Neural 3D Video Synthesis

TIANYE LI∗†, University of Southern California, USA and Facebook Reality Labs Research, USA
MIRA SLAVCHEVA∗, Facebook Reality Labs Research, USA
MICHAEL ZOLLHOEFER, Facebook Reality Labs Research, USA
SIMON GREEN, Facebook Reality Labs Research, USA
CHRISTOPH LASSNER, Facebook Reality Labs Research, USA
CHANGIL KIM, Facebook, USA
TANNER SCHMIDT, Facebook Reality Labs Research, USA
STEVEN LOVEGROVE, Facebook Reality Labs Research, USA
MICHAEL GOESELE, Facebook Reality Labs Research, USA
ZHAOYANG LV, Facebook Reality Labs Research, USA

Fig. 1. We propose a novel method for representing and rendering high quality 3D video. Our method trains a novel type of dynamic neural radiance field (DyNeRF) in an efficient way. Our representation is very compact (28 MB for 10 s 30 FPS video captured from 18 views) and can be rendered from novel viewpoints continuously in space and time. Our method demonstrates near photorealistic dynamic novel view synthesis in space and time for complex scenes including challenging scene motions and strong view-dependent effects. This figure contains embedded animations that only play in Adobe Reader or KDE Okular.

We propose a novel approach for 3D video synthesis that is able to represent multi-view video recordings of a dynamic real-world scene in a compact, yet expressive representation that enables high-quality view synthesis and motion interpolation. Our approach takes the high quality and compactness of static neural radiance fields in a new direction: to a model-free, dynamic setting. At the core of our approach is a novel time-conditioned neural radiance field that represents scene dynamics using a set of compact latent codes. To exploit the fact that changes between adjacent frames of a video are typically small and locally consistent, we propose two novel strategies for efficient training of our neural network: 1) An efficient hierarchical training scheme, and 2) an importance sampling strategy that selects the next rays for training based on the temporal variation of the input videos.

In combination, these two strategies significantly boost the training speed, lead to fast convergence of the training process, and enable high quality results. Our learned representation is highly compact and able to represent a 10 second 30 FPS multi-view video recording by 18 cameras with a model size of just 28MB. We demonstrate that our method can render high-fidelity wide-angle novel views at over 1K resolution, even for highly complex and dynamic scenes. We perform an extensive qualitative and quantitative evaluation that shows that our approach outperforms the current state of the art. We include additional video and information at: https://neural-3d-video.github.io.


Additional Key Words and Phrases: 3D Video Synthesis, Dynamic Scenes, Neural Radiance Fields, Novel View Synthesis, Neural Rendering.
1 INTRODUCTION

Photorealistic representation and rendering of dynamic real-world scenes are highly challenging research topics, with many important applications that range from movie production to virtual and augmented reality. Dynamic real-world scenes are notoriously hard to model using classical mesh-based representations, since they often contain thin structures, semi-transparent objects, specular surfaces, and topology that constantly evolves over time due to the often complex scene motion of multiple objects and people.

In theory, the 6D plenoptic function $P(x, d, t)$ gives an answer to this representation and rendering problem, since it completely explains our visual reality and enables rendering every possible view, at every moment in time [Adelson and Bergen 1991]. Here, $x \in \mathbb{R}^3$ is the position of the camera in 3D space, $d = (\theta, \phi)$ specifies the viewing direction, and $t$ is time. We model color based on three RGB channels. Thus, measuring the plenoptic function requires placing an omnidirectional camera at every position in space at every possible time. This is infeasible — instead we will constrain the problem to a subvolume of space-time and employ a multi-view camera rig to record only a subset of this function, while interpolating it across space and time within the subvolume.

Recent neural volume rendering approaches [Lombardi et al. 2019; Mildenhall et al. 2020] show a promising direction to reconstruct and render complex scenes with intricate geometry and viewpoint dependency from multi-view input. Neural radiance fields (NeRF) [Mildenhall et al. 2020] represents a static scene based on a Multi-Layer Perceptron (MLP) that maps a continuous 3D coordinate (camera position $x$ and viewing direction $d$) to local color and opacity estimates. This representation allows for high-fidelity reproduction of extremely complex real-world scenes that would pose significant challenges to commonly used representations and algorithms, while its continuous nature and compact memory footprint do not limit the resolution of the representations or final rendering. However, the ray casting, required both to train and to render a neural radiance field, involves hundreds of MLP evaluations for each ray. While this might be acceptable for a static snapshot of a scene, directly reconstructing a dynamic scene as a sequence of per-frame neural radiance fields would be prohibitive as both storage and training time increase linearly with time. For example, to represent a 10 second-long, 30fps multi-view video recording by 18 cameras, which we later demonstrate with our method, a per-frame NeRF would require about 15,000 GPU hours in training and about 1 GB in storage. More importantly, such obtained representations would only reproduce the world as a discrete set of snapshots, lacking any means to reproduce the world in-between. On the other hand, Neural Volumes [Lombardi et al. 2019] is able to handle dynamic objects and even renders at interactive frame rates. Its limitation is the underlying dense uniform voxel grid that limits the resolution and/or size of the reconstructed dynamic scene due to the inherent $O(n^3)$ memory complexity. This approach is restricted to modeling single objects in isolation and cannot be easily extended to an entire dynamic scene.

In this paper, we propose a novel approach to the 3D video synthesis of complex, dynamic real-world scenes that enables high-quality view synthesis and motion interpolation while being compact. Videos typically consist of a time-invariant component under stable lighting and a continuously changing time-variant component. This dynamic component typically exhibits locally correlated geometric deformations and appearance changes between frames. By exploiting this fact, we propose to reconstruct a dynamic neural radiance field based on two novel contributions.

