View Synthesis of Dynamic Scenes based on Deep 3D Mask Volume

Kai-En Lin*, Guowei Yang*, Lei Xiao, Feng Liu, Ravi Ramamoorthi, *Fellow, IEEE*

Abstract—Image view synthesis has seen great success in reconstructing photorealistic visuals, thanks to deep learning and various novel representations. The next key step in immersive virtual experiences is view synthesis of dynamic scenes. However, several challenges exist due to the lack of high-quality training datasets, and the additional time dimension for videos of dynamic scenes. To address this issue, we introduce a multi-view video dataset, captured with a custom 10-camera rig in 120FPS. The dataset contains 96 high-quality scenes showing various visual effects and human interactions in outdoor scenes. We develop a new algorithm, Deep 3D Mask Volume, which enables temporally-stable view extrapolation from binocular videos of dynamic scenes, captured by static cameras. Our algorithm addresses the temporal inconsistency of disocclusions by identifying the error-prone areas with a 3D mask volume, and replaces them with static background observed throughout the video. Our method enables manipulation in 3D space as opposed to simple 2D masks, We demonstrate better temporal stability than frame-by-frame static view synthesis methods, or those that use 2D masks. The resulting view synthesis videos show minimal flickering artifacts and allow for larger translational movements.

Index Terms—Computer Vision, View Synthesis

1 Introduction

Recent advances in view synthesis have shown promising results in creating immersive virtual experiences from images. Nonetheless, in order to reconstruct compelling and intimate interaction with the virtual scene, the ability to incorporate temporal information is much needed. In this paper, we study a specific setup where the input videos are from static, binocular cameras and novel views are mostly extrapolated from the input videos, similar to the case in StereoMag [2]. We believe that this case is useful as dual- and multi-camera smartphones are gaining traction and it could also prove to be interesting for 3D teleconferencing, surveillance or playback on virtual reality headsets. Moreover, we can acquire the dataset from a static camera rig as shown in Fig. 1. Although we can apply state-of-the-art image view synthesis algorithms [1]–[4] on each individual video frame, the results lack temporal consistency and often show flickering artifacts. The issues mostly come from the unseen occluded regions as the algorithm predicts them on a per-frame basis. The resulting estimations are not consistent across the time dimension, which causes some regions to become unstable when shown in a video.

In this paper, we address the temporal inconsistency when extrapolating views by exploiting the static background information across time. To this end, we employ a 3D mask volume, which allows manipulation in 3D space as opposed to a 2D mask, to reason about moving objects in the scene and reuse static background observations across the video. As shown in Fig. 4, we first promote the instantaneous and background inputs into two sets of multiplane images (MPI) [2] via an MPI network. Then, we warp the same set of input images to create a temporal plane sweep volume, providing information about the 3D structure of the scene. The mask network converts this volume to a 3D mask volume which allows us to blend between the two sets of MPIs. Finally, the blended MPI volume can render novel views with minimal flickering artifacts.

To train this network, we also introduce a new multi-view video dataset to address the lack of publicly available data. We build a custom camera rig comprised of 10 action cameras and capture high-quality 120FPS videos with the static rig (see Fig. 1). Our dataset contains 96 dynamic scenes of various outdoor environments and human motions. We show that the proposed method generates temporally stable results against previous state-of-the-art methods, while only using two input views.

Our contributions can be summarized as:

- a multi-view video dataset composed of 96 dynamic scenes (Sec. 3);
- a novel 3D volumetric mask able to segment dynamic objects from static background in 3D, producing higher-quality and temporally stable results than state-of-the-art methods (Sec. 4.2);
- a synthetic dataset to evaluate complex background (Sec. 5.3);
- experiments including comparison to recent NeRF-based dynamic view synthesis methods (Sec. 5.2, Sec. 5.3).

