Dynamic View Synthesis from Dynamic Monocular Video

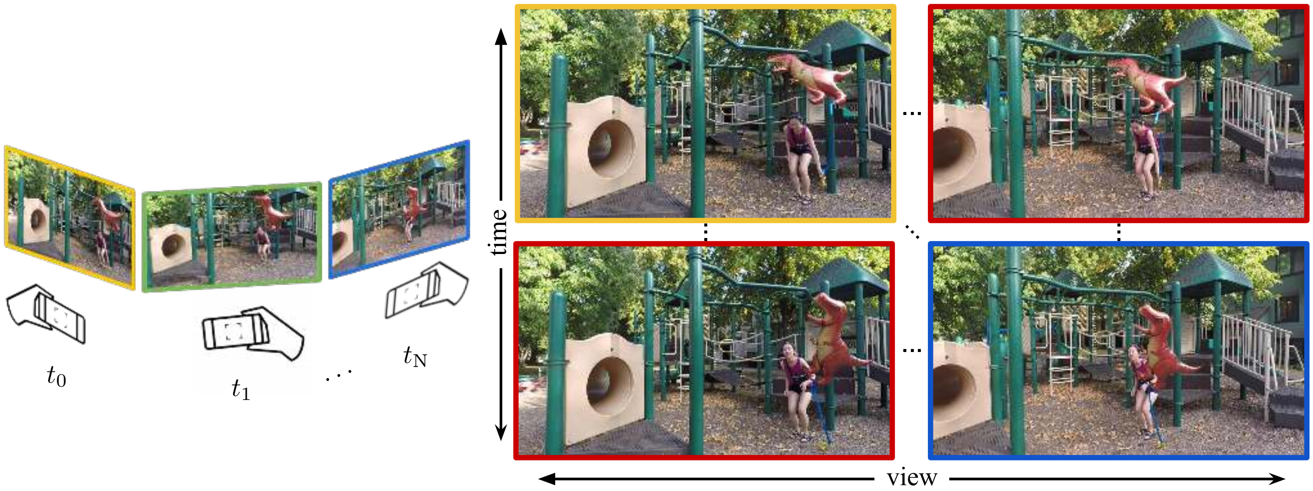
Chen Gao
Virginia Tech
chengao@vt.edu

Ayush Saraf
Facebook
ayush29feb@fb.com

Johannes Kopf
Facebook
jkopf@fb.com

Jia-Bin Huang
Virginia Tech
jbhuang@vt.edu

https://free-view-video.github.io



(a) Input: monocular video (b) Output: free-viewpoint rendering

Figure 1. **Dynamic view synthesis from dynamic monocular video.** Our method takes a monocular video as input (a). Each frame in the video is taken at a unique time step and from a different view (e.g., the yellow and blue frames). Our goal is to synthesize photorealistic novel views of a dynamic scene at arbitrary camera viewpoints and time steps (red frames). Such a system enables *free-viewpoint video*, providing immersive and almost life-like viewing experiences for users.

Abstract

We present an algorithm for generating novel views at arbitrary viewpoints and any input time step given a monocular video of a dynamic scene. Our work builds upon recent advances in neural implicit representation and uses continuous and differentiable functions for modeling the time-varying structure and the appearance of the scene. We jointly train a time-invariant static NeRF and a time-varying dynamic NeRF, and learn how to blend the results in an unsupervised manner. However, learning this implicit function from a single video is highly ill-posed (with infinitely many solutions that match the input video). To resolve the ambiguity, we introduce regularization losses to encourage a more physically plausible solution. We show extensive quantitative and qualitative results of dynamic view synthesis from casually captured videos.

1. Introduction

Video provides a window into another part of the real world. In traditional videos, however, the viewer observes the action from a fixed viewpoint and cannot navigate the scene. *Dynamic view synthesis* comes to the rescue. These techniques aim at creating photorealistic novel views of a dynamic scene at arbitrary camera viewpoints and time, which enables *free-viewpoint video* and *stereo rendering*, and provides an immersive and almost life-like viewing experience. It facilitates applications such as replaying professional sports events in 3D [7], creating cinematic effects like freeze-frame bullet-time (from the movie “The Matrix”), virtual reality [11, 5], and virtual 3D teleportation [37].

Systems for dynamic view synthesis need to overcome challenging problems related to video capture, reconstruction, compression, and rendering. Most of the existing methods rely on laborious and expensive setups such as custom fixed multi-camera video capture rigs [8, 61, 11, 37, 5]. While recent work relaxes some constraints and can handle *unstructured* video input (e.g., from hand-held cameras) [3, 4], many methods still require synchronous capture from multiple cameras, which is impractical for most people. Few methods produce dynamic view synthesis from a *single* stereo or even RGB camera, but they are limited to specific domains such as human performance capture [12, 19]. Recent work on depth estimation from monocular videos of dynamic scenes shows promising results [27, 58]. Yoon et al. [58] use estimated depth maps to warp and blend multiple images to synthesize an unseen target viewpoint. However, the method uses a *local* representation (i.e., per-frame depth maps) and processes each novel view *independently*. Consequently, the synthesized views are not consistent and may exhibit abrupt changes.

This paper presents a new algorithm for dynamic view synthesis from a dynamic video that overcomes this limitation using a *global* representation. More specifically, we use an implicit neural representation to model the time-varying volume density and appearance of the events in the video. We jointly train a time-invariant static neural radiance field (NeRF) [32] and a time-varying dynamic NeRF, and learn how to blend the results in an unsupervised manner. However, it is challenging for the dynamic NeRF to learn plausible 3D geometry because we have just *one and only one* 2D image observation at each time step. There are infinitely many solutions that can correctly render the given input video, yet only one is physically correct for generating photorealistic novel views. Our work focuses on resolving this ambiguity by introducing regularization losses to encourage plausible reconstruction. We validate our method's performance on the Dynamic multi-view dynamic scenes dataset by Yoon et al. [58].

The key points of our contribution can be summarized as follows:

- We present a method for modeling dynamic radiance fields by jointly training a time-invariant model and a time-varying model, and learn how to blend the results in an unsupervised manner.
- We design regularization losses for resolving the ambiguities when learning the dynamic radiance fields.
- Our model leads to favorable results compared to the state-of-the-art algorithms on the Dynamic Scenes Dataset.