First, we extend neural radiance fields to the space-time domain. Instead of directly using time as input, we parameterize scene motion and appearance changes by a set of compact latent codes that are simultaneously optimized during training. This results in a compact, continuous space-time representation that shares time-invariant information across the entire video. Compared to the more obvious choice of an additional ‘time coordinate’, the learned latent codes show more expressive power, allowing for recording the vivid details of moving geometry and texture. They also allow for smooth interpolation in time, which enables visual effects such as slow motion or ‘bullet time’.

Second, we propose novel importance sampling strategies for dynamic radiance fields. Ray-based training of neural scene representations treats each pixel as an independent training sample and requires thousands of iterations to go through all pixels observed from all views. However, captured dynamic video often exhibits a small amount of pixel change between frames. This opens up an opportunity to significantly boost the training progress by selecting the pixels that are most important for training. Specifically, in the time dimension, we schedule training with coarse-to-fine hierarchical sampling in the frames. We first train our model until convergence using a subset of selected keyframes. Afterwards, we employ the keyframe model to initialize the training on the full video sequence. In the ray/pixel dimension, our design tends to sample those pixels that are more time-variant than others. In particular, we propose a global and a two-frame temporal difference importance sampling strategy. These strategies allow us to shorten the training time of long sequences significantly, while retaining high quality reconstruction results.

We demonstrate our approach using a multi-view rig based on 18 GoPro cameras. We show results on multiple challenging dynamic environments with highly complex view-dependent and time-dependent effects. Our method achieves photorealistic continuous novel-view rendering in space and time, which enables various cinematic effects like bullet-time and slow-motion. Compared to the naive per-frame NeRF baseline, we show that with our combined temporal and spatial importance sampling we achieve one order of magnitude acceleration in training speed, with a model that is 40 times smaller in size for 10 seconds of a 30 FPS 3D video.

In summary, we make the following technical contributions:

- We propose a novel dynamic neural radiance field that achieves high quality 3D video synthesis of complex, dynamic real-world scenes. Our approach of jointly learning temporal latent codes allows for high-quality view synthesis and motion interpolation. Our representation is a compact version of the 6D plenoptic function within the chosen subvolume.
We present novel training strategies based on hierarchical training and importance sampling in the spatiotemporal domain, which boost training speed significantly and lead to higher quality results for longer sequences.

2 RELATED WORK

Our work is closely related to several research domains, such as novel view synthesis for static scenes, 3D video synthesis for dynamic scenes, image-based rendering, and neural rendering approaches. For a detailed discussion of neural rendering applications and neural scene representations, we refer to the survey of Tewari et al. [2020].

2.1 Novel View Synthesis for Static Scenes

Novel view synthesis for static scenes is a long-standing and well-explored problem. Techniques in this category enable interpolation in the spatial domain and aim to capture a single moment of time.

**Geometry-based Approaches.** Novel view synthesis has been tackled by explicitly reconstructing textured 3D models of the scene that can be rendered from arbitrary viewpoints. Multi-view stereo [Fu-rukawa and Ponce 2009; Schönberger et al. 2016] and visual hull reconstructions [Esteban and Schmitt 2004; Laurentini 1994] have been successfully employed in this setting. Methods such as Kinect Fusion [Newcombe et al. 2011] enable capturing scenes using a monocular RGB-D camera. For modeling appearance, textures are acquired by fusing images from multiple views [Waechter et al. 2014]. Complex viewpoint-dependent effects such as for specular or translucent objects can be captured by light transport acquisition methods [Debevec et al. 2000; Wood et al. 2000]. Learning-based methods have been proposed to relax the high number of required views or to accelerate the inference speed for geometry reconstruction [Gu et al. 2020; Kar et al. 2017; Yao et al. 2018] and appearance capture [Bi et al. 2020; Meka et al. 2019], or combined reconstruction techniques [Niemeier et al. 2020; Yariv et al. 2020]. Shih et al. [2020] decomposes a single RGB-D image into multiple layers, which allows for targeted inpainting, but this representation is severely limited in terms of viewpoint extrapolation.

**Image-based Rendering Approaches.** Novel view synthesis can also be achieved by reusing input image pixels with an implicit representation of geometry. Early works using this approach interpolate the viewpoints [Chen and Williams 1993]. The Light Field/Lumigraph method [Davis et al. 2012; Gortler et al. 1996; Levoy and Hanrahan 1996] resamples input image rays to generate novel views. These representations enabled photorealistic rendering of the recorded scene from varying viewpoints without explicit or accurate estimation of the geometry and in a computationally efficient manner. However, dense sampling of the representations is required for high quality rendering of complex scenes, which presented a practical difficulty in its adoption and even more so when they have to be sampled temporally as well. Debevec et al. [1996] employs view-dependent textures. More recently, [Flynn et al. 2019; Kalantari et al. 2016; Mildenhall et al. 2019; Srinivasan et al. 2019; Zhou et al. 2018] learn to fuse and resample pixels from reference views using neural networks.

**Neural Rendering Approaches.** Neural Radiance Fields (NeRFs) [Mildenhall et al. 2020] train an MLP-based radiance and opacity field and achieve state-of-the-art quality for novel view synthesis. Other approaches [Meshrfy et al. 2019; Wiles et al. 2020] employ an explicit point-based scene representation combined with a screen space neural network for hole filling. Lassner and Zollhöfer [2020] push this even further and encode the scene appearance in a differentiable sphere-based representation. Methods such as Sitzmann et al. [2019] employ a dense voxel grid of features in combination with a screen space network for view synthesis.

All these methods are excellent at interpolating views for static scenes, but it is unclear how to extend them to the dynamic setting.

2.2 3D Video Synthesis for Dynamic Scenes

Techniques in this category enable view synthesis for dynamic scenes and might also enable interpolation across time.

**Model-based Approaches.** For video synthesis, it is also possible to explicitly capture geometry and textures, which was pioneered by Kanade et al. [1997]. Reconstruction and animation is particularly well studied for humans [Carranza et al. 2003; Guo et al. 2019; Starck and Hilton 2007], but is usually performed model-based and/or only works with high-end capture setups. Li et al. [2012] captured temporally consistent surfaces by tracking and completion. Collet et al. [2015] proposes a system for capturing and compressing streamable 3D video with high-end hardware.