This paper is an extended version of Deep 3D Mask Volume for View Synthesis of Dynamic Scenes [5]. In this journal version, we conduct further experiments to evaluate concurrent NeRF-based methods [6]–[8] in Sec. 5.2. These methods target a monocular dynamic camera setup different from our static stereo camera setup. A moving monocular camera effectively provides multiple viewpoints of the static scene components. On the contrary, static stereo cameras can only supply two viewpoints and thus their methods do not perform as well as our proposed method. To show experiments in a more controlled environment and allow for more complex backgrounds, we created a new synthetic dataset to evaluate the performance in Sec. 5.3. We demonstrate how our method can tackle the dynamic background with multiple

** denotes equal contribution.*

- *K. Lin, G. Yang and R. Ramamoorthi are with the CSE Department at the University of California, San Diego, at La Jolla, CA 92037. Email: {k2lin@, g4yang@, ravir@cs }.ucsd.edu*
- *Lei Xiao is with Reality Labs Research at Meta, at Redmond, WA 98052 Email: lei.xiao@fb.com*
- *Feng Liu is with the Computer Science Department at Portland State University, at Portland, OR 97207 Email: fliu@cs.pdx.edu*

Fig. 1. Our custom camera rig. Top left figure shows the configuration we use for evaluation in Sec. 5. We show the input stereo image sequences from camera 4 and camera 5 in the middle. The rightmost column shows the crops of rendered novel view at camera 0. Artifacts appear when the novel view is translated by a larger distance. We use the conventional MPI method [1] as our baseline algorithm. Note how the area on top of the person's head is distorted and shows "stack of cards" artifacts. This type of artifact flickers in a dynamic video as the network hallucinates the disocclusion per-frame.

actors. Moreover, we detail how different loss functions would affect the visual results in Sec. 5.4, as well as large distance view extrapolation in Sec. 5.5 and extension to more input views in Sec. 5.6.

2 Related Work

Our goal is to achieve temporally stable view synthesis on dynamic scenes. We are inspired by several previous methods in view synthesis and space-time synthesis.

2.1 View synthesis

View synthesis is a complicated problem which has become a popular field of research in computer vision and graphics. Earlier lines of work utilize dense sampling from the scene to create light fields [17], [18]. Image-based rendering techniques [19], [20] exploit proxy geometry of the scene to produce novel view renderings. Later extensions on this topic introduce better modeling of the scene structure [21] and hand-crafted heuristics [22], [23]. As deep learning became dominant, learning-based methods [24]–[28] have shown promising results. Recently, a class of research works focuses on combining novel representations [1]–[3], [9], [29]–[34] with a differentiable rendering pipeline to produce high-quality results. Another exciting advance is neural radiance fields (NeRF) [29], which encodes the 3D scene structure in a compact continuous 5D volumetric function. Although NeRF has shown promising view synthesis results, it has to overfit to the given scene with enough samples (10 or more), requiring time-consuming per-scene training. Rendering time could take up to 30s for one image, whereas our pipeline allows inference and rendering in less than 2s without dedicated optimization, using only binocular input views.

Instead, in this paper we focus on a specific layered representation, multiplane images (MPI) [1]–[3], [10], [35], as it provides good generalizability across various scenes and efficiency capable of real-time rendering. Our proposed method directly tackles the temporal instability introduced in MPIs when the disoccluded areas lead to different estimations across time.

2.2 Space-time synthesis

Space-time synthesis is a more complicated problem since it not only involves movement of the novel viewpoint in space, but also incorporates differences of time. A body of work covers appearance changes such as relighting while changing views [27], [36]–[39]. However, these methods focus on the lighting change with respect to a static scene, treating dynamic objects in the scene as outliers. On the other hand, some methods directly target dynamic scenes [10], [11], [13], [15], [40]. While our method utilizes MPIs similar to Broxton et al. [10], they employ dense sampling of 46 cameras to reconstruct light fields of the viewing volume, essentially interpolating between cameras. Our method focuses on the stereo case similar to StereoMag [2], targeting extrapolation from stereo inputs like dual-camera smartphones. In addition, unlike depth-based methods [11], [13], we do not require any explicit depth maps to render novel viewpoints. As depth-based methods often yield flickering and require hole-filling, we instead use a representation more suitable for rendering. Another issue that these methods do not address is the lack of generalizability. Bansal et al. [13] is trained on limited data which could make the learned network overfit to a small number of scenes. Moreover, while Yoon et al. [11] uses a pretrained network to ensure generalizability on unseen scenes, it still requires human-generated masks for foreground and background separation. We capture various dynamic scenes with human interactions to train our network and ensure that it is generalizable across different unseen scenes. Also, our network utilizes the background information extracted from video and uses it to directly segment the foreground and background in 3D space without any human input.

