Space-time Neural Irradiance Fields for Free-Viewpoint Video

Wenqi Xian* Cornell Tech
Jia-Bin Huang Virginia Tech
Johannes Kopf Facebook
Changil Kim Facebook

https://video-nerf.github.io

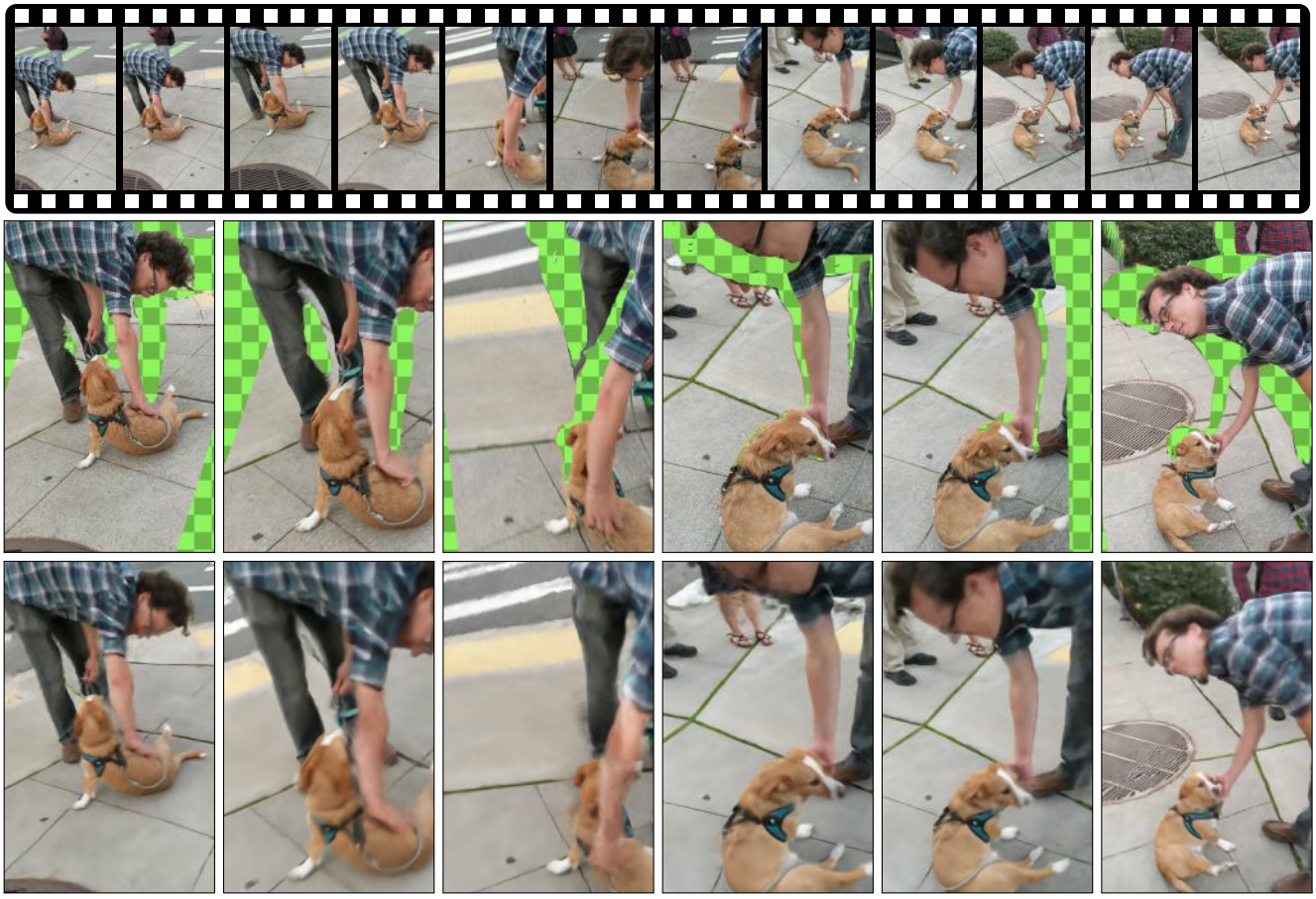


Figure 1. Our method takes a *single* casually captured video as input and learns a space-time neural irradiance field. (*Top*) Sample frames from the input video. (*Middle*) Novel view images rendered from textured meshes constructed from depth maps. (*Bottom*) Our results rendered from the proposed space-time neural irradiance field.

Abstract

We present a method that learns a spatiotemporal neural irradiance field for dynamic scenes from a single video. Our learned representation enables free-viewpoint rendering of the input video. Our method builds upon recent advances in implicit representations. Learning a spatiotemporal irradiance field from a single video poses significant challenges because the video contains only one observation of the scene at any point in time. The 3D geometry of a scene can be legitimately represented in numerous ways since varying geometry (motion) can be explained with varying appearance and vice versa. We address this ambiguity by constraining the time-varying geometry of our dynamic scene representation using the scene depth estimated from video depth estimation methods, aggregating contents from individual frames into a single global representation. We provide an extensive quantitative evaluation and demonstrate compelling free-viewpoint rendering results.

* This work was done while Wenqi was an intern at Facebook.

1. Introduction

This paper addresses the problem of rendering a video from novel viewpoints. Specifically, we learn a *globally consistent, dynamic* scene representation that can later be rendered from a novel viewpoint. We learn such a representation from a casually captured *single* video from everyday devices such as smartphones, without the assistance of multi-camera rigs or other dedicated hardware (which are typically not accessible to casual users).