**Neural Rendering Approaches.** More recently, learning-based methods such as Huang et al. [2018] achieve volumetric video capture for human performances from sparse camera views. Bansal et al. [2020] focus on more general scenes. They decompose them into a static and dynamic component, re-project information based on estimated coarse depth, and employ a U-Net in screen space to convert the intermediate result to realistic imagery. Neural Volumes [Lombardi et al. 2019] employs volume rendering in combination with a view-conditioned decoder network to parameterize dynamic sequences of single objects. Braxton et al. [2020] enable 6DoF video for VR applications based on independent alpha-textured meshes that can be streamed at the rate of hundreds of Mb/s. This approach employs a capture setup with 46 cameras and requires a large training dataset to construct a strong scene-prior. In contrast, we seek a unified space-time representation that enables continuous viewpoint and time interpolation, while being able to represent an entire multi-view video sequence of 10 second in as little as 28MB.

**Image-based Rendering.** Several image-based rendering methods have also been used for 3D video capture. Zitnick et al. [2004] propose a temporal layered representation that can be compressed and replayed at an interactive rate. Bemana et al. [2020] uses a neural network for space-time and illumination interpolation. Yoon et al. [2020] use single-view depth estimation for each image, followed by a model-based step for merging the depth maps to a unified representation that can be rendered from novel views. Most of these related methods use a warp field to model motion and require additional supervision from either depth or optical flow estimation. In
contrast, our method is model-free and does not require additional supervision or regularization.

We propose an approach for complex dynamic real-world scenes that is compact, yet expressive and enables high-quality view synthesis and motion interpolation.

2.3 Efficient Representations for 3D Video

One of the major issues of previous 3D video methods, for example [Broxton et al. 2020], is the long training/encoding time. Per-frame optimization of NeRF [Mildenhall et al. 2020] also would require a long training time. In the realm of traditional 2D video encoding, temporal redundancy is thoroughly studied and applied for designing state-of-the-art video codecs [Wiegand et al. 2003]. Due to geometry and viewpoint-dependent appearance changes and high dimensionality of the data (6D in our case), it is not trivial to adapt these strategies to 3D video.

**Sampling-based Methods.** An alternative to exploiting temporal redundancy is subsampling and supersampling to increase the efficiency. DeepFovea [Kaplanyan et al. 2019] uses foveated rendering to reduce the amount of computation required to render a scene by sampling of the scene according to the density of perceived rays. In contrast, Xiao et al. [2020] render full images at a lower resolution and uses statistical models to artificially and efficiently increase the resolution of the result. Müller et al. [2019] use a neural network to generate samples for Monte Carlo integration, reducing the rendering time dramatically. Sampling has also been applied directly to the plenoptic function for higher efficiency [Chai et al. 2000].

**Sparsity-based Methods.** Instead of sampling a representation, it is also possible to enforce the sparsity of the representation by design. This can be achieved for light field photography by using over-complete dictionaries and optimized projections [Marwah et al. 2013]. Neural Sparse Voxel Fields [Liu et al. 2020] represent the scene in a sparse voxel octree to accelerate rendering. Li et al. [2020b] proposes a DeepMPI representation to achieve reconstructions of scenes with varying reflectance and illumination.

We consider these works as orthogonal efforts to our work. Our method treats each ray/pixel as a separate training sample. This gives us the opportunity to explore sampling strategies for more efficient training, but also to go beyond that: We explore the idea of hierarchical training and ray importance sampling along rays, and these strategies not only accelerate the training progress, but at the same time also improve the rendering quality.

2.4 Non–Peer Reviewed Extensions of NeRF

MLP-based neural scene representations are currently one of the hot topics in computer graphics and vision research. There are several extensions of NeRF [Mildenhall et al. 2020] that are currently not peer reviewed. We recognize their efforts in improving neural radiance fields for in-the-wild scenes [Martin-Brual et al. 2020], generalization across scenes [Schwarz et al. 2020; Trevithick and Yang 2020; Yu et al. 2020], for non-rigid reconstruction [Du et al. 2020; Li et al. 2020a; Park et al. 2020; Pumarola et al. 2020; Tretschk et al. 2020; Xian et al. 2020], for rigid moving object reconstruction [Yuan et al. 2021], efficient rendering and training [Boss et al. 2020; Lindell et al. 2020; Rebain et al. 2020; Tancik et al. 2020b], and more suitable configurations and implementations [Google 2020; Zhang et al. 2020].

3 DyNeRF: DYNAMIC NEURAL RADIANCE FIELDS

We address the problem of reconstructing dynamic 3D scenes from time-synchronized input videos from multiple cameras with known intrinsic and extrinsic parameters. The representation we aim to reconstruct from such multi-camera recordings should allow us to render photorealistic images from a wide range of viewpoints at arbitrary time.

Building on NeRF [Mildenhall et al. 2020], we achieve this goal with dynamic neural radiance fields (DyNeRF) that are directly optimized from input videos captured with multiple video cameras. DyNeRF is a novel continuous space-time neural radiance field representation, controllable by a series of temporal latent embeddings that are jointly optimized during training. Our representation compresses a huge volume of input videos from multiple cameras to a compact 6D representation that can be queried continuously in both space and time. The learned embedding faithfully captures detailed temporal variations of the scene, such as complex photometric and topological changes, without explicit geometric tracking.

**Representation.** The problem of representing 3D video comprises of learning the 6D plenoptic function that maps a 3D position \( x \in \mathbb{R}^3 \), direction \( d \in \mathbb{R}^2 \), and time \( t \in \mathbb{R} \), to RGB radiance \( c \in \mathbb{R}^3 \) and opacity \( \sigma \in \mathbb{R} \). NeRF [Mildenhall et al. 2020] approximates the 5D plenoptic function of a static scene with a learnable function:

\[
F_\Theta : (x, d) \rightarrow (c, \sigma),
\]

realized by a Multi-Layer Perceptron (MLP) with trainable weights \( \Theta \). NeRF has been shown to synthesize high-fidelity novel views for static scenes; however, it is non-trivial to extend it to dynamic 3D scenes.