2. Related Work

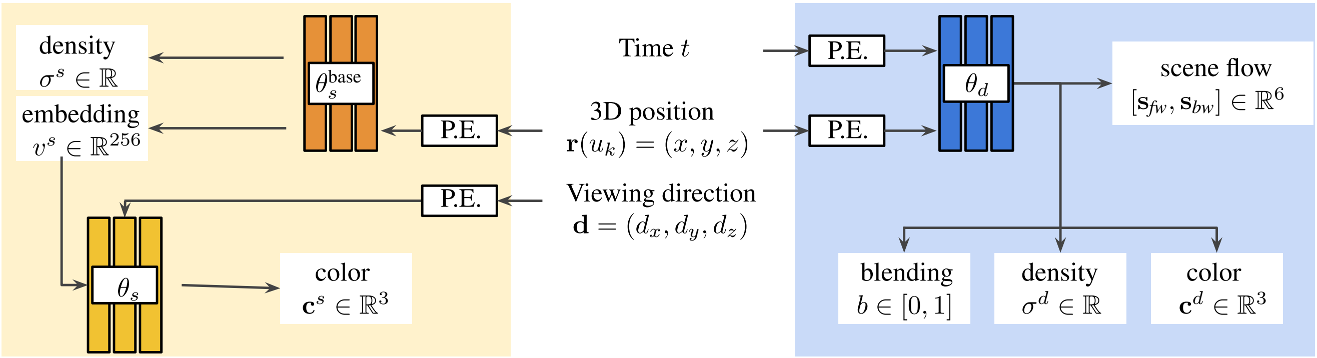
View synthesis from images. View synthesis aims to generate new views of a scene from multiple posed images [47]. Light fields [25] or Lumigraph [18] synthesize realistic appearance but require capturing and storing many views. Using explicit geometric proxies allows high-quality synthesis from relatively fewer input images [6]. However, estimating accurate scene geometry is challenging due to untextured regions, highlights, reflections, and repetitive patterns. Prior work addresses this via local warps [9], operating in the gradient domain [23], soft 3D reconstruction [41], and learning-based approaches [21, 15, 14, 20, 44]. Recently, neural implicit representation methods have shown promising view synthesis results by modeling the continuous volumetric scene density and color with a multilayer perceptron [32, 34, 57, 59].

Several methods tackle novel view synthesis from one single input image. These methods differ in their underlying scene representation, including depth [35, 53], multiplane images [52], or layered depth images [46, 24]. Compared with existing view synthesis methods that focus on *static objects or scenes*, our work aims to achieve view synthesis of *dynamic scenes* from one single video.

View synthesis for videos. Free viewpoint video offers immersive viewing experiences and creates freeze-frame (bullet time) visual effects [28]. Compared to view synthesis techniques for images, capturing, reconstructing, compressing, and rendering dynamic contents in videos is significantly more challenging. Many existing methods either focus on specific domains (e.g., humans) [8, 12, 19] or transitions between input views only [3]. Several systems have been proposed to support interactive viewpoint control watching videos of generic scenes [61, 11, 37, 4, 5, 1]. However, these methods require either omnidirectional stereo camera [1], specialized hardware setup (e.g., custom camera rigs) [61, 11, 5, 37], or synchronous video captures from multiple cameras [4]. Recently, Yoon et al. [58] show that one can leverage depth-based warping and blending techniques in image-based rendering for synthesizing novel views of a dynamic scene from a single camera. Similar to [58], our method also synthesizes novel views of a dynamic scene. In contrast to using explicit depth estimation [58], our implicit neural representation based approach facilitates geometrically accurate rendering and smoother view interpolation.

Implicit neural representations. Continuous and differentiable functions parameterized by fully-connected networks (also known as multilayer perceptron, or MLPs) have been successfully applied as compact, implicit representations for modeling 3D shapes [10, 55, 38, 17, 16], object appearances [36, 34], 3D scenes [48, 32, 40]. These methods train MLPs to regress input coordinates (e.g., points in 3D space) to the desired quantities such as occupancy value [30, 45, 40], signed distance [38, 2, 31], volume density [32], color [36, 48, 45, 32]. Leveraging differentiable rendering [50, 22], several recent works have shown training these MLPs with multiview 2D images (without using direct 3D supervision) [34, 56, 32].

Most of the existing methods deal with *static scenes*. Directly extending the MLPs to encode the additional time dimension does not work well due to 3D shape and motion


density
$\sigma^s \in \mathbb{R}$
embedding
$v^s \in \mathbb{R}^{256}$
θ_s^{base}
P.E.
P.E.
θ_s
color
$\mathbf{c}^s \in \mathbb{R}^3$
Time t
3D position
$\mathbf{r}(u_k) = (x, y, z)$
Viewing direction
$\mathbf{d} = (d_x, d_y, d_z)$
P.E.
P.E.
θ_d
scene flow
$[\mathbf{s}_{fw}, \mathbf{s}_{bw}] \in \mathbb{R}^6$
blending
$b \in [0, 1]$
density
$\sigma^d \in \mathbb{R}$
color
$\mathbf{c}^d \in \mathbb{R}^3$

(a) Static NeRF (Section 3.2)

(b) Dynamic NeRF (Section 3.3)

Figure 2. **Method overview.** We propose to use two different models to represent the (a) *static* and (b) *dynamic* scene components.**(a) Static NeRF**: For static components, we train a NeRF model following [32], but excluding all the pixels marked as dynamic. This allows us to reconstruct the background's structure and appearance without conflicting the moving objects. **(b) Dynamic NeRF**: Modeling a dynamic scene from a single video is highly ill-posed. To resolve the ambiguity, we leverage the multi-view constraints as follow: Our Dynamic NeRF takes both $\mathbf{r}(u_k)$ and t as input to predict 3D scene flow from time t to $t+1$ ($\mathbf{s}_{fw}$) and from time t to $t-1$ ($\mathbf{s}_{bw}$). Using the predicted scene flow, we can create a *warped* radiance field by resampling the radiance field modeled at the adjacent time instances and apply temporal consistency. Thus, at each instance, we can have multiple views associated with different time t to train the model.

entanglement. The method in [29] extends NeRF for handling crowdsourced photos that contain lighting variations and transient objects. Our use of static/dynamic NeRF is similar to [29], but we focus on modeling the *dynamic* objects (as opposed to *static* scene in [29]). The work that most related to ours is [33], which learns a continuous motion field over space and time. Our work is similar in that we also disentangle the shape/appearance and the motion for dynamic scene elements. Unlike [33], our method models the shape and appearance of a dynamic scene from a casually captured video *without* accessing ground truth 3D information for training.

