Delta C_{k,\text{cam}}(\mathbf{P}_{\text{cam}} \cdot \mathbf{x}) \geq \mathbf{c}_{Th}. \quad (7)$$

The intersection of these frustums from multiple viewpoints yields a 3D difference region ΔG_k with high probability of containing motion, as shown in Fig. 3:

$$\Delta G_k = \bigcap_{\text{cam}} \Delta G_{k,cam}. \quad (8)$$

The transition function thus emerges from the union of this 3D difference region ΔG_k and the previous occupancy grid G_{k-1}^J:

$$G_k^0 = max(G_{k-1}^J, \Delta G_k). \quad (9)$$

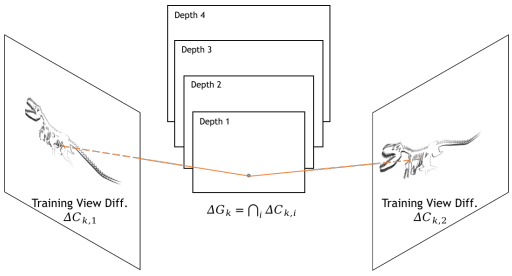

Figure 3. The 3D difference region is calculated by aggregating the difference in traning views using plane sweeping.

In practical implementation, the 3D difference region is determined through a majority vote across all training views rather than requiring unanimous agreement, to account for occlusions. This region is efficiently computed using a plane sweeping algorithm.

Occupancy Update After the transition function is applied to the occupancy grid at the start of the optimization for each frame, it needs to be updated with the new observations. Similar to the existing occupancy grid methods, we update the occupancy grid probabilities using the volume density output $\sigma(\mathbf{x})$ of the NeRF model of both randomly sampled points in each voxel as in [15] and points sampled on the ray during training. If the sampled density $\sigma(\mathbf{x})$ is higher than the current occupancy grid value $G_k(\mathbf{x})$, that occupancy grid value is updated to $\sigma(\mathbf{x})$.

Gradient Scaling and Dynamic-Static Fusion The 3D difference region, derived to accommodate large motions for the occupancy grid transition, can be applied to other techniques as well. For instance, it enables gradient downscaling for spatial features $f(\mathbf{x})$ in regions outside of the 3D difference region with a multiplier $0 < \alpha_g < 1$:

$$(\frac{\partial \mathcal{L}}{\partial f(\mathbf{x})})' = \frac{\partial \mathcal{L}}{\partial f(\mathbf{x})} \cdot (\Delta G_k(\mathbf{x}) + \alpha_g(1 - \Delta G_k(\mathbf{x}))). \quad (10)$$

Given that these regions are expected to exhibit minimal variation, scaling down the gradients in these regions can mitigate the occurrence of floater artifacts during the on-the-fly training process.

Additionally, this 3D difference region can be used in the dynamic-static fusion during rendering. When the rendering view remains static over a duration, it's only necessary to re-render the pixels outside projected area of the 3D difference region. This selective rendering significantly reduces computation time, while the pixels $\mathbf{u}$ outside this

zone retain their previously rendered color, ensuring continuity and efficiency:

$$\mathbf{c}_k(\mathbf{u}) = \begin{cases} \mathbf{c}_k(\mathbf{u}), & \text{if } \exists \mathbf{x}, (\mathbf{u} = \mathrm{P}_{cam} \cdot \mathbf{x}) \wedge \Delta G_k(\mathbf{x}) \\ \mathbf{c}_{k-1}(\mathbf{u}), & \text{otherwise} \end{cases}. \tag{11}$$

This mask can be efficiently generated using plane sweeping and 2D homography warping.

5. Experiments

We demonstrate the efficient on-the-fly training capability of our method on both synthetic and real-world novel view synthesis datasets. As explained in Section 3, our focus is on multi-view forward-facing camera configurations, which align closely with practical streaming use cases.

For the synthetic dataset, we employed the widely-recognized D-NeRF [20] dataset. To approximate a more pragmatic scenario, we created a modified version of this dataset featuring a forward-facing camera setup, instead of the unrealistic teleporting camera setup. This dataset will be made publicly available and more details are included in the supplementary. Beyond our primary experimental setup, we tested the limit of our model under additional challenging conditions. These scenarios including surrounding camera setup, forgetting problems and randomly missing camera feeds. The results of these auxiliary experiments are included in the supplementary.