A potential solution would be to add a time dependency to the function:

\[
F_\Theta : (x, d, t) \rightarrow (c, \sigma).
\]

The 1-dimensional time variable \( t \) can be mapped via positional encoding [Tancik et al. 2020b] to a higher dimensional space, in a
manner similar to how NeRF handles the inputs \( x \) and \( d \). However, we empirically found that it is challenging for this design to capture complex dynamic 3D scenes with challenging topological changes and time-dependent volumetric effects, such as flames. We will employ this model as one of the baselines in our evaluation.

**Dynamic Neural Radiance Fields.** Inspired by DeepSDF [Park et al. 2019], we model the dynamic scene by latent codes \( z_t \in \mathbb{R}^D \), as shown by Fig. 2. In practice, we learn a set of time-dependent latent codes, indexed by a discrete time variable \( t \):

\[
F_Θ : (x, d, z_t) \rightarrow (c, \sigma) .
\]

We found that the latent codes resulted in a compact representation of the state of a dynamic scene at a certain time, which can handle a variety of complex scene dynamics and radiance changes implicitly. We apply positional encoding to the input position coordinates to map them to a higher-dimensional vector, using a series of sinusoidal functions [Tancik et al. 2020b]. No positional encoding is applied to the dynamic neural radiance field. Given a ray \( r(s) = o + sd \) with the origin \( o \) and direction \( d \) defined by the specified camera pose and intrinsics, the rendered color of the pixel corresponding to this ray \( C(r) \) is an integral over the radiance weighted by accumulated opacity [Mildenhall et al. 2020]:

\[
C(t)(r) = \int_{s_n}^{s_f} T(s) \sigma(r(s), z_t) e(r(s), d, z_t)) \, ds .
\]

Here, the accumulated opacity \( T(s) = \exp(-\int_{s_n}^{s} \sigma(r(p), z_t)) \, dp) \), and \( s_n \) and \( s_f \) denote the bounds of the volume depth range. The quadrature approximates evaluating radiance and opacity from discrete samples along the rays. We apply a hierarchical sampling strategy with first stratified sampling on the coarse level followed by importance sampling on the fine level, which is the same strategy used in [Mildenhall et al. 2020].

**Continuous Time Interpolation.** We train our dynamic neural radiance field from videos that capture appearance only at discrete time frames. To render at an arbitrary and continuous time, we linearly interpolate the neighboring latent codes that have been learned for the discrete time frames. We found that rendering with interpolated latent codes resulted in a smooth and plausible representation of dynamics between the two neighboring input frames. This enables rendering of special visual effects such as slow motion by interpolating sub-frame latent codes between two discrete time-dependent latent codes and the ‘bullet time’ effect with viewpoint-dependent effect by querying any latent code at any continuous time within the video.

**Loss Function.** The network parameters \( \Theta \) and the latent codes \( z_t \) are simultaneously trained by minimizing the \( \ell_2 \)-loss between the rendered colors \( \hat{C}(r) \) and the ground truth colors \( C(r) \), and summed over all rays \( r \) that correspond to the image pixels from all training camera views \( \mathcal{R} \) and throughout all time frames \( t \) of the recording:

\[
\mathcal{L} = \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{R}} \left( \left\| \hat{C}^{(t)}(r) - C^{(t)}(r) \right\|_2^2 + \left\| \hat{C}^{(t)}(r) - C^{(t)}(r) \right\|_2^2 \right) .
\]

We evaluate the loss at both coarse and fine level, respectively denoted by \( \hat{C}^{(t)} \) and \( C^{(t)} \), as in NeRF. We train with a stochastic version of this loss function, by randomly sampling ray data and optimizing on the loss of a ray batch. Please note our dynamic radiance field is trained with this plain \( \ell_2 \)-loss without any special regularization.

4 EFFICIENT TRAINING

One challenge of the ray casting–based neural rendering is the significant amount of training time. For example, training NeRF requires about 50 GPU hours for a single frame captured from about twenty 1K resolution images. It becomes infeasible to scale up this computation to training long 3D video sequences and achieve similar photorealistic quality.

The appearance changes between adjacent frames in natural videos are typically small and locally consistent. To explore how temporal redundancy can be exploited in the context of 3D video, we propose two strategies to accelerate the training process (see Fig. 3): (1) Hierarchical training that optimizes data over a coarse-to-fine frame selection (Sec. 4.1) and (2) importance sampling that prefers rays around regions of higher temporal variance (Sec. 4.2). These two strategies combined can be regarded as an adaptive sampling approach for reconstructing the 6D plenoptic function, contributing to significantly faster training and improved rendering quality.

4.1 Hierarchical Training

Keyframes are one of the important foundations of video compression techniques. We adapt this idea to our 3D video by firstly training on a set of keyframes and then training on all in-between frames jointly with the keyframes.

We first train a DyNeRF model on keyframes, which we sample equidistantly from the multi-view image sequence at fixed intervals \( K \). Once the model converges with keyframe supervision, we use it to initialize the final model, which has the same temporal resolution as the full video. To this end, we assign a fine-level latent embedding to every frame. Since the per-frame motion of the scene within each segment (divided by neighboring keyframes) is smooth, we initialize the fine-level latent embeddings by linearly interpolating between the coarse embeddings. Finally, we train using data from all the frames jointly, further optimizing the network weights and the latent embeddings. The coarse keyframe model has already captured an approximation of the time-invariant information across the video, which also appears in the in-between frames. Therefore, the fine full-frame training only needs to learn the time-variant information per-frame.