Concurrent work on dynamic view synthesis. Very recently, several methods concurrently to ours have been proposed to extend NeRF for handling dynamic scenes [54, 26, 51, 39, 42]. These methods either disentangle the dynamic scenes into a canonical template and deformation fields for each frame [51, 42, 39] or directly estimate dynamic (4D spatiotemporal) radiance fields [54, 26]. Our work adopts the 4D radiance fields approach due to its capability of modeling large scene dynamics. In particular, our approach shares high-level similarity with [26] in that we also regularize the dynamic NeRF through scene flow estimation. Our method differs in several important technical details, including scene flow based 3D temporal consistency loss, sparsity regularization, and the rigidity regularization of the scene flow prediction. For completeness, we include experimental comparison with one template-based method [51] and one 4D radiance field approach [26].

3. Method

3.1. Overview

Our method takes as input (1) monocular video $\{\boldsymbol{I}_0, \boldsymbol{I}_1, \ldots, \boldsymbol{I}_{N-1}\}$ with N frames, and (2) a binary mask $\boldsymbol{M}$ of the foreground object for each frame. The mask can be obtained automatically via segmentation or motion segmentation algorithms or semi-automatically via interactive methods such as rotoscoping. Our goal is to learn a global representation that facilitates free-viewpoint rendering at arbitrary views and input time steps.

Specifically, we build on neural radiance fields (NeRFs) [32] as our base representation. NeRF models the scene implicitly with a continuous and differentiable function (i.e., an MLP) that regresses an input 3D position $\mathbf{x} = (x, y, z)$ and the normalized viewing direction $\mathbf{d} = (d_x, d_y, d_z)$ to the corresponding volume density σ and color $\mathbf{c} = (r, g, b)$. Such representations have demonstrated high-quality view synthesis results when trained with multiple images of a scene. However, NeRF assumes that the scene is *static* (with constant density and radiance). This assumption does not hold for casually captured videos of dynamic scenes.

One straightforward extension of the NeRF model would be to include *time* as an additional dimension as input, e.g., using 4D position (x, y, z, t) input where t denotes the index of the frame. While this model theoretically can represent the time-varying structure and appearance of a dynamic scene, the model training is highly ill-posed, given that we only have one single 2D image observation at each time step. There exist infinitely many possible solutions that match the input video exactly. Empirically, we find that directly training the "NeRF + time" model leads to low visual

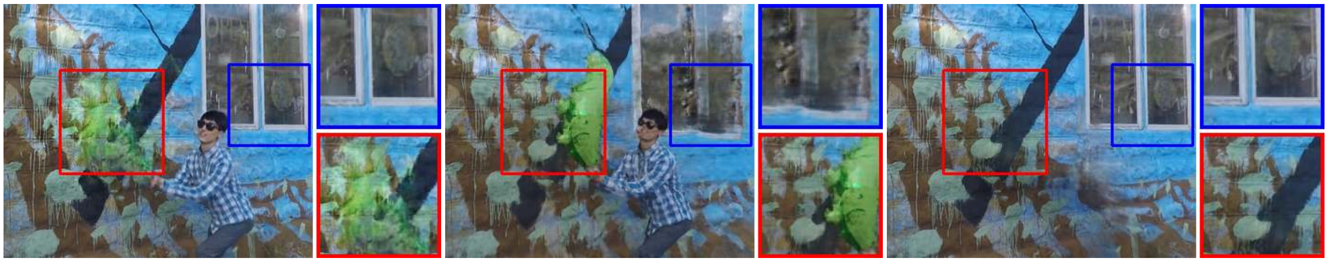

(a) NeRF (b) NeRF + time (c) Static NeRF (Ours)

Figure 3. **Why static NeRF?** NeRF [32] assumes that the scene is entirely static. (a) Directly training a NeRF model on a dynamic scene inevitably results in blurry reconstruction (even for the static regions of the scene). (b) One straightforward extension is to include time as an additional input dimension (NeRF + time). However, such a method suffers from ambiguity because the input video can be explained either with time-varying geometry or appearance or both. The representation reconstructs the input frames well but produces visual artifacts at novel views. (c) To tackle this issue, we model the static components of the scene using a static NeRF. We exclude all the pixels marked as "dynamic" from training the model. This allows us to accurately reconstruct the background's structure and appearance without conflicting the moving objects.

quality.

The key contribution of our paper lies in resolving this ambiguity for modeling the time-varying radiance fields. To this end, we propose to use different models to represent *static* or *dynamic* scene components using the user-provided dynamic masks.

For *static components* of the scene, we apply the original NeRF model [32], but exclude all "dynamic" pixels from training the model. This allows us to reconstruct the background's structure and appearance without conflicting reconstruction losses from moving objects. We refer to this model as "Static NeRF" (Figure 3).

For *dynamic components* of the scene (e.g., moving objects), we train an MLP that takes a 3D position and time (x, y, z, t) as input to model the volume density and color of the dynamic objects at each time instance. To leverage the multi-view geometry, we use the same MLP to predict the additional three-dimensional scene flow from time t to the previous and next time instance. Using the predicted forward and backward scene flow, we create a *warped* radiance field (similar to the backward warping 2D optical flow) by resampling the radiance fields implicitly modeled at time $t+1$ and $t-1$. For each 3D position, we then have up to three multi-view observations to train our model. We refer to this model as "Dynamic NeRF" (Figure 4). Additionally, our Dynamic NeRF predicts a blending weight and learns how to blend the results from both the static NeRF and dynamic NeRF in an unsupervised manner. In the following, we discuss the detailed formulation of the proposed static and dynamic NeRF models and the training losses for optimizing the weights for the implicit functions.