Free-viewpoint video rendering typically requires a complicated hardware setup consisting of multiple cameras to capture the scene of interest from different viewpoints [58, 9, 12, 5]. The multi-camera setup in existing methods is required because conventional 3D reconstruction algorithms (multi-view stereopsis) assume a fully *static* scene and thus can perform reconstruction using the multiple captured viewpoints of a dynamic scene at any time. Most methods represent the geometric reconstructions as some form of *per-frame* representation (e.g., depth maps [58] or meshes[5]). Rendering from a novel viewpoint can then be achieved, e.g., by warping available views using their depth maps to the new viewpoint.

Following the success of single-image depth estimation, recent monocular video depth estimation methods allow for the acquisition of consistent per-frame depth estimates from only a single video [27, 54]. While still at an early stage, this line of work opens up new possibilities, where monocular scene depth estimates can be directly used for view synthesis. However, naïve approaches such as per-frame depth-based warping would lead to unnatural stretches and reveal holes in disoccluded regions (even with perfect depth estimates). One can alleviate this by post-processing the incomplete rendering [54]. However, such per-frame processing methods often lead to temporal flickers. The core problem lies in the use of a *frame-wise representation* (e.g., depth maps associated with the input images), and therefore suffer from issues ranging from temporal inconsistency to high redundancy and thus excessive storage requirements and data transfer bandwidth.

In this work, we build on recent monocular video depth estimation methods and aggregate the entire spatiotemporal aspects of a dynamic scene in a single global representation. While fusing multiple depth maps into a single, global representation has a long tradition, most work on volumetric depth integration has focused on static scenes [10, 34] or geometry alone without textures [33]. These methods typically use *discrete* representations such as voxel grids, meshes, or point clouds. Consequently, these methods often suffer from premature hard decisions on geometry estimation and limited resolution due to high storage requirements.

In this paper, we, instead, turn to the recent advances in neural implicit representations, which allow for *continuous* representations of a scene without resolution loss. Recent work has shown that these representations achieve high-quality view interpolation of complex *static* scenes while retaining their advantages over discrete representations [32, 55, 26]. The current approaches to learn them, however, require either multiple posed images of a *fully static* scene [32, 55, 26] or ground truth 3D representations [44, 45]. While videos often contain appearances of a scene seen from multiple viewpoints, they only contain *exactly one* viewpoint at *any given time*. Combined with the dynamic nature of video, this renders it nontrivial to extend current approaches to learn spatiotemporal representations from a single video.

Specifically, we learn neural irradiance fields as a function of both space and time for each video. We do not model view dependency, hence we use the term irradiance. Using supervision from only color frames of the input video as in [32] is futile, since the variations between frames can be explained with *either* a change of appearance *or* geometry, or a *combination* of both. We resolve this ambiguity using the per-frame scene depth estimated from monocular video depth estimation. Our depth supervision constrains the scene's geometry at any moment and disambiguates it from appearance variations. While this enables us to encode physically correct appearance and geometry in a global representation, it fails to fill the holes that could be seen at other time steps in the video. We address this by encouraging the color and volume density to propagate across time whenever spatial locations are not supervised otherwise. The resulting representation allows us to render the video from novel viewpoints and time: our implicit model can be queried at any spatiotemporal location and rendered using standard volume rendering[†].

Our technical contributions include the following:

- We aggregate frame-wise 2.5D representations into a globally consistent spatiotemporal representation from a single monocular video.
- We address the inherent motion–appearance ambiguity using video depth supervision and constrain the disoccluded contents by propagating the color and volume density across time.
- We demonstrate a compelling free-viewpoint video rendering experience on various casual videos shot from smartphones, preserving motion and texture details while conveying a vivid sense of 3D.

2. Related Work

View synthesis for images. Creating novel views from multiple images is a long-standing problem in computer vision and computer graphics. Existing image-based

[†]Note that our method can render arbitrary viewpoints at all the *observed* time steps. We do not extrapolate or interpolate the time steps.

rendering techniques first extract approximated geometric proxy and create novel target views by warping and blending the corresponding contents from multiple source frames [23, 42, 17, 43]. Recently, neural implicit representation has shown high-quality view synthesis results by implicitly modeling the volume density and color of the scene using the weights of a multilayer perceptron [32, 55, 26]. A another line of work further pushes the requirement of the number of input images to a narrow-baseline stereo pair [56, 48, 8] or a single image [57, 50, 38, 51, 24, 47]. Our work uses NeRF [32] as our base scene representation for view synthesis. Unlike NeRF that only models *static* scenes, our focus is on creating new views from arbitrary viewpoint and time for *dynamic* scenes.

View synthesis for videos. Compared to images, view synthesis for video poses significant challenges due to the need to handle *time-varying* scene geometry and appearances. Consequently, most of the existing methods typical require laborious multi-camera setup [58, 9, 40, 11, 5], special hardware [1], or synchronous video captures from multiple viewpoints [3, 4]. Several methods can reduce the required number of input views by focusing on specific domains such as performance capture [7, 12, 16]. In contrast, our work aims to enable view synthesis of a *complex dynamic scene* at any given viewpoints and time from a *single* video. Very recently, Yoon et al. [54] also explore the same problem setup. Our work differs in three major aspects. First, unlike [54], our method does not require manual segmentation of dynamic objects as inputs. Second, our method does not assume a simple two-layer (foreground-background) model of the scene and can handle more generic scenes. Third, in contrast to [54] that processes each novel view *independently* (by warping blending multiple images), we can render an interpolation video with temporally smooth transitions across viewpoints.