4.2 Ray Importance Sampling

Existing ray-based neural reconstruction methods, including NeRF, are trained on uniformly randomly sampled rays. The number of iterations they take in training per epoch scales linearly with the
We first normalize the weights of the rays across all input views for different importance based on the temporal variation in the week at a rate of 1 second per iteration. Given that each ray needs weight map of each ray at each time frame, based on the residual difference of its color from the global median of the same ray across time. We call this strategy importance sampling based on global median map using a heatmap (red and opaque means high probability).

However, for a natural video, a large proportion of the dynamic scene is either time-invariant or only contains a small radiance change at a particular timestamp across the entire observed video. This fact leads to a significant imbalance between the number of pixel observations and their contribution to the final representation in a uniform sampling setting. On one hand, the perceptual quality of attention using uniform sampling would be a waste of computational resources and time to achieve photorealistic 3D video.

To address this issue, we propose to sample the rays across time with different importance based on the temporal variation in the input videos. For each observed ray \( r \) at time \( t \), we compute a weight \( W^{(t)}(r) \). In each training iteration we pick a time frame \( t \) at random. We first normalize the weights of the rays across all input views for frame \( t \), and then apply inverse transform sampling to select rays based on these weights.

To calculate the weight of each ray, we propose two strategies based on two different insights. In the first one, we calculate the weight map of each ray at each time frame, based on the residual difference of its color from the global median of the same ray across time. We call this strategy importance sampling based on global median map (DyNeRF-ISG). In the other one, we consider two input color frames close in time and calculate the weight based on the residuals of the two. In a static multi-camera rig, this approximates the motion derivative between the frames, which we call importance sampling based on temporal difference (DyNeRF-IST). We explain the details of the two strategies below, assuming a static rig.

**Sampling Based on Global Median Maps (DyNeRF-ISG).** For each ground truth video, we first calculate the global median value of each ray across time \( \bar{C}(r) = \text{median}_{t \in T} C^{(t)}(r) \) and cache the global median image. During training, we compare each frame to the global median image and compute the residual. We choose a robust norm of the residuals to balance the contrast of weight. The norm measures the transformed values by a non-linear transfer function \( \psi(\cdot) \) that is parameterized by \( \gamma \) to adjust the sensitivity at various ranges of variance:

\[
W^{(t)}(r) = \frac{1}{3} \| \psi \left( \bar{C}^{(t)}(r) - C^{(t)}(r) : \gamma \right) \|_1.
\]

Here, \( \psi(x:y) = \frac{x^2}{\sqrt{x^2+y^2}} \) is the Geman-McLure robust function [Geman and McClure 1985] applied element-wise. Intuitively, a larger \( \gamma \) will lead to a high probability to sample the time-variant region, and \( \gamma \) approaching zero will approximate uniform sampling. \( \bar{C}(r) \) is a representative image across time, which can also take other forms such as a mean image. We empirically validated that using a median image is more effective to handle high frequency signal of moving regions across time, which helps us to approach sharp results faster during training.

**Sampling Based on Temporal Difference (DyNeRF-IST).** An alternative strategy, DyNeRF-IST, calculates the residuals by considering two nearby frames in time \( t_i \) and \( t_j \). In each training iteration we load two frames within a 25-frame distance, \( |t_i - t_j| \leq 25 \). In this strategy, we focus on sampling the pixels with largest temporal difference. We calculate the residuals between the two frames,
averaged over the 3 color channels
\[ W^{(t_i)}(r) = \min \left[ \frac{1}{3} \| C^{(t_i)} - C^{(t_j)} \|_1, \alpha \right]. \]  
(7)
To ensure that we do not sample pixels whose values changed due to spurious artifacts, we clamp \( W^{(t_i)}(r) \) with a lower-bound \( \alpha \), which is a hyper-parameter. Intuitively, a small value of \( \alpha \) would favor highly dynamic regions, while a large value would assign similar importance to all rays.

**Combined Method (DyNeRF-IS\(^*\)).** We empirically observed that training DyNeRF-ISG with a high learning rate leads to very quick recovery of dynamic detail, but results in some jitter across time. On the other hand, training DyNeRF-IST with a low learning rate produces a smooth temporal sequence which is still somewhat blurry. Thus, we combine the benefits of both methods in our final strategy, DyNeRF-IS\(^*\), which first obtains sharp details via DyNeRF-ISG and then smoothens the temporal motion via DyNeRF-IST.

5 EXPERIMENTS
We demonstrate our approach on a large variety of captured daily events, such as cooking, boiling water, playing with dogs. These contain challenging scene motions (topology changes, fast motions), varying illumination and self-cast shadows, view-dependent appearance (specularity and transparency) and highly volumetric effects (steam and flames). We also performed ablation studies, as well as quantitative and qualitative evaluations.

5.1 Datasets

**Multi-view Capture Setup.** We build a mobile multi-view capture system using action cameras. The system consists of 21 GoPro Black Hero 7 cameras. Fig. 5 shows the layout of all cameras. For all results discussed in this paper, we capture using the linear camera mode at a resolution of \( 2028 \times 2704 \) (2.7K) and frame rate of 30 FPS. We attach a timecode system to each camera. As a preprocessing step, we synchronize all the captured video streams given the recorded timestamps. During each capture we keep the camera rig fixed. All of the frames share the same intrinsic and extrinsic parameters for each of the views. For calibration, we take the first frame of the each video and calculate the intrinsic and extrinsic camera parameters using COLMAP [Schönberger and Frahm 2016]. We found that the GoPro linear FOV mode sufficiently well compensates for fisheye effects, thus we employ a pinhole camera model for all our experiments.

Our collected data can provide sufficient synchronized camera views for high quality 4D reconstruction in a natural daily indoor environment, which do not exist in public 4D datasets. Additionally, we captured the data with a focus on the dynamic scenes that contain challenging dynamics and significant amount of view-dependent effects which also rarely exists in other 4D datasets. We highlight challenges for our captures in the following.