For the real-world dataset, we evaluate on the MeetRoom [8] and DyNeRF [10] datasets, which are both captured with multi-view forward-facing cameras. All results of our method are reported for models trained with a single RTX3090 GPU. Some rendered videos are included in the supplementary. Although memory storage usage is not the focus of our paper, our model for the real-world scenes only require 90MB memory, comparable to the 100MB memory usage of the multi-view training images per frame.

5.1. Synthetic Dataset

On the synthetic dataset, we implement our method on top of the TiNeuVox [6] and evaluate the improvements in on-the-fly training speed and rendering quality. We also compare the performance of our method against the reported results of many other baseline models as shown in Tab. 1, but most of these models are trained jointly instead of trained on-the-fly. Our model is trained on the first frame for 200 iterations (for around 2 seconds), and then optimized for just 10 iterations per frame on-the-fly. Compared to the baseline methods, our model has significantly faster training speed of 10.5FPS instead of 1.27FPS, while achieving a superior rendering quality measured in PSNR and LPIPS [34].

As shown in Fig. 4, we also demonstrate some qualitative results of our method compared to TiNeuVox[6]. The images are rendered from slightly different camera angles to better illustrate the 3D geometry of the reconstructed scene. Our method not only achieves a $10\times$ training speed acceleration but also better rendering details. To better compare the performance of our proposed method with the baseline and various ablation methods, we present a plot showing their rendering quality (PSNR) against their training time in Fig. 5. Our proposed OD-NeRF model renders at better quality than TiNeuVox[6] at the same time constraint, or trains faster to achieve the same rendering quality. The improvements from the models without the two proposed component also suggest the effectiveness of both the projected color and the occupancy grid transition. More qualitative ablation results are included in the supplementary material.

Method	Train FPS↑	PSNR↑	LPIPS↓
D-NeRF[20]*	0.0013	30.50	0.07
K-Plane[22]*	0.04	31.67	NA
TiNeuVox-S[6]*	0.21	30.75	0.07
TiNeuVox-B[6]*	0.08	32.67	0.04
TempInterp[18]*	0.21	29.84	0.06
TiNeuVox[6]	1.27	30.57	0.07
Ours	**10.49**	**32.87**	**0.04**

*Result reported for joint training instead of on-the-fly training. NA if rendering speed not reported.

Table 1. Quantitative results of speed and novel view synthesis qualities on the D-NeRF dataset.

Method	Train Time (s)↓	Train Accel.	Eval PSNR↑
K-Plane*[22]	21.6	13.0x	31.63
HexPlane*[2]	24.0	14.5x	29.26
HyperReel*[1]	108.0	65.1x	31.1
NeRFplayer-INGP*[23]	66.0	39.8x	30.29
NeRFplayer-TRF*[23]	72.0	43.4x	30.69
4D-GS*[29]	24.0	14.5x	31.02
4K-4D*[30]	288.0	173.5x	32.86
Im4D*[12]	5.5	3.3x	32.58
Masked ST Hash*[26]	4.0	2.4x	**33.09**
StreamRF[8]	15.0	9.0x	28.26
InstantNGP[15]	11.5	6.9x	27.66
Ours	1.7	1.0x	28.31
No Color Proj.	**1.3**	0.8x	25.93
No Gauss Transit	1.6	1.0x	28.13
No Diff. Transit	1.7	1.0x	27.92
No Scene Fusion	2.1	1.2x	28.17
No Scale Grad.	1.5	0.9x	28.00

*Result reported for joint training instead of on-the-fly training.

Table 2. Quantitative results and ablation study on DyNeRF[10] dataset. "Train Accel." represents the training speed improvement of our method w.r.t. the method at each row.

Figure 4. Qualitative comparison between TiNeuVox and our method, rendered from random views instead of the ground truth view.