Concurrently, there are several NeRF-based algorithms [6]–[8] which demonstrate state-of-the-art performance on monocular video inputs with a moving camera. For static parts of the scene, a moving camera provides multi-view cues to the network and they can be reconstructed in the same process as the original NeRF [29]. For dynamic parts of the scene, NSFF [7] learns an implicit representation of the scene flow and warps the sampled points to render the scene at different timesteps. Similarly, NeRFlow [6] also uses an MLP network to learn the underlying scene flow but it incorporates a neural ODE to enforce consistency across continuous time. Non-rigid NeRF [8] optimizes for a canonical volume model, then it uses deformation fields to generate renderings at different timestamps. Although these methods work well for a single moving camera, they are not able to acquire good 3D geometry for a pair of static cameras. As demonstrated in

TABLE 1
Comparison of different multi-view datasets.

Dataset	Scene count	Rigid rig	Large disparity	Views	Dynamic	Public	Remarks
Real Forward-Facing [1]	65	✗	✓	25	✗	✗	Loosely gridlike formation
Spaces [9]	100	✓	✓	16	✗	✓	Strictly gridlike formation
Immersive LF Video [10]	130	✓	✓	46	✓	✗	Spherical formation
Dynamic Scene [11]	8	✓	✓	12	✓	✓	Few temporal frames
Single Image LF [12]	∼2000	✓	✗	196	✗	✓	Small baseline light fields
RealEstate10K [2]	∼10000	✗	✓	1	✗	✓	Static scenes
Open4D [13]	6	✗	✓	15	✓	✓	Free-viewpoint capture
MannequinChallenge [14]	∼2000	✗	✓	1	✗	✓	Mostly static scenes
X-Fields [15]	8	✓	✓	5	✓	✓	Few temporal frames
KITTI [16]	400	✓	✓	2	✓	✓	Binocular setup on cars
Ours	96	✓	✓	10	✓	✓	Publicly released

TABLE 2
Number of videos that contain each occlusion type as described in Sec. 3.3. Note that most scenes typically contain multiple types of occlusion.

Occlusion types	**(a)**	**(b)**	**(c)**	**(d)**	**Total videos**
Count	90	96	42	19	96

Sec. 5.2, our MPI-based method is able to utilize better geometry priors to provide high-quality results during extrapolation with less distortion and flickering.

3 Dataset

High-quality video datasets are crucial for learning-based novel-view video synthesis algorithms. The ideal datasets would contain a diversity of scenes, captured at multiple synchronized views. In this work we introduce a novel multi-view video dataset. We discuss the limitations of existing datasets compared to our dataset in Sec. 3.1. We describe our data capture and generation process in Sec. 3.2. Finally, we discuss the statistics and advanced properties of our dataset in Sec. 3.3.

3.1 Multi-view video dataset

As shown in Table 1, we evaluate several properties which are important to train a generalized view synthesis network. Specifically, a rigid camera rig is preferred as it can provide good pose priors and ensure the accuracy of the estimated camera poses. On the contrary, unstructured captures like Real Forward-Facing [1] and Open4D [13] do not use pose priors and utilize structure from motion, which could produce varying accuracy depending on scene geometry and the texture presented. In addition, rigid camera rigs allow for capture of dynamic scenes with multiple simultaneous camera views. On account of the above reasons, our dataset is captured with a custom camera rig that is rigid and robust enough to offer good pose priors.