**The Challenges in the Videos.** Our captured data demonstrates a variety of challenges for video synthesis, including objects of high specularity, translucency and transparency. It also contains scene changes and motions with changing topology (poured liquid), self-cast moving shadows, volumetric effects (fire flame), and an entangled moving object with strong view-dependent effects (the torch gun and the pan). We visualize one snapshot of the sequence in Fig. 4. In total, We trained our methods on a 60 seconds video sequence (`flame salmon`) in 6 chunks with each 10 seconds in length, and five other 10 seconds sequences captured at different time with different motion and lighting.

**Data Split.** We hold out the top center camera for testing, and use the rest of the cameras for training. For each captured multi-view sequence, we removed a particular camera if the time synchronization did not work or the illumination was inconsistent from a particular view. In the end, for all the results we employ a subset of 18 camera views for training, and 1 view for quantitative evaluation. We calculate a continuous interpolated spiral trajectory based on the 18 camera views, which we employ for qualitative novel view evaluation.

5.2 Implementation Details
In the following, we provide more details, such as the network architecture and training hyperparameters.

**Network Architecture.** Our implementation of neural radiance fields follows the implementation of NeRF [Mildenhall et al. 2020]. We use the same multi-layer perceptron (MLP) architecture except that we use 512 activations for the first 8 MLP layers instead of 256. We employ 1024-dimensional latent codes. We will discuss the choice of network width and latent code embedding size in the following ablation study. For each timestamp, the input to the MLP is a concatenation of the positionally-encoded position and the corresponding latent code. Same as [Mildenhall et al. 2020], we concatenate positional encoded direction with features after the 8 MLP layers, passing it to two additional MLP layers and predict the final output.

**Training.** We implement our approach in PyTorch. In the hierarchical training we first only train on keyframes that are \( K = 30 \) frames apart. We apply global median map importance sampling (DyNeRF-ISG) in both the keyframe training and full video training stage, and subsequently refine with temporal derivative importance sampling only for the full video. For faster computation in DyNeRF-ISG we calculate temporal median maps and pixel weights for each view at \( \frac{1}{4} \)th of the resolution, and then upsample the median image map to the input resolution. For \( y \) in the Geman-McClure robust norm, we set \( 1e^{-3} \) during keyframe training, and \( 2e^{-2} \) in the full video training stage. Empirically, this samples the background more densely in the keyframe training stage than for the following full video training. We also found out that using importance sampling makes a larger impact in the full video training, as keyframes are highly different. We set \( \alpha = 0.1 \) in DyNeRF-IST. We train the radiance field with an \( L_2 \) loss between the ground truth images and the rendered images without any special regularization terms. We employ the Adam optimizer [Kingma and Ba 2015] with parameters \( \beta_1 = 0.9 \) and \( \beta_2 = 0.999 \). In the keyframe training stage, we set a learning rate of \( 5e^{-4} \) and train for 300K iterations. In the full video training stage we first train for 250K iterations of DyNeRF-ISG with
Fig. 4. Frames from our captured multi-view video flame salmon sequence (top). We use 18 camera views for training (downsized on the right), and held out the upper row center view of the rig as novel view for quantitative evaluation. We captured sequences at different physical locations, time, and under varying illumination conditions. Our data shows a large variety of challenges in photorealistic 3D video synthesis.

Fig. 5. Our multi-view capture setup using synchronized GoPro Black Hero 7 cameras.

learning rate $1 \times 10^{-4}$ and then for another 100K iterations of DyNeRF-IST with learning rate $1 \times 10^{-5}$. The total training takes about a week with 8 NVIDIA V100 GPUs and a total batch size of 24576 rays.

**Latent Code Optimization.** In each iteration, we keep the norm of each latent code to be 1 by normalizing the vectors by their norm. We set the latent code learning rate to be $10 \times$ higher than for the other network parameters. Following the implementation of DeepSDF [Park et al. 2019], the latent codes are initialized from $\mathcal{N}(0, 0.01 / \sqrt{D})$, where $D = 1024$ and we initialize the latent codes for the different frames independently.

5.3 Evaluation Settings

We evaluated DyNeRF trained with three different importance sampling strategies: 1) DyNeRF-ISG refers to DyNeRF trained with importance sampling using residuals compared to global median map, 2) DyNeRF-IST refers to DyNeRF trained with importance sampling using the residuals from a temporal difference of two nearby frames, and 3) DyNeRF-IS★ refers to training with DyNeRF-ISG first followed by DyNeRF-IST. Unless otherwise specified, the results in our accompanying video are achieved with our full strategy DyNeRF-IS★ and will also refer to it as DyNeRF.

**Baselines.** We compare our approach to the following baselines:

- NeRF-T: Refers to the version in Eq. 2, which is a straightforward temporal extension of NeRF. We implement it following the details in Mildenhall et al. [2020], with only one difference in the input. The input concatenates the original positionally-encoded location, view direction, and time. We choose the positionally-encoded bandwidth for the time variable to be 4 and we do not find that increasing the bandwidth further improves results.

- DyNeRF†: We compare to DyNeRF without our proposed hierarchical training strategy and without importance sampling, i.e. this strategy uses per-frame latent codes that are trained jointly from scratch.

- DyNeRF with varying hyper-parameters: We vary the dimension of the employed latent codes (8, 64, 256, 1024, 8192).

