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 cam-
ing accurate scene geometry is challenging due to untex-

ture 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], multiview 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
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 \{I_0, I_1, \ldots, I_{N-1}\} with \(N\) frames, and (2) a binary mask \(M\) 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 \(x = (x, y, z)\) and the normalized viewing direction \(d = (d_x, d_y, d_z)\) to the corresponding volume density \(\sigma\) and color \(c = (r, g, b)\). 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
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.

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 \( \sigma^s \) and color \( \mathbf{c}^s \):

\[
(\sigma^s, \mathbf{c}^s) = \text{MLP}_0(\mathbf{r}(u_k)) \tag{1}
\]

where MLP0 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]:

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

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

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 \( u_t \) and \( u_f \) [32]. \( T^s(u_k) \) indicates the accumulated transmittance from \( u_t \) to \( u_k \).

Static rendering photometric loss. To train the weights \( \theta_s \) 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 \( r_{ij}(u) = \mathbf{o}_i + (u) \mathbf{d}_{ij} \). We can then optimize \( \theta_s \) 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 \( \mathbf{M}(\mathbf{r}_{ij}) = 0 \)):

\[
\mathcal{L}_{\text{static}} = \sum_{ij} \| (\mathbf{C}^s(\mathbf{r}_{ij}) - \mathbf{C}^s(\mathbf{r}_{ij})) \cdot (1 - \mathbf{M}(\mathbf{r}_{ij})) \|_2^2 \tag{4}
\]
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 $(r(u_k), t)$ as input and predict 3D scene flow vectors $s_{fw}$, $s_{bw}$, volume density $\sigma^d$, color $c^d$ and blending weight $b$:

$$\begin{align*}
(s_{fw}, s_{bw}, \sigma^d, c^d, b) &= \text{MLP}_{\theta_d}(r(u_k), t) \quad (5)
\end{align*}$$

Using the predicted scene flow $s_{fw}$ and $s_{bw}$, we obtain the scene flow neighbors $r(u_k) + s_{fw}$ and $r(u_k) + s_{bw}$. 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 $\sigma^d$ and color $c^d$ through querying the same MLP model at $r(u_k) + s$:

$$\begin{align*}
(\sigma^d_{t+1}, c^d_{t+1}) &= \text{MLP}_{\theta_d}(r(u_k) + s_{fw}, t + 1) \quad (6) \\
(\sigma^d_{t-1}, c^d_{t-1}) &= \text{MLP}_{\theta_d}(r(u_k) + s_{bw}, t - 1) \quad (7)
\end{align*}$$

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

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

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

$$L_{dyn} = \sum_{t' \in \{t-1, t+1\}} \sum_{ij} \left\| (C^d_{t'}(r_{ij}) - C^d_{t'}(r_{ij})) \right\|_2^2 \quad (9)$$

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, 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).

**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 induced optical flow with the estimated 2D optical flow.

\[ L_{\text{slow}} = \sum_{ij} \|s_{fu}(r_{ij}, t) + s_{nu}(r_{ij})\|_1 + \|s_{nu}(r_{ij})\|_1 \]  
\[ L_{\text{smooth}} = \sum_{ij} \|s_{fu}(r_{ij}, t) + s_{nu}(r_{ij})\|_2 + \|s_{nu}(r_{ij})\|_2 \]

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:

\[ L_{\text{cyc}} = \sum_{ij} \|s_{fu}(r, t) + s_{nu}(r + s_{fu}(r, t), t + 1)\|_2 + \|s_{nu}(r, t) + s_{fu}(r + s_{nu}(r, t), t - 1)\|_2 \]

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.

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 \( D^d \) and \( D^s \) for all the pixels in the static regions (where \( M(r_{ij}) = 0 \)):

\[ L_{\text{depth}} = \sum_{ij} \|\overline{D}^d(r_{ij}) - \overline{D}^s(r_{ij})\|_2^2 + \|\overline{D}^d(r_{ij}) - \overline{D}^s(r_{ij})\|_2^2, \]

where \( \overline{D} \) 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 close 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:

\[ C^{\text{full}}(r) = \sum_{k=1}^{K} T^{\text{full}}(\alpha^d(\sigma^d \delta_k)(1 - b) + \alpha^s(\sigma^s \delta_k) b) \]

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*.