Neural implicit representation. Implicit representation has emerged as a powerful tool to overcome conventional limitations of *discrete* 3D representations such as voxel grids or meshes. The core idea is to use a multilayer perceptron (MLPs) to implicitly model the occupancy [30, 31], signed distance functions [41, 2], object appearance [39], volumetric density [32] in the 3D space. Differentiable rendering techniques enable training these models without accessing ground truth data for direct 3D supervision [28, 36, 32, 26]. However, extending the above methods to handle scene dynamics is not trivial due to the motion–appearance ambiguity. Occupancy flow [35] achieves 4D reconstruction (3D shape + 1D time) by learning continuous motion fields. Our work builds upon the recent advances in neural implicit representation but focuses on representing a *dynamic video*. Compared to 4D reconstruction in [35], our method differs in the following two aspects. First, our method does not require direct 3D ground truth training data. Second, in addition to model the time-varying 3D geometry, we also model the appearance of complex scenes.

Video depth estimation. Estimating dense depth from a dynamic video is a challenging task. Existing multi-view stereo (MVS) methods (either geometric-based [46, 13] or learning-based [19, 53, 15]) assume static scene and thus not suitable for dynamic videos as it often produces erroneous depth for moving objects or untextured regions. Several video depth estimation methods predict depth using the cost volume computed from nearby frames [49, 25]. Similar to MVS algorithms, these video-to-depth methods have difficulties in handling dynamic scenes well. Very recently, hybrid methods that combine MVS and single-image depth estimation models have been proposed [27, 54]. Our work leverages the estimated depth from [27] to help resolve the ambiguity when learning the spatiotemporal neural radiance fields using a single video. Our approach renders photorealistic views with correctly filled dis-occluded contents compared to view synthesis with per-frame depth-based warping.

Video completion. Video completion algorithms aim to fill in plausible contents for missing regions in a video in a temporally coherent manner [20]. State-of-the-art methods achieve temporally consistent completion by propagating known contents to missing regions along flow trajectories [18, 52, 14]. One may first render a free-viewpoint video according to each frame's estimated depth, followed by filling the missing pixels (disoccluded regions due to view changes) using video completion algorithms. Our method also produces *completed* novel views (i.e., with no missing pixels from disocclusion). However, unlike video completion algorithms that inpaint the dis-occluded pixels in the *screen space*, our approach fills in the dis-occluded content implicitly in the *3D space*. Our experiments validate that our approach produces significantly fewer artifacts than the baseline method using video completion.

Depth map fusion. A line of research work focuses on fusing a sequence of RGB-Depth images in a video into a *global* with voxel-, point-based, signed distance field-based representation [59]. Examples include 3D reconstruction for static scenes [37, 22] or dynamic objects from a single [33, 21] or multiple RGB-D cameras [12]. Our work differs from prior 3D reconstruction methods in two aspects. First, we do not assume a fixed, canonical 3D model as in existing dynamic 3D reconstruction methods and, therefore, can naturally handle an entire dynamic scene (as opposed to only individual objects). Second, our approach with neural implicit representations jointly models time-varying geometry *and* appearance.

3. Background

Our representation builds on the neural radiance field, or NeRF [32], which we recap in this section. NeRF represents the radiance $\mathbf{c} = (r, g, b)$ and differential volume density σ at a 3D location $\mathbf{x} = (x, y, z)$ of a scene observed from a viewing direction $\mathbf{d} = (\theta, \phi)$ as a continuous multi-variate function using a multi-layer perceptron (MLP): $F_{\text{NeRF}} : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$.

The color of a pixel can be rendered by integrating the radiance modulated by the volume density along the camera ray $\mathbf{r}(s) = \mathbf{o} + s\mathbf{d}$, shot from the camera center through the center of the pixel:

$$C(\mathbf{r}) = \int_{s_\text{n}}^{s_\text{f}} T(s)\sigma\Big(\mathbf{r}(s)\Big)\mathbf{c}\Big(\mathbf{r}(s), \mathbf{d}\Big)\mathrm{d}s, \tag{1}$$

where

$$T(s) = \exp\left(-\int_{s_\text{n}}^{s} \sigma(\mathbf{r}(p))\mathrm{d}p\right) \tag{2}$$

is the accumulated transmittance along the ray $\mathbf{r}$ up to s.

One can train the MLP using multiple posed images, capturing a *static* scene from different viewpoints. Specifically, we minimize the photometric loss that compares the rendering through a ray $\mathbf{r}$ with the corresponding ground truth color from an input image:

$$\mathcal{L}_{\text{NeRF}} = \sum_{\mathbf{r}\in\mathcal{R}} \left\|\hat{C}(\mathbf{r}) - C(\mathbf{r})\right\|_2^2, \tag{3}$$

where $\mathcal{R}$ denotes a set of rays, and $C(\mathbf{r})$ and $\hat{C}(\mathbf{r})$ the ground truth and the estimated color, respectively.