3.2. Static NeRF

Formulation. Our static NeRF follows closely the formulation in [32] and is represented by a fully-connected neural network. Consider a ray from the camera center $\mathbf{o}$ through a given pixel on the image plane as $\mathbf{r}(u_k) = \mathbf{o} + u_k \mathbf{d}$, where $\mathbf{d}$ is the normalized viewing direction, our static NeRF maps a 3D position $\mathbf{r}(u_k)$ and viewing direction $\mathbf{d}$ to volume density σ^s and color $\mathbf{c}^s$:

$$(\sigma^s, \mathbf{c}^s) = \text{MLP}_\theta\left(\mathbf{r}(u_k)\right), \tag{1}$$

where MLP$_\theta$ stands for two cascaded MLP, detailed in Figure 2. We can compute the color of the pixel (corresponding the ray $\mathbf{r}(u_k)$) using numerical quadrature for approximating the volume rendering interval [13]:

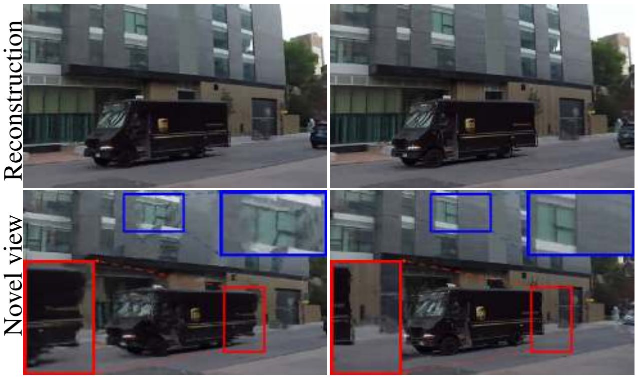
$$\mathbf{C}^s(\mathbf{r}) = \sum_{k=1}^{K} T^s(u_k)\, \alpha^s(\sigma^s(u_k)\,\delta_k)\, \mathbf{c}^s(u_k), \tag{2}$$

$$T^s(u_k) = \exp\left(-\sum_{k'=1}^{k-1} \sigma^s(u_k)\,\delta_k\right), \tag{3}$$

where $\alpha(x) = 1 - \exp(-x)$ and $\delta_k = u_{k+1} - u_k$ is the distance between two quadrature points. The K quadrature points $\{u_k\}_{k=1}^{K}$ are drawn uniformly between u_n and u_f [32]. $T^s(u_k)$ indicates the accumulated transmittance from u_n to u_k.

Static rendering photometric loss. To train the weights θ_s of the static NeRF model, we first construct the camera rays using all the pixels for all the video frames (using the associated intrinsic and extrinsic camera poses for each frame). Here we denote $\mathbf{r}_{ij}$ as the rays passing through the pixel j on image i with $\mathbf{r}_{ij}(u) = \mathbf{o}_i + (u)\mathbf{d}_{ij}$. We can then optimize θ_s by minimizing the *static rendering photometric loss* for all the color pixels $\mathbf{C}(\mathbf{r}_{ij})$ in frame $i \in \{0, \ldots, N-1\}$ in the static regions (where $\mathbf{M}(\mathbf{r}_{ij}) = 0$):

$$\mathcal{L}_{static} = \sum_{ij} \|(\mathbf{C}^s(\mathbf{r}_{ij}) - \mathbf{C}^{gt}(\mathbf{r}_{ij})) \cdot (1 - \mathbf{M}(\mathbf{r}_{ij}))\|_2^2 \tag{4}$$



(a) NeRF + time (b) Ours

Figure 4. **Why dynamic NeRF?** (*Top*) Since the training objective is to minimize the image reconstruction loss on the input video frames, NeRF + time explains the input frames very well. (*Bottom*) However, there are infinitely many solutions that can correctly render the given input video, yet only one of them is physically correct for generating photorealistic novel views. NeRF + time tries to disentangle view from time using time as additional input. However, the problem becomes under-constrained and leads to artifacts in both static and dynamic regions. Our dynamic NeRF produces plausible view synthesis results for moving objects.

3.3. Dynamic NeRF

In this section, we introduce our core contribution to modeling time-varying radiance fields using *dynamic NeRF*. The challenge lies in that we only have *one single* 2D image observation at each time instance t. So the training lacks multi-view constraints. To resolve this training difficulty, we predict the forward and backward scene flow and use them to create a *warped* radiance field by resampling the radiance fields implicitly modeled at time $t+1$ and $t-1$. For each 3D position at time t, we then have up to three 2D image observations. This multi-view constraint effectively constrains the dynamic NeRF to produce temporally consistent radiance fields.

Formulation. Our dynamic NeRF takes a 4D-tuple $(\mathbf{r}(u_k), t)$ as input and predict 3D scene flow vectors $\mathbf{s}_{\mathit{fw}}$, $\mathbf{s}_{\mathit{bw}}$, volume density σ^d, color $\mathbf{c}^d$ and blending weight b:

$$\left(\mathbf{s}_{\mathit{fw}}, \mathbf{s}_{\mathit{bw}}, \sigma^d_t, \mathbf{c}^d_t, b\right) = \mathrm{MLP}_{\theta_d}\left(\mathbf{r}(u_k), t\right) \tag{5}$$

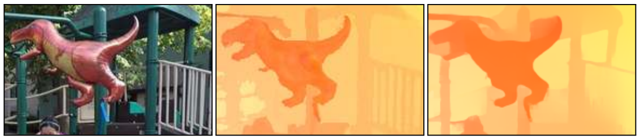
Using the predicted scene flow $\mathbf{s}_{\mathit{fw}}$ and $\mathbf{s}_{\mathit{bw}}$, we obtain the scene flow neighbors $\mathbf{r}(u_k)+\mathbf{s}_{\mathit{fw}}$ and $\mathbf{r}(u_k)+\mathbf{s}_{\mathit{bw}}$. We also use the predicted scene flow to *warp* the radiance fields from the neighboring time instance to the current time. For every 3D position at time t, we obtain the occupancy σ^d and color $\mathbf{c}^d$ through querying the same MLP model at $\mathbf{r}(u_k)+\mathbf{s}$:

$$\left(\sigma^d_{t+1},\, \mathbf{c}^d_{t+1}\right) = \mathrm{MLP}_{\theta_d}(\mathbf{r}(u_k)+\mathbf{s}_{\mathit{fw}},\, t+1) \tag{6}$$

$$\left(\sigma^d_{t-1},\, \mathbf{c}^d_{t-1}\right) = \mathrm{MLP}_{\theta_d}(\mathbf{r}(u_k)+\mathbf{s}_{\mathit{bw}},\, t-1) \tag{7}$$

For computing the color of a dynamic pixel at time t', we use the following approximation of volume rendering integral:

$$\mathbf{C}^d_{t'}(\mathbf{r}) = \sum_{k=1}^{K} T^d_{t'}(u_k)\, \alpha^d(\sigma^d_{t'}(u_k)\, \delta_k)\, \mathbf{c}^d_{t'}(u_k) \tag{8}$$



(a) Input (b) Induced flow (c) Estimated flow

Figure 5. **Scene flow induced optical flow.** We supervise the predicted scene flow by minimizing the endpoint error between the estimated optical flow [49] and our scene flow induced optical flow. Since we jointly train our model with *both* photometric loss and motion matching loss, our learned volume density helps render a more accurate flow than the estimated flow (e.g., the complex structures of the fence on the right).