**Metrics.** We evaluate the rendering quality using our held-out test view and the following quantitative metrics: peak signal-to-noise ratio (PSNR); structural dissimilarity index measure (DSSIM), which remaps the structural similarity index (SSIM) [Wang et al. 2004] into the range $[0, 1]$ by the formula $\text{DSSIM}(x, y) = (1 - \text{SSIM}(x, y)) / 2$ for pixel $(x, y)$ [Sara et al. 2019; Upchurch et al. 2016]; perceptual quality measure LPIPS [Zhang et al. 2018]; FLIP [Andersson et al. 2020]; and MSE. Higher PSNR scores indicate better
reconstruction quality. For all other metrics, lower numbers indicate better quality. Considering the significant amount of rendering time required for all models at high resolution, we perform the evaluation in the following way. For any video of length shorter than 60 frames, we evaluate the model frame-by-frame on the complete video, and for any video of length equal or longer than 300 frames, we evaluate the model every 10 frames. We verified on 2 video sequences with a frame length of 300 that the PSNR differs by at most 0.02 comparing evaluating them every 10th frame vs. on all frames. We evaluate all
We demonstrate our novel view rendering results on different settings. We will also provide a discussion about the effects of the latent code sampling strategy. DyNeRF-ISG uses global median map to guide importance sampling, while DyNeRF-IST uses temporal two-frame difference. DyNeRF-IS* uses both sampling strategies and thus runs for more iterations: 250K iterations of ISG, followed by 100K of IST; it is shown here only for completeness.

Table 1. Quantitative comparison of our proposed method to baselines at 200K iterations on a 10-second sequence. NeRF-T uses a per-frame input timestamp that goes through positional encoding. DyNeRF† represents training from scratch with per-frame latent codes without hierarchical strategy. DyNeRF-ISG, DyNeRF-IST and DyNeRF-IS* represent variants of our proposed hierarchical training with difference only in importance sampling strategy. DyNeRF-ISG uses global median map to guide importance sampling, while DyNeRF-IST uses temporal two-frame difference. DyNeRF-IS* uses both sampling strategies and thus runs for more iterations: 250K iterations of ISG, followed by 100K of IST; it is shown here only for completeness.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR ↑</th>
<th>MSE ↓</th>
<th>DSSIM ↓</th>
<th>LPIPS ↓</th>
<th>FLIP ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeRF-T</td>
<td>28.4487</td>
<td>0.00144</td>
<td>0.0228</td>
<td>0.1000</td>
<td>0.1415</td>
</tr>
<tr>
<td>DyNeRF†</td>
<td>28.4994</td>
<td>0.00143</td>
<td>0.0234</td>
<td>0.0985</td>
<td>0.1455</td>
</tr>
<tr>
<td>DyNeRF-ISG</td>
<td>29.4623</td>
<td>0.00113</td>
<td>0.0201</td>
<td>0.0854</td>
<td>0.1375</td>
</tr>
<tr>
<td>DyNeRF-IST</td>
<td>29.7161</td>
<td>0.00107</td>
<td>0.0197</td>
<td>0.0885</td>
<td>0.1340</td>
</tr>
<tr>
<td>DyNeRF-IS*</td>
<td>29.5808</td>
<td>0.00110</td>
<td>0.0197</td>
<td>0.0832</td>
<td>0.1347</td>
</tr>
</tbody>
</table>

We also highlight the models at 1K resolution, and report the average of the result from every evaluated frame.

Supplemental Video. We strongly recommend the reader to also watch our supplemental video to better judge the photorealism of our approach, which cannot be represented well by the metrics. Particularly, we find the metrics are not good indicators for the quality in the moving regions, including the sharp details of the moving region’s appearance and the natural and smooth motion transition across frames.

5.4 Discussion of the Results

We demonstrate our novel view rendering results on different sequences in Fig. 6. We demonstrate that our method can represent a 30FPS multi-view video of up to 10 seconds in length with high quality. With the reconstructed model, we can enable near photorealistic continuous novel-view rendering at 1K resolution. Please refer to our supplementary video for the 3D video visualizations.

In the following, we discuss the benefits of our method with respect to the axes of quality, compression, and training time. We will also provide a discussion about the effects of the latent code size and different importance sampling strategies.

Quantitative Comparison to the Baselines. Tab. 1 shows the quantitative comparison of our methods to the baselines. We train all baselines and our method the same number of iterations for fair comparison. Compared to the time-variant NeRF baseline NeRF-T and our basic DyNeRF model without our proposed training strategy (DyNeRF†), our DyNeRF model variants trained with our proposed training strategy perform significantly better in all metrics. DyNeRF-ISG and DyNeRF-IST can both achieve high quantitative performance, with DyNeRF-IST slightly more favorable in terms of the metrics. Our complete strategy DyNeRF-IS* requires more iterations and is added to the table only for completeness.

Qualitative Comparison to the Baselines. We also highlight a visual comparison of our methods to the baselines in Fig. 7. This highlights the advantages of our approach in terms of photorealism that are not well quantified using the metrics. NeRF-T can only capture a blurry motion representation, which loses all appearance details in the moving regions and cannot capture view-dependent effects. Though DyNeRF† has a similar quantitative performance as NeRF-T, it has significantly improved visual quality in the moving regions compared to NeRF-T, but still struggles to recover the sharp appearance details. DyNeRF with our proposed training strategy, DyNeRF-ISG, DyNeRF-IST and DyNeRF-IS*, can recover sharp details in the moving regions, including the torch gun, which is highly specular, and the flames.

Comparisons in terms of Model Compressibility. Our model is compact in terms of model size. In Tab. 2, we compare our model DyNeRF to the alternatives in terms of storage size. Compared to the raw videos stored in different images, e.g., PNG or JPEG, our method is more than two orders of magnitude smaller. Compared to a highly compact 2D video codec (HEVC), which is used as the default video codec for the GoPro camera, our model is also 50 times smaller. It is worth noting that these compressed 2D representations do not provide a 6D continuous representation as we do. Though NeRF is a compact network for a single static frame, representing the whole captured video without dropping frames requires a stack of frame-by-frame reconstructed NeRF networks, which is more than 50 times larger in size compared to our single DyNeRF model.

Comparisons in terms of Training Time. Though we still require significant compute to train our models, our proposed method is highly efficient compared to alternative solutions. Training a NeRF model frame-by-frame is the only baseline that can achieve the same photorealism as DyNeRF. However, we find that training a single frame NeRF model to achieve the same photorealism requires about 50 GPU hours, which in total requires 15K GPU hours for a 30FPS video of 10 seconds length. Our method only requires 1.3K GPU hours for the same video, which reduces the required compute by one order of magnitude.