In the implementation, the continuous volume rendering of (1) is approximated by numerical quadrature, i.e., computing the color using a finite number of sampled 3D points along a ray and calculate the summation of the radiances, weighted by the discrete transmittance. As this weighted summation process is differentiable, the gradient can propagate backward for optimizing the MLP. We perform the sampling in two steps. First, a ray is sampled uniformly in s, and then, it is sampled with respect to the approximate transmittance so that more samples are used around surfaces in the scene. The two groups of samples are evaluated in separate *coarse* and *fine* networks, and both are used to measure the loss (3).

4. Space-time Neural Irradiance Fields

We represent a 4D space-time irradiance field as a function that maps a spatiotemporal location $(\mathbf{x}, t)$ to the emitted color and volume density, $F : (\mathbf{x}, t) \rightarrow (\mathbf{c}, \sigma)$. Our input video is represented as a stream of RGB-D images, $I_t : \mathbf{u} \rightarrow (\mathbf{c}, d)$ at discrete time steps $t \in \mathcal{T} = \{1, 2, ..., N_\text{f}\}$, where $\mathbf{u} = (u, v)$ is 2D pixel coordinates, and their associated camera calibrations $\mathcal{P}_t$.

(a) w/o depth loss (b) w/ depth loss

Figure 2. **Depth loss.** (a) A model trained *without* our depth loss can reconstruct the image from the original viewpoint well (*first*). However, with a slight viewpoint change, the synthesized image (*second*) suffers from strong visual artifacts due to incorrect geometry (*third*). (b) A trained *with* the proposed depth loss render the novel viewpoint without clearly visible artifacts.

A ray $\mathbf{r}$ at time t can be determined by a pixel location $\mathbf{u}$ and the camera calibration $\mathcal{P}_t$: it marches from the camera center through the center of pixel denoted by $\mathbf{u}$. Additionally, we parameterize a ray such that the parameter s denotes the scene depth. This is achieved by setting the directional vector $\mathbf{d}$ such that its projection onto the principal ray has a unit norm in the camera space.

Color reconstruction loss. To learn the implicit function F from the input video I, first and foremost, we constrain our representation F such that it reproduces the original video I when rendered from the original viewpoint for each frame. Specifically, we penalize the difference between the volume-rendered image at each time t and the corresponding input image I_t. This amounts to the reconstruction loss of the original NeRF [32]:

$$\mathcal{L}_{\text{color}} = \sum_{(\mathbf{r},t)\in\mathcal{R}} \left\|\hat{C}(\mathbf{r}, t) - C(\mathbf{r}, t)\right\|_2^2, \tag{4}$$

where $\mathcal{R}$ is a batch of rays, each associated with a time t.

Unlike NeRF, for dynamic scenes, we have to reconstruct the *time-varying* scene geometry at *every* time t. However, a single video contains only one observation of the scene at any point in time, rendering the estimation of scene geometry severely under-constrained. That is, the 3D geometry of a scene can be legitimately represented in numerous (infinitely possible) ways since varying geometry can be explained with the varying appearance and vice versa. For example, any input video can be reconstructed with a "a flat TV" solution (with a planar geometry with each frame texture-mapped).

Thus, the color reconstruction loss provides the ground for accurate reconstruction *only* when the learned representation is rendered from the same camera trajectory of the input, lacking any machinery that drives learning correct geometry. Incorrect geometry would lead to artifacts as soon as we start deviating from the original video's camera trajectory, as shown in Figure 2a.

Depth reconstruction loss. We resolve this motion–appearance ambiguity by constraining the time-varying geometry of our dynamic scene representation using the per-frame scene depth of the input video (estimated from video depth estimation methods). We estimate the scene depth from the learned volume density of the scene, and measure its difference from the input depth d_t. It is a non-trivial question on how to define the scene depth of a ray. One possibility is to measure the distance where the accumulated transmittance T becomes less than a certain threshold. Such an approach, however, involves heuristics and hard decisions. Instead, we accumulate depth values along the ray modulated both with the transmittance and volume density, similarly to the depth composition in layered scene representations [50].

Our depth reconstruction loss is of the form:

$$\mathcal{L}_{\text{depth}} = \sum_{(\mathbf{r},t)\in\mathcal{R}} \left\| \frac{1}{\hat{D}(\mathbf{r},t)} - \frac{1}{D(\mathbf{r},t)} \right\|_2^2, \tag{5}$$

where

$$\hat{D}(\mathbf{r},t) = \int_{s_\text{n}}^{s_\text{f}} T(s,t)\,\sigma(\mathbf{r}(s),t)\,s\,\mathrm{d}s, \tag{6}$$

is the integrated sample depth values along the ray and $D(\mathbf{r})$.

Empty-space loss. Constraining the depth predicted by our model using the estimated scene depth is not sufficient to capture accurate scene geometry. This is because the predicted depth is, in essence, a *weighted sum* of depth values along the ray. Consequently, we sometimes see haze-like visual artifacts when rendering at novel views. We propose to encourage *empty space* between the camera and the first visible scene surface to address this issue. A similar idea has been used in volumetric depth integration [10]. To this end, we penalize non-zero volume densities measures along each ray up to the point no closer than a small margin $\varepsilon = 0.05 \cdot (s_\text{f} - s_\text{n})$ to the scene depth for each ray:

$$\mathcal{L}_{\text{empty}} = \sum_{(\mathbf{r},t)\in\mathcal{R}} \int_{s_\text{n}}^{d_t(\mathbf{u})-\varepsilon} \sigma(\mathbf{r}(s),t)\,\mathrm{d}s, \tag{7}$$

where $\mathbf{u}$ denotes the pixel coordinates where $\mathbf{r}$ intersects with the image plane at t, $d_t(\mathbf{u})$ denote the scene depth for the pixel $\mathbf{u}$ at time t.