Dynamic rendering photometric loss. Similar to the static rendering loss, we train the dynamic NeRF model by minimizing the reconstruction loss:

$$\mathcal{L}_{\mathit{dyn}} = \sum_{t' \in \{t,\, t-1,\, t+1\}} \sum_{ij} \left\|\left(\mathbf{C}^d_{t'}(\mathbf{r}_{ij}) - \mathbf{C}^{gt}(\mathbf{r}_{ij})\right)\right\|_2^2 \tag{9}$$

3.4. Regularization Losses for Dynamic NeRF

While leveraging the multi-view constraint in the dynamic NeRF model reduces the amount of ambiguity, the model training remains ill-posed without proper regularization. To this end, we design several regularization losses to constrain the Dynamic NeRF.

Motion matching loss. As we do not have direct 3D supervision for the predicted scene flow from the motion MLP model, we use 2D optical flow (estimated from input image pairs using [49]) as *indirect* supervision. For each 3D point at time t, we first use the estimated scene flow to obtain the corresponding 3D point in the reference frame. We then project this 3D point onto the reference camera so we can compute the *scene flow induced optical flow* and enforce it to match the estimated optical flow (Figure 5). Since we jointly train our model with both photometric loss and motion matching loss, the learned volume density helps render a more accurate flow than the estimated flow. Thus, we do not suffer from inaccurate optical flow supervision.

Motion regularization. Unfortunately, matching the rendered scene flow with 2D optical flow does not fully resolve all ambiguity, as a 1D family of scene flow vectors produces the same 2D optical flow (Figure 6). We regularize the scene flow to be *slow* and *temporally smooth*:

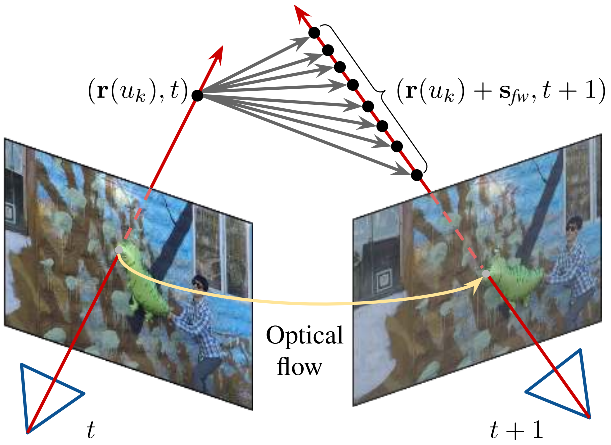


Figure 6. **Ambiguity of optical flow supervision.** Matching the scene flow induced optical flow with the estimated 2D optical flow does not fully resolve the ambiguity. There exists a 1D family of scene flow predictions that produce the same 2D optical flow.

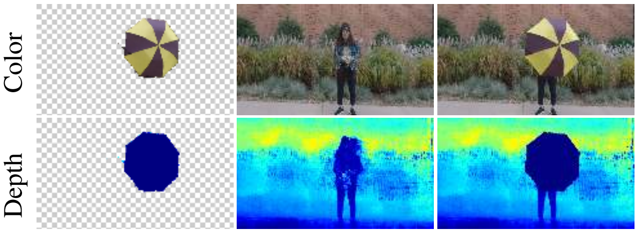
$$\mathcal{L}_{slow} = \sum_{ij} \left\| \mathbf{s}_{fw}(\mathbf{r}_{ij}) \right\|_1 + \left\| \mathbf{s}_{bw}(\mathbf{r}_{ij}) \right\|_1 \tag{10}$$

$$\mathcal{L}_{smooth} = \sum_{ij} \left\| \mathbf{s}_{fw}(\mathbf{r}_{ij}) + \mathbf{s}_{bw}(\mathbf{r}_{ij}) \right\|_2^2 \tag{11}$$

We further regularize the scene flow to be *spatially smooth* by minimizing the difference between neighboring 3D points' scene flow. To regularize the consistency of the scene flow, we have the scene flow cycle consistency regularization:

$$\begin{aligned}\mathcal{L}_{cyc} = \sum & \left\| \mathbf{s}_{fw}(\mathbf{r}, t) + \mathbf{s}_{bw}(\mathbf{r} + \mathbf{s}_{fw}(\mathbf{r}, t),\ t+1) \right\|_2^2 \\ & + \left\| \mathbf{s}_{bw}(\mathbf{r}, t) + \mathbf{s}_{fw}(\mathbf{r} + \mathbf{s}_{bw}(\mathbf{r}, t),\ t-1) \right\|_2^2 \end{aligned} \tag{12}$$

Sparsity regularization. We render the color using principles from classical volume rendering. One can see through a particle if it is partially transparent. However, one can not see through the scene flow because the scene flow is not an intrinsic property (unlike color). Thus, we minimize the entropy of the rendering weights $T^d \alpha^d$ along each ray so that few samples dominate the rendering.



(a) Dynamic NeRF (b) Static NeRF (c) Full model

Figure 7. **Full model rendering.** We compose the (a) dynamic and (b) static NeRF model into (c) our full model and render full frames at novel viewpoints and time steps.