<table>
<thead>
<tr>
<th>PSNR ↑ / LPIPS ↓</th>
<th>Jumping</th>
<th>Skating</th>
<th>Truck</th>
<th>Umbrella</th>
<th>Balloon1</th>
<th>Balloon2</th>
<th>Playground</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeRF</td>
<td>20.58 / 0.305</td>
<td>23.05 / 0.316</td>
<td>22.61 / 0.225</td>
<td>21.08 / 0.441</td>
<td>19.07 / 0.214</td>
<td>24.08 / 0.098</td>
<td>20.86 / 0.164</td>
<td>21.62 / 0.252</td>
</tr>
<tr>
<td>NeRF + time</td>
<td>16.72 / 0.489</td>
<td>19.23 / 0.542</td>
<td>17.17 / 0.403</td>
<td>17.17 / 0.752</td>
<td>17.33 / 0.304</td>
<td>19.67 / 0.236</td>
<td>13.80 / 0.444</td>
<td>17.30 / 0.453</td>
</tr>
<tr>
<td>Yoon et al. [58]</td>
<td>20.16 / 0.148</td>
<td>21.75 / 0.135</td>
<td>23.93 / 0.109</td>
<td>20.35 / 0.179</td>
<td>18.76 / 0.178</td>
<td>19.89 / 0.138</td>
<td>15.09 / 0.183</td>
<td>19.99 / 0.153</td>
</tr>
<tr>
<td>Tretschk et al. [51]</td>
<td>19.38 / 0.295</td>
<td>23.29 / 0.234</td>
<td>19.02 / 0.453</td>
<td>19.26 / 0.752</td>
<td>16.98 / 0.353</td>
<td>22.23 / 0.212</td>
<td>14.24 / 0.336</td>
<td>19.20 / 0.330</td>
</tr>
<tr>
<td>Li et al. [26]</td>
<td>24.12 / 0.156</td>
<td>28.91 / 0.135</td>
<td>25.94 / 0.171</td>
<td>22.58 / 0.302</td>
<td>21.40 / 0.225</td>
<td>24.09 / 0.228</td>
<td>20.91 / 0.183</td>
<td>23.99 / 0.205</td>
</tr>
<tr>
<td>Ours</td>
<td>24.23 / 0.144</td>
<td>28.90 / 0.124</td>
<td>25.78 / 0.134</td>
<td>23.15 / 0.146</td>
<td>21.47 / 0.125</td>
<td>25.97 / 0.059</td>
<td>23.65 / 0.120</td>
<td>24.74 / 0.118</td>
</tr>
</tbody>
</table>

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.

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 \( \{t_0, t_1, \ldots, t_{11}\} \). The input twelve-frames monocular video \( \{I_0, I_1, \ldots, I_{11}\} \) 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 \( I_i \) 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 \( 480 \times 270 \) resolution.

**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.

<table>
<thead>
<tr>
<th></th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>LPIPS ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours w/o (L_{\text{full}})</td>
<td>12.90</td>
<td>0.1549</td>
<td>0.991</td>
</tr>
<tr>
<td>Ours w/o static NeRF</td>
<td>18.81</td>
<td>0.4969</td>
<td>0.293</td>
</tr>
<tr>
<td>Ours w/o blending</td>
<td>22.23</td>
<td>0.7610</td>
<td>0.153</td>
</tr>
<tr>
<td>Ours w/o (L_{\text{sparsity}})</td>
<td>22.32</td>
<td>0.7884</td>
<td>0.137</td>
</tr>
<tr>
<td>Ours w/o (L_{\text{motion}})</td>
<td>22.53</td>
<td>0.7972</td>
<td>0.124</td>
</tr>
<tr>
<td>Ours w/o rigidity</td>
<td>22.61</td>
<td>0.8027</td>
<td>0.137</td>
</tr>
<tr>
<td>Ours w/o (L_{\text{depth}})</td>
<td>22.73</td>
<td>0.8142</td>
<td>0.118</td>
</tr>
<tr>
<td>Ours w/o mask</td>
<td>22.99</td>
<td>0.8170</td>
<td>0.117</td>
</tr>
<tr>
<td>Ours</td>
<td>23.43</td>
<td>0.8205</td>
<td>0.102</td>
</tr>
<tr>
<td>Ours</td>
<td>23.65</td>
<td>0.8452</td>
<td>0.093</td>
</tr>
</tbody>
</table>

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 \(L_{\text{depth}}\), our dynamic NeRF model learns the correct relative depth and renders plausible content.

**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