The empty-space loss combined with the depth reconstruction loss provides geometric constraints for our representation up to and around *visible* scene surfaces at each frame. The learned representations can thus produce geometrically correct novel view synthesis, as shown in Figure 2b.

Static scene loss. A large portion of spaces that is *hidden* from the input frame's viewpoint at any given time is still *not constrained*, i.e., the MLP has not seen the 3D positions and time as input queries during training. As a result, when these unconstrained spaces are disoccluded due to viewpoint changes, they are prone to artifacts (see Figure 4 for an example). However, there is a high chance that a portion of disoccluded spaces is observed from a *different viewpoint* at *another time*. Our idea is to constrain the MLP by propagating these partially observed contents across time. However, instead of explicitly correlating surfaces over time, e.g, using the scene flow, we choose to constrain the spaces surrounding the surface regions. This allows us to avoid misalignment of scene surfaces due to unreliable geometry estimates or other image aberrations commonly seen in captured videos such as exposure or color variations.

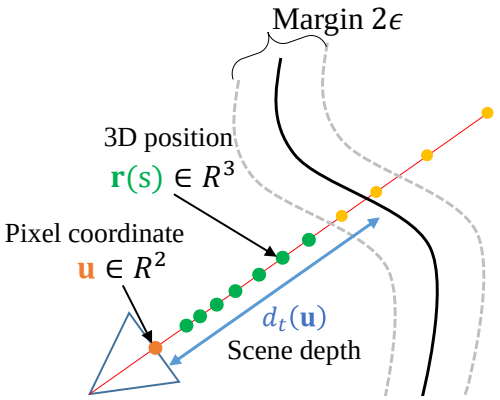


Figure 3. **Scene sampling.** We measure the *Empty-space loss* on 3D positions until hitting the estimated scene depth (green). We use all the samples along the ray (green and yellow) to compute the *Depth* and *Color reconstruction loss*. For *Static loss*, we sample the scene from the union of the space spanned by all camera rays (within the range of $s \in [z_\text{n}, z_\text{f}]$) of all input frames. We exclude any samples that are close to any surface than ε.

We make a simple assumption on *unobserved* spaces: every part of the world should stay static unless observed not as such. Enforcing this assumption prevents the part of spaces that are not observed from going entirely unconstrained. Our static scene constraint encourages the shared color and volume density at the same spatial location $\mathbf{x}$ between two distinct times t and t':

$$\mathcal{L}_{\text{static}} = \sum_{(\mathbf{x},t)\in\mathcal{X}} \left\| F(\mathbf{x},t) - F(\mathbf{x},t') \right\|_2^2, \tag{8}$$

where both $(\mathbf{x},t)$ and $(\mathbf{x},t')$ are *not* close to any visible surfaces, and $\mathcal{X}$ denotes a set of sampling locations where the loss is measured.

Scene sampling. While we have locations for the color, depth, and free-space supervisions explicitly dictated by quadrature used by volume rendering [32], we are free to

choose *where* we apply the static constraints. A straightforward approach would be to use the same sampling locations that are used for other losses. We can then randomly draw another time t' that is distinct from the current time t and enforce the MLP to produce similar appearances and volume densities at these two spatiotemporal locations.

However, this still leaves a large part of the scene unconstrained when the camera motion is large. Uniformly sampling in the scene bounding volume would also not be ideal since sampling would be highly inefficient because of perspective projection (except for special cases like a camera circling some bounded volume).

As a simple solution to meet both the sampling efficiency and the sample coverage, we propose to take the *union* of all sampling locations along *all* rays of *all* frames to form our sample pool $\mathcal{X}$. We exclude all points that are closer to any observed surfaces than a threshold ε (see Figure 3). We randomly draw a fixed number of sampling locations from this pool at each training iteration and add small random jitters to each sampling location. At time t' the static scene loss is measured against is also randomly chosen for each sample location $\mathbf{x}$, while ensuring the resulting location $(\mathbf{x}, t')$ is not close to any scene surfaces.

Total loss. Our total loss for training the space-time irradiance fields is a linear combination of all losses presented above:

$$\mathcal{L} = \mathcal{L}_{\text{color}} + \alpha\mathcal{L}_{\text{depth}} + \beta\mathcal{L}_{\text{empty}} + \gamma\mathcal{L}_{\text{static}}. \qquad (9)$$

We validate the effectiveness of these losses in Section 5.

Implementation details. We use the hierarchical volume sampling as in the original NeRF and simultaneously train both the coarse and fine networks. We apply all losses to supervise the predictions from both networks.

We calculate all the losses except the static scene loss on a batch of $N_{\text{r}} = 1024$ rays that are randomly drawn from an input frame I_t without replacement. We randomly choose $N_{\text{s}} = 1024$ from $\mathcal{X}$ at each step (also without replacement) for the static scene loss. We normalize the time t such that $\mathcal{T} = [-1, 1]$ and apply the positional encoding with 4 frequency bands. Following NeRF [32], we apply positional encoding to spatial positions $\mathbf{x}$. While we do not use the normalized device coordinates, we sample each ray uniformly in *inverse* depth. In all our experiments, we fix the weights for losses as $\alpha = 1$, $\beta = 100$, and $\gamma = 10$. We set the depth range z_{n} and z_{f} as the global minimum and maximum of all frames' depth values.