Depth order loss. For a moving object, we can either interpret it as an object close to the camera moving slowly or an object far away moving fast. To resolve the ambiguity, we leverage the state-of-the-art single-image depth estimation [43] to estimate the input depth. As the depth estimates are up to shift and scale, we cannot directly use them to supervise our model. Instead, we use the robust loss as in [43] to constrain our dynamic NeRF, and further constrain our dynamic NeRF with our static NeRF. We additionally minimize the L2 difference between $\mathbf{D}^s$ and $\mathbf{D}^d$ for all the pixels in the static regions (where $\mathbf{M}(\mathbf{r}_{ij}) = 0$):

$$\begin{aligned}\mathcal{L}_{depth} = \sum_{ij} & \left\| \overline{\mathbf{D}^d}(\mathbf{r}_{ij}) - \overline{\mathbf{D}^{gt}}(\mathbf{r}_{ij}) \right\|_2^2 + \\ & \left\| (\mathbf{D}^d(\mathbf{r}_{ij}) - \mathbf{D}^s(\mathbf{r}_{ij})) \cdot (1 - \mathbf{M}(\mathbf{r}_{ij})) \right\|_2^2 ,\end{aligned}$$

where $\overline{\mathbf{D}}$ stands for the normalized depth.

3D temporal consistency loss. If an object remains unmoved for a while, the network can not learn the correct volume density and color of the *occluded background* at the current time because those 3D positions are omitted during volume rendering. When rendering a novel view, the model may generate holes for the occluded region. To address this issue, we propose the 3D temporal consistency loss *before* rendering. Specifically, we enforce the volume density and color of each 3D position to match its scene flow neighbors'. The correct volume density and color will then be *propagated* across time steps.

Rigidity regularization of the scene flow. Our model prefers to explain a 3D position by the static NeRF if this position has no motion. For static position, we want the blending weight b to be closed to 1. For a non-rigid position, the blending weight b should be 0. This learned blending weight can further constrain the rigidity of the predicted scene flow by taking the product of the predicted scene flow and $(1 - b)$. If a 3D position has no motion, the scene flow is forced to be zero.

3.5. Combined model

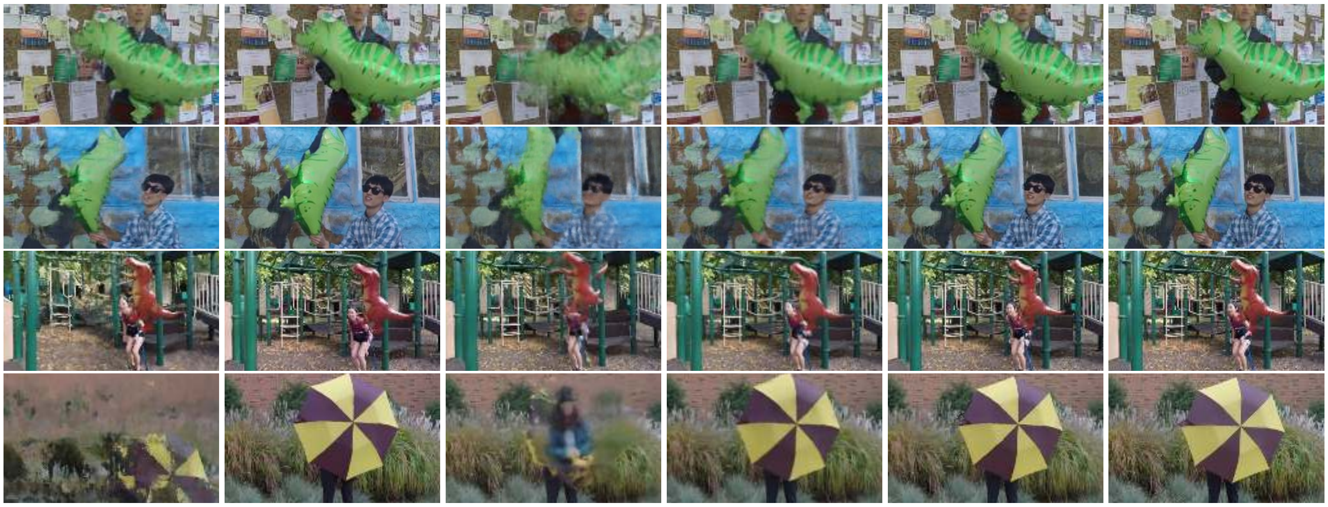
With both the static and dynamic NeRF model, we can easily compose them into a complete model using the predicted blending weight b and render full color frames at novel views and time:

$$\mathbf{C}^{full}(\mathbf{r}) = \sum_{k=1}^{K} T^{full} \Big(\alpha^d(\sigma^d \delta_k)(1 - b)\mathbf{c}^d + \alpha^s(\sigma^s \delta_k) b \mathbf{c}^s \Big)$$

We predict the blending weight b using the dynamic NeRF to enforce the time-dependency. Using the blending weight, we can also render a dynamic component only frame where

Table 1. **Novel view synthesis results.** We report the average PSNR and LPIPS results with comparisons to existing methods on Dynamic Scene dataset [58]. The best performance is in **bold** and the second best is underscored.

PSNR ↑ / LPIPS ↓	Jumping	Skating	Truck	Umbrella	Balloon1	Balloon2	Playground	Average
NeRF	20.58 / 0.305	23.05 / 0.316	22.61 / 0.225	21.08 / 0.441	19.07 / 0.214	24.08 / 0.098	20.86 / 0.164	21.62 / 0.252
NeRF + time	16.72 / 0.489	19.23 / 0.542	17.17 / 0.403	17.17 / 0.752	17.33 / 0.304	19.67 / 0.236	13.80 / 0.444	17.30 / 0.453
Yoon et al. [58]	20.16 / 0.148	21.75 / 0.135	23.93 / **0.109**	20.35 / 0.179	18.76 / 0.178	19.89 / 0.138	15.09 / 0.183	19.99 / 0.153
Tretschk et al. [51]	19.38 / 0.295	23.29 / 0.234	19.02 / 0.453	19.26 / 0.427	16.98 / 0.353	22.23 / 0.212	14.24 / 0.336	19.20 / 0.330
Li et al. [26]	24.12 / 0.156	**28.91** / 0.135	**25.94** / 0.171	22.58 / 0.302	21.40 / 0.225	24.09 / 0.228	20.91 / 0.220	23.99 / 0.205
Ours	**24.23** / **0.144**	28.90 / **0.124**	25.78 / 0.134	**23.15** / **0.146**	**21.47** / **0.125**	**25.97** / **0.059**	**23.65** / **0.093**	**24.74** / **0.118**



NeRF + time Yoon et al. [58] Tretschk et al. [51] Li et al. [26] Ours Ground truth

Figure 8. **Novel view synthesis.** Our model enables the free-viewpoint synthesis of a dynamic scene. Compared with Yoon et al. [58], our results appear slightly blurry (because we reconstruct the entire frame as opposed to warp and blend input images), but align with the ground truth image better and create smoother view-interpolation results. When compared to other NeRF-based methods, our results are sharper and closer to the ground truth. Please refer to the supplementary material for video results.

the static region is transparent (Figure 7).