Method	Train Time (s)↓	Train Accel.	Eval PSNR↑
JaxNeRF*[5]	28380.0	9274.5x	27.11
Plenoxels[33]	840.0	274.5x	27.15
LLFF[25]	180.0	58.8x	22.88
StreamRF[8]	10.2	3.3x	26.72
InstantNGP[15]	22.2	7.3x	22.22
Ours	3.1	1.0x	**27.35**
No Color Proj.	3.0	1.0x	25.96
No Gauss Transit	3.1	1.0x	26.50
No Diff. Transit	**2.2**	0.7x	26.68
No Scene Fusion	3.5	1.1x	27.16
No Scale Grad.	3.2	1.0x	26.54

*Result reported for joint training instead of on-the-fly training.

Table 3. Quantitative results and ablation study on MeetRoom[8] dataset. "Train Accel." represents the training speed improvement of our method w.r.t. the method at each row.

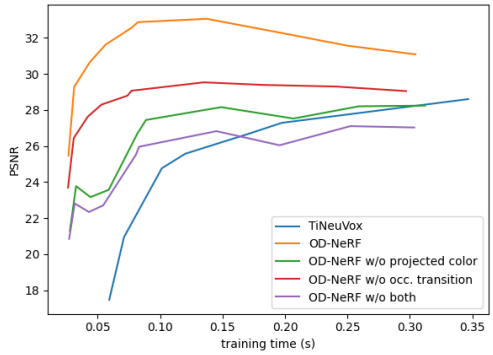

Figure 5. Ablation comparison of rendering quality with different training time.

5.2. Real-World Datasets

For the real-world datasets, our model is trained on the first frame for 6000 iterations and later 100 iterations per frame. We compare the training speed and rendering quality of our method implemented on top of the NerfAcc [9] implementation of InstantNGP [15] against various baseline models used for dynamic and static NeRFs. The performance of our model on the DyNeRF dataset is measured on the hardest "Flame Salmon" scene, following the evaluation proposed in the original paper[10]. Please note that many other works only report the performance on easier scenes[30], a mixture of scenes[29], or an unspecified set of scenes[23], so the rendering quality evaluation can be disadvantageous to our method.

Although some of the models (*e.g.* StreamRF [8]) do not claim the on-the-fly training ability, they are compatible with the on-the-fly training proposed in our paper. As shown in Tab. 3 and Tab. 2, our model can be trained on-the-fly significantly faster than the baseline models while maintaining a similar rendering quality. Other methods including HyperReel[1] and NeRFPlayer[23] are designed for fast rendering, but are trained at least 60× slower than our proposed method. Some previous methods including Im4D[12] and Masked ST Hash[26] designed for fast training, are incapable of on-the-fly training due to their 4D grid or hash grid design and are a few times slower than our model.

We also present some of the qualitative results of our model compared to StreamRF [8] and InstantNGP [15] as shown in Fig. 6 on MeetRoom [8] dataset and in Fig. 7 on DyNeRF [10] dataset. Our model is free of many common artifacts present in the rendered results of the baseline model while maintaining a significantly faster on-the-fly training speed.

5.3. Ablation

We present the ablation study of different components proposed in our method, including color projection, occupancy transition using Gaussian kernel and using 3D difference region, scene fusion and gradient scaling based on 3D different region in Tab. 2 and Tab. 2. All models are trained with the same number of iterations, so they train with a similar speed but produces different rendering quality. The color projection guidance contributes most to the rendering per-