We used the same MLP architecture as in [32]. Our models are trained with various combinations of the losses presented in Section 4 but otherwise with the same hyperparameters. We used the Adam optimizer with momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a learning rate of 0.0005. We train the MLP for 800k iterations. It takes about 50 hours to train a network with about 100 video frames at the 960×540 resolution with two NVIDIA V100 GPUs.

(a) Input frame (b) w/o static loss (c) w/ static loss

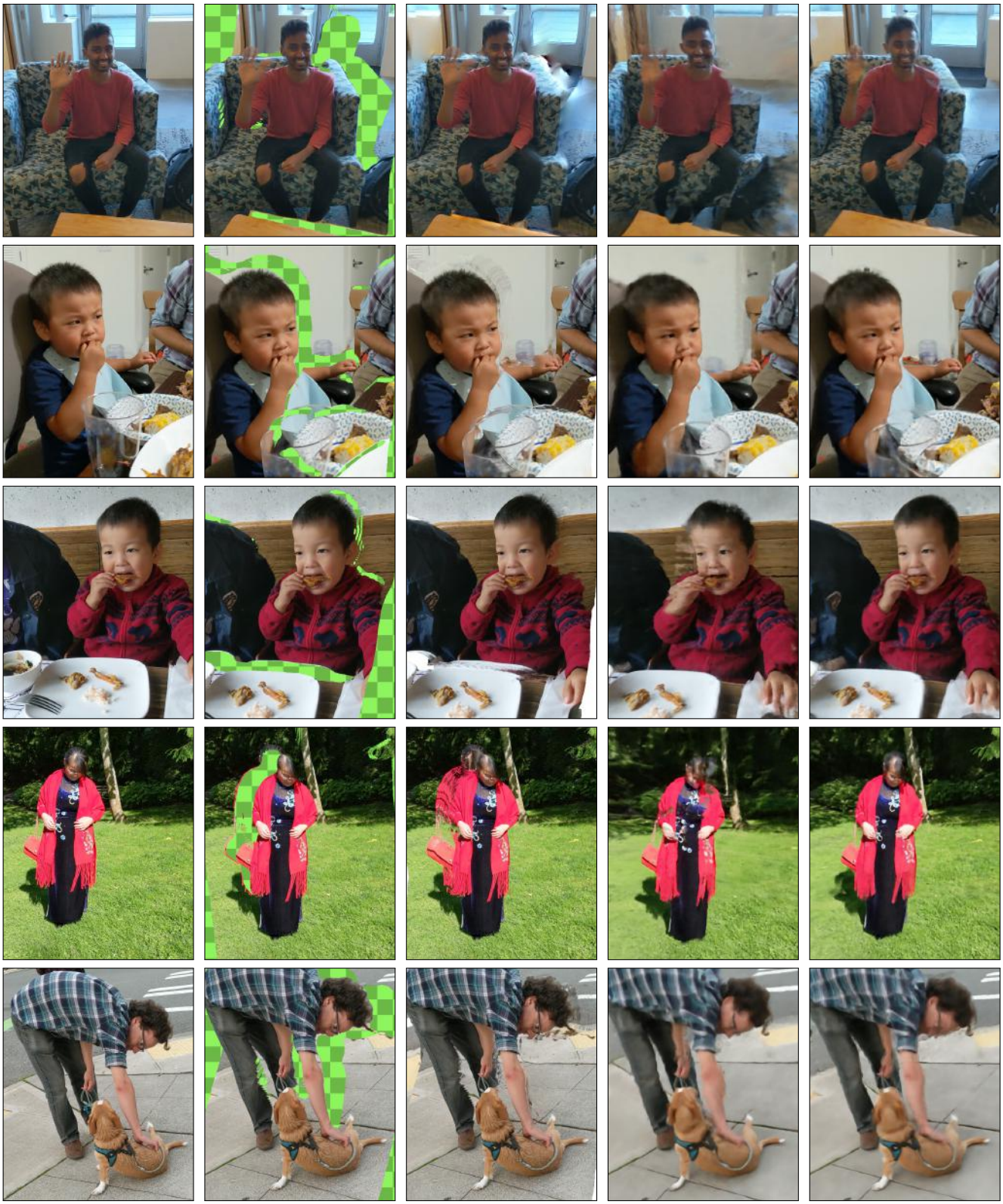
Figure 4. **Static scene loss.** Our static scene loss addresses the regions that are *not* constrained by the color and depth reconstruction loss, which handles only *visible* surfaces in the input frames (a). We render two frames at $t_1 = 9$ and $t_2 = 90$ from the viewpoint of $t = 40$. The camera has panned in the top row and zoomed out in the bottom row. When these frames are rendered from this novel viewpoint, previously hidden regions become *disoccluded*. They are, without the static scene loss, completely unconstrained and prone to ghosting or haze artifacts, as shown in (b). Our static loss alleviate these artifacts.

5. Experimental Results

We first compare our method with baseline approaches using the videos of dynamic scenes. Note that we always render new views at one of the observed times. That is, we do not evaluate our method's capability of temporal interpolation/extrapolation since our method is not designed to address it. We then provide extensive quantitative ablation studies with the variants of our model where each loss is added one at a time. We urge the readers to watch our supplementary videos in *our project webpage* (https://video-nerf.github.io), where we provide free-viewpoint rendering of our learned representations.