Full rendering photometric loss. We train the two NeRF models jointly by applying a reconstruction loss on the composite results:

$$\mathcal{L}_{full} = \sum_{ij} \left\| \mathbf{C}^{full}(\mathbf{r}_{ij}) - \mathbf{C}^{gt}(\mathbf{r}_{ij}) \right\|_2^2 \tag{13}$$

4. Experimental Results

4.1. Experimental setup

Dataset. We evaluate our method on the Dynamic Scene Dataset [58], which contains 9 video sequences. The sequences are captured with 12 cameras using a static camera rig. All cameras simultaneously capture images at 12 different time steps $\{t_0, t_1, \dots, t_{11}\}$. The input twelve-frames monocular video $\{\boldsymbol{I}_0, \boldsymbol{I}_1, \dots, \boldsymbol{I}_{11}\}$ is obtained by sampling the image taken by the i-th camera at time t_i. Please note that a different camera is used for each frame of the video to simulate camera motion. The frame $\boldsymbol{I}_i$ contains a background that does not change in time, and a time-varying dynamic object. Like NeRF [32], we use COLMAP to estimate the camera poses and the near and far bounds of the scene. We assume all the cameras share the same intrinsic parameter. We exclude the DynamicFace sequence because COLMAP fails to estimate camera poses. We resize all the sequences to 480 × 270 resolution.

4.2. Evaluation

Quantitative evaluation. To quantitatively evaluate the synthesized novel views, we fix the view to the first camera and change time. We show the PSNR and LPIPS [60] between the synthesized views and the corresponding ground truth views in Table 1. We obtain the results of Li et al. [26] and Tretschk et al. [51] using the official implementation with default parameters. Note that the method from Tretschk et al. [51] needs per-sequence hyper-parameter tuning. The visual quality might be improved with careful hyper-parameter tuning. Our method compares favorably against the state-of-the-art algorithms.

Qualitative evaluation. We show the sample view synthesis results in Figure 8. With the learned neural implicit representation of the scene, our method can synthesize *novel views* that are never seen during training. Please refer to

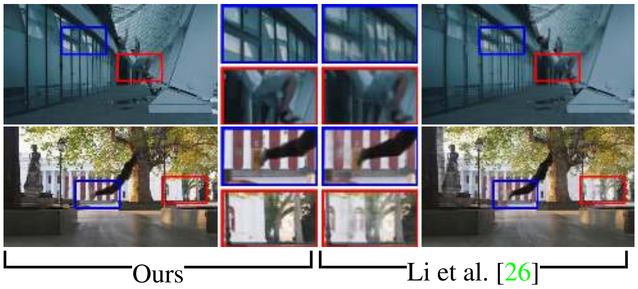


Figure 9. **Comparison with [26].** We show that our proposed regularizations are the keys to better visual results.

Table 2. **Ablation study on different losses.** We report PSNR, SSIM and LPIPS on the Playground sequence.

	PSNR ↑	SSIM ↑	LPIPS ↓
Ours w/o $\mathcal{L}_{full}$	12.90	0.1549	0.991
Ours w/o static NeRF	18.81	0.4969	0.293
Ours w/o blending	22.23	0.7610	0.153
Ours w/o $\mathcal{L}_{sparsity}$	22.32	0.7884	0.137
Ours w/o $\mathcal{L}_{reg}$	22.53	0.7972	0.124
Ours w/o $\mathcal{L}_{motion}$	22.61	0.8027	0.137
Ours w/o rigidity	22.73	0.8142	0.118
Ours w/o $\mathcal{L}_{depth}$	22.99	0.8170	0.117
Ours w/o mask	23.43	0.8205	0.102
Ours	**23.65**	**0.8452**	**0.093**

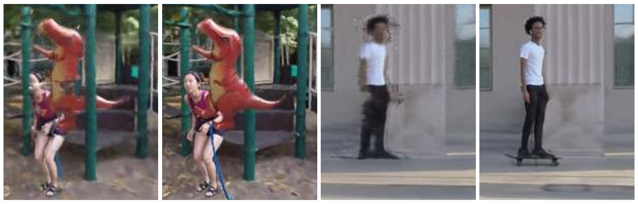
the supplementary video results for the novel view synthesis, and the extensive qualitative comparison to the methods listed in Table 1.

Figure 9 shows the comparison with Li et al. [26] on large motion sequences taken in the wild. Unlike [26] which predicts the blending weight using a static NeRF, we learn a time-varying blending weight. This weight helps better distinguish the static region and yields a clean background. Our rigidity regularization encourages the scene flow to be zero for the rigid region. As a result, the multi-view constraints enforce the background to be static. Without this regularization, the background becomes time-variant and leads to floating artifacts in [26].

4.3. Ablation Study

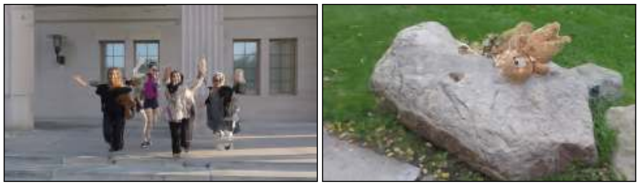
Table 2 analyzes the contribution of each loss quantitatively.

Depth order loss. For a complicated scene, we need additional supervision to learn the correct geometry. In Figure 10 we study the effect of the depth order loss. Since the training objective is to minimize the image reconstruction loss on the input views, the network may learn a solution that correctly renders the given input video. However, it may be a physically incorrect solution and produces artifacts at novel views. With the help of the depth order loss $\mathcal{L}_{depth}$, our dynamic NeRF model learns the correct relative depth and renders plausible content.



Without depth order loss | *With* depth order loss | *Without* motion regularization | *With* motion regularization

Figure 10. **Depth order loss and motion regularization.** Training with depth order loss ensures the correct relative depth of the dynamic object. Regularizing our scene flow prediction in dynamic NeRF can help handle videos with large object motion.