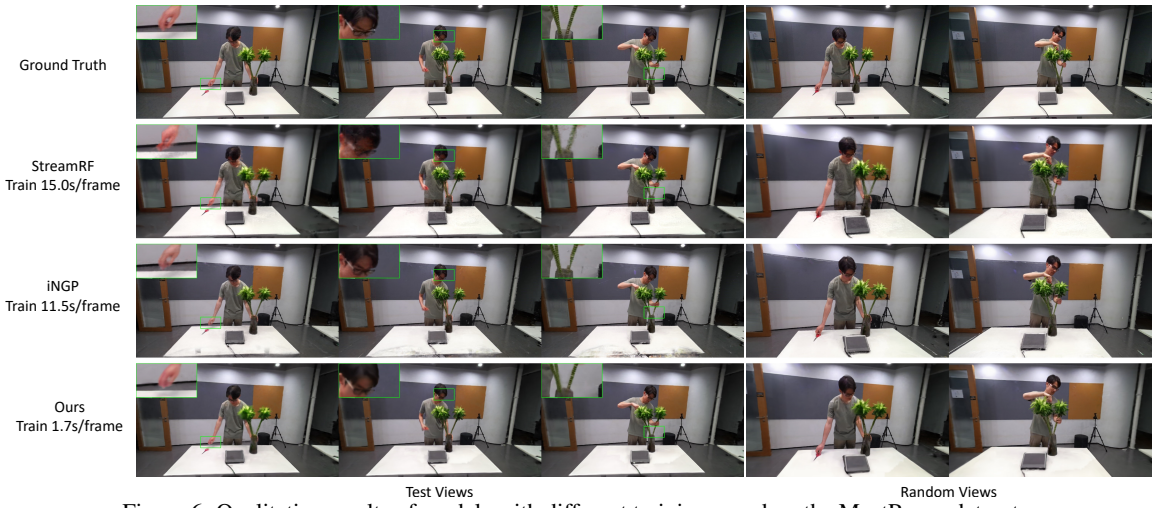

Figure 6. Qualitative results of models with different training speed on the MeetRoom dataset.

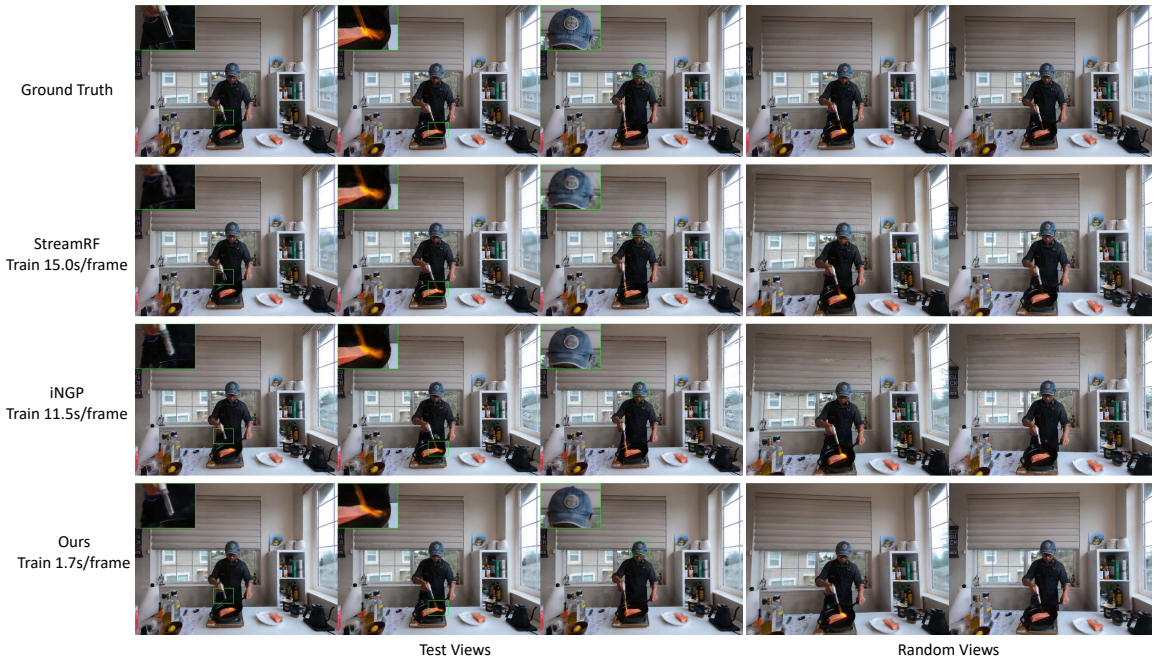

Figure 7. Qualitative results of models with different training speed on the DyNeRF dataset.

formance because of the cross frame generalization derived from it. The two types of transition methods contributes differently to different scenes. The scene fusion and gradient scaling also improve the overall rendering quality.