Datasets. We use the videos of dynamic scenes from the recent consistent video estimation method of Xuan et al. [27] along with the camera calibration and the per-frame depth maps provided together. [‡] Their dataset (denoted by

[‡]We also intended to use the dataset and video depth of Yoon et al. [54], but were unable to obtain the depth maps used in their results.



(a) Input (b) Mesh (c) Inpainted (d) NeRF-T (e) Ours

Figure 5. **Comparisons on novel view synthesis.** We compare our results with three baselines. (a) shows an input image and (b–d) show view synthesis results. (b) shows rendered mesh representation; (c) inpainted version of (b); (d) NeRF with an additional temporal parameter; (e) our method. The green areas represent disocclusion. Notice how the baselines are unable to recover the realistic appearance that respects an accurate geometry of the background scene. Results are best appreciated in the supplementary videos.

Table 1. **Ablation study.** We trained our model on the synthetic "Sintel" dataset with varying combinations of our losses and measure the view synthesis quality using three metrics: (1) "PSNR (All)" the PSNR on all pixels, (2) "PSNR (Occ.)" the PSNR on disoccluded pixels only, and (3) "SSIM" the structural similarity. With all metrics, the higher the better. Bold faces mean the best score, and the underlined mean the second best.

Model	Time	$\mathcal{L}_{\text{depth}}$ (5)	$\mathcal{L}_{\text{empty}}$ (7)	$\mathcal{L}_{\text{static}}$ (8)	$\mathcal{L}_{\text{flow}}$	View dir.	"Bandage 1"			"Bandage 2"			"Sleeping 1"			"Sleeping 2"			"Alley 1"			"Bamboo 1"		
							PSNR (All)	PSNR (Occ.)	SSIM (All)	PSNR (All)	PSNR (Occ.)	SSIM (All)	PSNR (All)	PSNR (Occ.)	SSIM (All)	PSNR (All)	PSNR (Occ.)	SSIM (All)	PSNR (All)	PSNR (Occ.)	SSIM (All)	PSNR (All)	PSNR (Occ.)	SSIM (All)
NeRF	-	-	-	-	-	-	11.65	11.32	0.638	13.41	12.19	0.499	14.67	16.03	0.641	20.49	20.32	0.818	10.93	10.93	0.641	17.28	17.52	0.792
NeRF-T	✓	-	-	-	-	-	11.81	11.27	0.658	13.84	12.26	0.462	15.45	15.81	0.663	13.83	22.08	0.480	11.50	12.50	0.665	15.59	15.32	0.732
1	✓	✓	-	-	-	-	14.82	12.87	0.796	18.61	18.81	0.807	21.85	21.43	0.830	31.01	32.93	0.965	24.67	25.28	0.930	**28.35**	**26.34**	**0.966**
2	✓	✓	✓	-	-	-	14.62	13.06	0.791	**22.26**	**23.64**	**0.877**	21.70	21.19	0.841	35.06	32.61	0.965	**26.75**	26.40	**0.941**	25.69	23.83	0.965
3	✓	✓	-	✓	-	-	15.48	13.15	0.812	20.41	22.11	0.860	21.85	21.59	0.841	**35.79**	**34.79**	**0.982**	25.34	21.42	0.937	27.20	23.59	0.957
4 ("Ours")	✓	✓	✓	✓	-	-	16.34	15.54	0.844	20.25	21.99	0.855	**22.53**	**21.65**	**0.844**	35.01	33.91	0.981	26.56	**26.75**	0.931	27.87	25.28	0.964
5	✓	✓	-	✓	✓	-	17.24	**16.84**	0.826	18.67	16.75	0.820	21.96	21.19	0.837	28.43	27.21	0.957	25.37	25.24	0.940	25.45	22.97	0.924
6	✓	✓	✓	-	✓	-	17.24	14.50	0.832	17.07	16.59	0.772	22.37	21.57	0.841	28.60	26.20	0.957	21.65	22.60	0.920	21.30	18.62	0.895
7	✓	✓	✓	✓	✓	✓	**18.14**	14.02	**0.855**	18.77	18.32	0.810	18.36	18.95	0.808	28.26	26.71	0.944	20.05	24.05	0.899	20.70	18.08	0.885

"CVD") consists of short videos of moving subjects captured by a smartphone.

For quantitative evaluation, we use the synthetic stereo videos from the MPI Sintel dataset [6]. We select 6 videos that show a variety of characteristics in terms of scene motion, camera motion, and the size of moving subjects. We use the left video to train our models and render them from the right video's viewpoints for ground truth comparisons.

Baselines. We compare our method against several baseline methods: ("Mesh") textured mesh representations directly reconstructed from the input depth maps as demonstrated by Xuan et al. [27]; ("Inpainted") its inpainted version, where disoccluded empty pixels in the 2D rendered images are inpainted using a recent video inpainting method [14]; and ("NeRF-T") a version of NeRF with an extra time parameter, which is our model trained with only the color reconstruction loss (4). Note that we do *not* use the viewing directions for the NeRF baseline.

For "Mesh", we first tessellate an input image to a triangle mesh using the corresponding depth map. The mesh is then rendered from a novel viewpoint. We tear the mesh across large depth discontinuity to avoid excessive stretches of textures by removing triangles whose vertices have large depth differences.

Qualitative comparisons. Figure 5 presents the comparisons of our model against the baselines using the "CVD" dataset, where we show view synthesis results from novel viewpoints. We used our full method with all losses presented in Section 4 to create these results. Please refer to *our project webpage* for the full video results.