Non-rigid deformation | Incorrect flow

Figure 11. **Failure cases.** (*Left*) Our method does not handle non-rigid deformation very well. (*Right*) Our dynamic NeRF heavily relies on the optical flow estimation and produces artifacts with inaccurate flow estimates.

Motion regularization. Supervising scene flow prediction with the 2D optical flow is under-constrained. We show in Figure 10 that without a proper motion regularization, the synthesized results are blurry. The scene flow may point to the wrong location. By regularizing the scene flow to be *slow*, *temporally and spatially smooth*, and *consistent*, we obtain plausible results.

Rigidity regularization of the scene flow. The rigidity regularization helps with a more accurate scene flow prediction for the static region. The dynamic NeRF is thus trained with a more accurate multi-view constraint. We show in Figure 9 that the rigidity regularization is the key to a clean background.

4.4. Failure Cases

Dynamic view synthesis remains a challenging problem. We show and explain several failure cases in Figure 11.

5. Conclusions

We have presented a new algorithm for dynamic view synthesis from a single monocular video. Our core technical contribution lies in scene flow based regularization for enforcing temporal consistency and alleviates the ambiguity when modeling a dynamic scene with only one observation at any given time. We show that our proposed scene flow based 3D temporal consistency loss and the rigidity regularization of the scene flow prediction are the keys to better visual results. We validate our design choices and compare favorably against the state of the arts.

References

[1] Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. MatryODShka: Real-time 6DoF video view synthesis using multi-sphere images. In *ECCV*, 2020. 2

[2] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *CVPR*, 2020. 2

[3] Luca Ballan, Gabriel J Brostow, Jens Puwein, and Marc Pollefeys. Unstructured video-based rendering: Interactive exploration of casually captured videos. *ACM TOG (Proc. SIGGRAPH)*, 2010. 2

[4] Aayush Bansal, Minh Vo, Yaser Sheikh, Deva Ramanan, and Srinivasa Narasimhan. 4D visualization of dynamic events from unconstrained multi-view videos. In *CVPR*, 2020. 2

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

[6] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001. 2

[7] Canon. Free viewpoint video system. https://global.canon/en/technology/frontier18.html, 2008. 1

[8] Joel Carranza, Christian Theobalt, Marcus A Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. *ACM TOG (Proc. SIGGRAPH)*, 22(3):569–577, 2003. 1, 2

[9] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM TOG (Proc. SIGGRAPH)*, 32(3):1–12, 2013. 2

[10] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 2

[11] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM TOG (Proc. SIGGRAPH)*, 34(4):1–13, 2015. 1, 2

[12] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4D: Real-time performance capture of challenging scenes. *ACM TOG (Proc. SIGGRAPH)*, 35(4):1–13, 2016. 2

[13] Robert A Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4):65–74, 1988. 4

[14] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. DeepView: View synthesis with learned gradient descent. In *CVPR*, 2019. 2

[15] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR*, 2016. 2

[16] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3D shape. In *CVPR*, 2020. 2

[17] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *ICCV*, 2019. 2

[18] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the conference on Computer graphics and interactive techniques*, pages 43–54, 1996. 2

[19] Marc Habermann, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Livecap: Real-time human performance capture from monocular video. *ACM TOG (Proc. SIGGRAPH)*, 38(2):1–17, 2019. 2

[20] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM TOG (Proc. SIGGRAPH)*, 37(6):1–15, 2018. 2

[21] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM TOG (Proc. SIGGRAPH)*, 35(6):1–10, 2016. 2

[22] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020. 2

[23] Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. Image-based rendering in the gradient domain. *ACM TOG (Proc. SIGGRAPH Asia)*, 32(6), 2013. 2

[24] Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, et al. One shot 3D photography. *ACM TOG (Proc. SIGGRAPH)*, 39(4):76–1, 2020. 2

[25] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the annual conference on Computer graphics and interactive techniques*, 1996. 2

[26] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 3, 7, 8

[27] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM TOG (Proc. SIGGRAPH)*, 2020. 2

[28] Marcus Magnor, Marc Pollefeys, German Cheung, Wojciech Matusik, and Christian Theobalt. Video-based rendering. In *ACM SIGGRAPH 2005 Courses*, 2005. 2

[29] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *arXiv preprint arXiv:2008.02268*, 2020. 3

[30] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019. 2

[31] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *ICCV*, 2019. 2

[32] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 3, 4, 7

[33] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4D reconstruction by learning particle dynamics. In *ICCV*, 2019. 3

[34] M. Niemeyer, Lars M. Mescheder, Michael Oechsle, and A. Geiger. Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision. *CVPR*, 2020. 2

[35] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3D ken burns effect from a single image. *ACM TOG (Proc. SIGGRAPH Asia)*, 38(6):1–15, 2019. 2

[36] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *ICCV*, 2019. 2

[37] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. Holoportation: Virtual 3D teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, 2016. 1, 2

[38] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2

[39] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo-Martin Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020. 3

[40] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 2

[41] Eric Penner and Li Zhang. Soft 3D reconstruction for view synthesis. *ACM TOG (Proc. SIGGRAPH)*, 36(6):1–11, 2017. 2

[42] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 3

[43] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020. 6

[44] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, 2020. 2

[45] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, 2019. 2

[46] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3D photography using context-aware layered depth inpainting. In *CVPR*, 2020. 2

[47] Harry Shum and Sing Bing Kang. Review of image-based rendering techniques. In *Visual Communications and Image Processing 2000*, volume 4067, pages 2–13, 2000. 2

[48] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *NeurIPS*, 2019. 2

[49] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 5

[50] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, 2020. 2

[51] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a deforming scene from monocular video. *arXiv preprint arXiv:2012.12247*, 2020. 3, 7

[52] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020. 2

[53] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 2

[54] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, 2021. 3

[55] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3D reconstruction. In *NeurIPS*, 2019. 2

[56] Lior Yariv, Matan Atzmon, and Yaron Lipman. Universal differentiable renderer for implicit neural representations. *arXiv preprint arXiv:2003.09852*, 2020. 2

[57] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *NeurIPS*, 2020. 2

[58] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *CVPR*, 2020. 2, 7

[59] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[60] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 7

[61] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM TOG (Proc. SIGGRAPH)*, 23(3):600–608, 2004. 1, 2