6. Limitations

The implicit correspondence of our projected color-guided NeRF relies on the relative invariance of the projected color of a point. However, this invariance can be violated with specular surfaces and occluded points. It may be possible to filter out the outlier projected colors caused by specularity and occlusion, or explicitly detect occlusion.

7. Conclusion

We introduced a new on-the-fly dynamic NeRF training setting, where the radiance field is trained and rendered frame by frame. To tackle the efficient on-the-fly training challenge, we propose a projected color-guided dynamic NeRF conditioned on the spatial and color input to efficiently optimize the radiance field with better cross-frane generalizability. We also propose a transition and update function to the occupancy grid for efficient point sampling in space. The experiment results in both synthetic and real-world datasets indicate the superior on-the-fly training speed of our method while maintaining a comparable rendering quality.

Acknowledgement. This work is supported by the Agency for Science, Technology and Research (A*STAR) under its MTC Programmatic Funds (Grant No. M23L7b0021).

References

[1] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16610–16620, 2023. 2, 6, 7

[2] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 6

[3] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021. 2

[4] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 1

[5] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020. 7

[6] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 2, 6

[7] Xin Huang, Qi Zhang, Ying Feng, Xiaoyu Li, Xuan Wang, and Qing Wang. Local implicit ray function for generalizable radiance field representation. *arXiv preprint arXiv:2304.12746*, 2023. 2

[8] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. Streaming radiance fields for 3d video synthesis. *arXiv preprint arXiv:2210.14831*, 2022. 1, 2, 3, 6, 7

[9] Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: Efficient sampling accelerates nerfs. *arXiv preprint arXiv:2305.04966*, 2023. 7

[10] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022. 1, 2, 3, 6, 7

[11] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. *arXiv preprint arXiv:2211.11082*, 2022. 2

[12] Haotong Lin, Sida Peng, Zhen Xu, Tao Xie, Xingyi He, Hujun Bao, and Xiaowei Zhou. Im4d: High-fidelity and real-time novel view synthesis for dynamic scenes. *arXiv preprint arXiv:2310.08585*, 2023. 2, 6, 7

[13] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 210–227. Springer, 2022. 2

[14] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2

[15] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 1, 5, 6, 7

[16] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 4

[17] 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. 1, 2, 3

[18] Sungheon Park, Minjung Son, Seokhwan Jang, Young Chun Ahn, Ji-Yeon Kim, and Nahyup Kang. Temporal interpolation is all you need for dynamic neural radiance fields. *arXiv preprint arXiv:2302.09311*, 2023. 1, 2, 3, 5, 6

[19] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *CVPR*, 2021. 4

[20] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. pages 10318–10327, 2021. 1, 2, 3, 6

[21] Yufan Ren, Fangjinhua Wang, Tong Zhang, Marc Pollefeys, and Sabine Süsstrunk. Volrecon: Volume rendering of signed ray distance functions for generalizable multi-view reconstruction. *arXiv preprint arXiv:2212.08067*, 2022. 2

[22] Sara Fridovich-Keil and Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 2, 6

[23] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742, 2023. 2, 6, 7

[24] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. In *European Conference on Computer Vision*. Springer, 2022. 2

[25] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Light field neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8269–8279, 2022. 2, 7

[26] Feng Wang, Zilong Chen, Guokang Wang, Yafei Song, and Huaping Liu. Masked space-time hash encoding for efficient dynamic scene reconstruction. *arXiv preprint arXiv:2310.17527*, 2023. 2, 6, 7

[27] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021. 2

[28] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 2

[29] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 2, 6, 7

[30] Zhen Xu, Sida Peng, Haotong Lin, Guangzhao He, Jiaming Sun, Yujun Shen, Hujun Bao, and Xiaowei Zhou. 4k4d: Real-time 4d view synthesis at 4k resolution. 2023. 6, 7

[31] Zhiwen Yan, Chen Li, and Gim Hee Lee. Nerf-ds: Neural radiance fields for dynamic specular objects. *arXiv preprint arXiv:2303.14435*, 2023. 2

[32] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 2

[33] 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. 2, 7

[34] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6