Ablation studies. We trained our model with the different combinations of losses and measured the view synthesis quality in three metrics, PSNR on the entire image, PSNR on the disoccluded regions only, and SSIM on the entire image. Table 1 summarizes the results. In addition to the four losses presented in Section 4, we test an addition loss, $\mathcal{L}_{\text{flow}}$, which measures the consistency of the color and the volume density between two corresponding spatiotemporal locations via *scene flow*. We obtain the scene flow from the 2D optical flow raised to 3D using the per-frame scene depth. Our motivation is to further encourage temporal smoothness and disocclusion handling. Since we are effectively gather view-dependent appearances with the scene flow loss, we tested both models with and without the viewing directions used.

For quantitative evaluation, we train our models using the Sintel dataset. The left videos are used to train the models rendered from the right videos' viewpoints and then compared to the ground truth right videos. All metrics aggregate the scores over all frames. Our model always works better when trained with the depth loss. While varying depending on scene types, the static loss and empty space loss help improve the results as well. However, we find that the scene flow loss does not help improve quality. We suspect that this is because casual videos often include strong image-space aberration such as exposure or color changes and monocular depth estimates does not provide as accurate scene depth as stereo-based methods do. For example, this could be addressed by a latent code factoring out such variations, similarly done in NeRF-W [29]. While our "Model-2" works slightly better than our full model ("Model-4") quantitatively, we have found that our full model works usually the best for real data.

6. Conclusions

We have presented a simple yet effective algorithm for learning space-time irradiance fields from single casually captured videos. Our core technical contributions are (1) leveraging monocular video depth estimation to constrain the time-varying geometry of our learned neural implicit functions and (2) designing a static scene loss and a sampling strategy to propagate scene contents across time. We extensively validate and justify our design choices both visually and quantitatively on the Sintel dataset. We showcase free-viewpoint video rendering of several challenging dynamic scenes captured with hand-held cellphone cameras.

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. 3

[2] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *CVPR*, 2020. 3

[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. 3

[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. 3

[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. 2, 3

[6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 8

[7] 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. 3

[8] Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *ICCV*, 2019. 3

[9] 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. 2, 3

[10] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996. 2, 5

[11] Łukasz Dąbała, Matthias Ziegler, Piotr Didyk, Frederik Zilly, Joachim Keinert, Karol Myszkowski, H-P Seidel, Przemyslaw Rokita, and Tobias Ritschel. Efficient multi-image correspondences for on-line light field video processing. In *Computer Graphics Forum*, 2016. 3

[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 Transactions on Graphics (TOG)*, 35(4):1–13, 2016. 2, 3

[13] Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015. 3

[14] Chen Gao, Ayush Saraf, Jia-Bin Huang, and Johannes Kopf. Flow-edge guided video completion. In *ECCV*, 2020. 3, 8

[15] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, 2020. 3

[16] 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. 3

[17] 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. 3

[18] Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. Temporally coherent completion of dynamic video. *ACM TOG (Proc. SIGGRAPH Asia)*, 35(6):1–11, 2016. 3

[19] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *CVPR*, 2018. 3

[20] Shachar Ilan and Ariel Shamir. A survey on data-driven video completion. *Computer Graphics Forum*, 34(6):60–85, 2015. 3

[21] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. VolumeDeform: Real-time volumetric non-rigid reconstruction. In *ECCV*, pages 362–379, 2016. 3

[22] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. KinectFusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011. 3

[23] 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. 3

[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. 3

[25] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G Narasimhan, and Jan Kautz. Neural rgb (r) d sensing: Depth and uncertainty from a video camera. In *CVPR*, 2019. 3

[26] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 2, 3

[27] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM TOG (Proc. SIGGRAPH)*, 39(4), 2020. 2, 3, 6, 8

[28] R Mantiuk and V Sundstedt. State of the art on neural rendering. *STAR*, 39(2), 2020. 3

[29] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the wild: Neural radiance fields for unconstrained photo collections. In *arXiv*, 2020. 8

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

[31] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *ICCV*, 2019. 3

[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, 5, 6

[33] Richard A Newcombe, Dieter Fox, and Steven M Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, 2015. 2, 3

[34] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 2

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

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

[37] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM TOG (Proc. SIGGRAPH)*, 32(6):1–11, 2013. 3

[38] 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. 3

[39] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *ICCV*, 2019. 3

[40] 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 Annual Symposium on User Interface Software and Technology*, 2016. 3

[41] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 3

[42] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017. 3

[43] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, 2020. 3

[44] 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

[45] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. PIFuHD: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *CVPR*, 2020. 2

[46] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 3

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

[48] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*, 2019. 3

[49] Zachary Teed and Jia Deng. DeepV2D: Video to depth with differentiable structure from motion. In *ICLR*, 2020. 3

[50] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020. 3, 5

[51] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 3

[52] Rui Xu, Xiaoxiao Li, Bolei Zhou, and Chen Change Loy. Deep flow-guided video inpainting. In *CVPR*, 2019. 3

[53] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018. 3

[54] 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, 3, 6

[55] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2, 3

[56] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM TOG (Proc. SIGGRAPH)*, 2018. 3

[57] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *ECCV*, 2016. 3

[58] 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. 2, 3

[59] Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb. State of the art on 3d reconstruction with rgb-d cameras. *Computer graphics forum*, 37(2):625–652, 2018. 3