Table 2. Comparison in model storage size of our method (DyNeRF) to alternative solutions. All calculation are based on 10 seconds of 30 FPS videos captured by 18 cameras. For HEVC, we use the default GoPro 7 video codec. For JPEG, we employ a compression rate that maintains the highest image quality. For NeRF, we use a set of the original NeRF networks [Mildenhall et al. 2020] reconstructed frame by frame. For HEVC, PNG and JPEG, the required memory may vary within a factor of 3 depending on the video appearance. For NeRF and DyNeRF, the required memory is constant.

<table>
<thead>
<tr>
<th></th>
<th>HEVC</th>
<th>PNG</th>
<th>JPEG</th>
<th>NeRF</th>
<th>DyNeRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Size (MB)</td>
<td>1.406</td>
<td>21.609</td>
<td>3.143</td>
<td>1.080</td>
<td>28</td>
</tr>
</tbody>
</table>

Impact of latent embedding size on DyNeRF. To evaluate the impact of the latent-code dimension on the final results, we ran an ablation on latent code length on 60 continuous frames and present the results in Table 3. In this experiment, we do not include keyframe training or importance sampling. We ran the experiments until 300K iterations, which is when most models are starting to converge without significant further improvements to the scores. Note that with a code length of 8,192 we cannot fit the same number of samples in the GPU memory as in the other cases, so we report
Fig. 7. Qualitative comparisons of DyNeRF variants on one image of the sequence whose averages are reported in Tab. 1. From left to right we show the rendering by each method, then zoom onto the moving flame gun, then visualize DSSIM and FLIP for this region using the viridis colormap (dark blue is 0, yellow is 1, lower is better). Clearly, NeRF-T with timestamps is not capable of recovering the dynamic parts. DyNeRF† that uses latent codes without hierarchical training manages, but is blurry. The three hierarchical DyNeRF variants outperform these baselines: DyNeRF-ISG has sharper details than DyNeRF-IST, but DyNeRF-IST recovers more of the flame, while DyNeRF★ combines both of these benefits.

a score from a later iteration when roughly the same number of samples have been used. We use 4×16GB GPUs and network width 256 for the experiments with this short sequence.

From the metrics we clearly conclude that a code of length 8 is insufficient to represent the dynamic scene well. Moreover, we have visually observed that results with such a short code are typically blurry. With increasing latent code size, the performance also increases respectively, which however saturates at a dimension of 1024. A latent code size of 8192 has longer training time per iteration. Taking the capacity and speed jointly in consideration, we
Fig. 8. **Comparison of importance sampling strategies over training iterations.** All methods shown use hierarchical training of keyframes first, the difference is in the subsequent joint training of all frames. In Fig. 8a we compare no importance sampling (DyNeRF-noIS) to our global (DyNeRF-ISG) and two-frame (DyNeRF-IST) strategies over iterations. With importance sampling the moving flame gun is recovered earlier and with sharper details. The quality of both DyNeRF-ISG and DyNeRF-IST at 50 thousand iterations is already visibly better than DyNeRF-noIS at 100 thousand iterations. In Fig. 8b we compare the result of no importance sampling and our combined strategy DyNeRF-IS★ upon convergence, demonstrating much crisper details with our final method.
We have demonstrated high-quality results for representing and that our video importance sampling scheme is able to accelerate further decrease training time and to speed up rendering at test of training speed compared to the baseline approaches, training 2) While we already achieve a significant improvement in terms explicit motion modeling could help to further improve results. the training views is challenging and might lead to artifacts in the time are required. 3) Viewpoint extrapolation beyond the bounds of training for dynamic radiance fields.

6 LIMITATIONS

choose 1024 as our default latent code size for all the sequences in this paper and the supplementary video.

The impact of importance sampling in training. We show that our video importance sampling scheme is able to accelerate training for dynamic radiance fields.

In Figure 8 we evaluate the effect of our importance sampling strategies, DyNeRF-ISG, DyNeRF-IST and DyNeRF-IS*, against a baseline DyNeRF-noIS that also employs a hierarchical training strategy with latent codes initialized from trained keyframes, but instead of selecting rays based on importance, selects them at random like in standard NeRF [Mildenhall et al. 2020]. The figure shows zoomed-in crop-outs of the dynamic region for better visibility. We clearly see that all the importance sampling strategies manage to recover the moving flame gun better than DyNeRF-noIS in two times less iterations. At 100k iterations DyNeRF-ISG and DyNeRF-IST look similar, though they converge differently with DyNeRF-IST being blurrier in early iterations and DyNeRF-ISG managing to recover moving details slightly faster. The visualizations of the final results upon convergence in Fig. 8b demonstrate the superior photorealism that DyNeRF-IS* achieves, as DyNeRF-noIS remains much blurrier in comparison.

6 LIMITATIONS

We have demonstrated high-quality results for representing and rendering dynamic scenes using neural radiance fields. Nevertheless, our approach is subject to a few limitations. 1) Highly dynamic scenes with large and fast motions are challenging to model and learn, which might lead to blur in the moving regions. An adaptive sampling strategy during the hierarchical training that places more keyframes during the challenging parts of the sequence or more explicit motion modeling could help to further improve results. 2) While we already achieve a significant improvement in terms of training speed compared to the baseline approaches, training still takes a lot of time and compute resources. Finding ways to further decrease training time and to speed up rendering at test time are required. 3) Viewpoint extrapolation beyond the bounds of the training views is challenging and might lead to artifacts in the rendered imagery. We hope that, in the future, we can learn strong scene priors that will be able to fill in the missing information. 4) We discussed the importance sampling strategy and its effectiveness based on the assumption of videos observed from static cameras. We leave the study of this strategy on videos from moving cameras as future work. We believe these current limitations are good directions to explore in follow-up work and that our approach is a stepping stone in this direction.