Number of views is also an important factor for a multi-view dataset since different combinations of input and target camera pairs provide diversity in baselines and camera motions. X-Fields [15] and KITTI [16] provide limited views and camera motions and thus are not as useful for video view synthesis tasks. Our dataset offers 10 different camera views in a gridlike formation (see Fig. 1). For our binocular view synthesis task, we choose 2 views out of 10 and 1 from the rest to construct a training pair.

The most important feature is to have enough temporal frames and dynamic movements for training. Most datasets fail at this part as they target the image view synthesis task instead of a video one. Although the Dynamic Scene Dataset presented by Yoon et al. [11] targets the dynamic scenes, it uses frame skips to keep salient movements. Thus, the movements shown in the dataset are not smooth and fail to provide enough training samples. To address this issue, our dataset is captured in 120 FPS and synchronized as a post-process (see Sec. 3.2), making it easy to perform and evaluate view synthesis at different framerates.

One dataset that targets the purpose of video view synthesis is the Immersive Light Field Video dataset proposed by Broxton et al. [10], which contains 46 camera views and 130 different dynamic scenes. However, the full dataset is not publicly available to the community. Our full dataset can be found at http://cseweb.ucsd.edu/%7eviscomp/projects/ICCV21Deep/

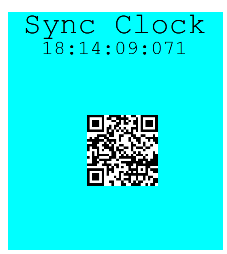


Fig. 2. Digital clock and the randomly moving QR code pattern used to perform synchronization. We have two ways to do synchronization: (1) matching the timestamp; (2) aligning the QR code location in all views. We use these methods to ensure the synchronization is accurate enough.

3.2 Dataset generation

Our video dataset is captured with a custom camera rig that consists of 10 GoPro Hero 7 Black action cameras as shown in Fig. 1. The horizontal baseline between neighbor cameras is approximately 10 cm and the vertical distance between rows is around 14 cm. We captured 96 outdoor videos in 120 FPS, with the camera rig being static for each video. As GoPros only allow fisheye mode for high FPS captures, we calibrate the cameras with a 17x14 checkerboard pattern (squares have side lengths of 40mm) and undistort the videos using a pinhole camera model [41] implemented in OpenCV [42]. For camera extrinsics, we choose the first frame from all views as inputs to COLMAP [43], [44], which then does feature extraction, feature matching, and sparse reconstruction. The reconstructed camera poses are

(a) Walking (b) Jumping (c) Sitting (d) Thin Structures

(e) Dynamic Occlusion (f) Static Occlusion (g) Two Dynamic Subjects (h) Specularity

Fig. 3. A selection of still frames from our dataset. We captured various dynamic scenes with human motions, including walking, running, jumping and sitting down. Note that cameras remain static for the whole duration of the capture.

assumed to be fixed for the duration of each video. In addition, to achieve synchronization, we display a digital clock with randomly appearing QR code patterns (see Fig. 2) on a high refresh rate screen that can be seen by all cameras at the same time. Then, we manually edit and align the multi-view videos according to the digital clock and QR code pattern.

3.3 Dataset statistics

Our videos are mostly around 1 to 2 minutes long and all videos are shot in 120 FPS. We cover different scenes to ensure that the surface reflectance variety is high enough. For example, in Fig. 3 we show that in our dataset we cover different buildings, furniture, foliage and specularity effects. Another important aspect of our dataset is the inclusion of different human motions, including slower motions like walking, sitting down and faster motions, such as running, jumping and arms waving. We now discuss four possible types of occlusion interactions and show the numbers of their occurrences in Table 2.

(a) Static occluder and static background. For example, the table in Fig. 3c occludes the areas behind. Most view synthesis methods target this case as this is one of the most common cases. We describe it as a static occluder in the scene blocking the line-of-sight from the cameras to the background scene. Background information can only be acquired from the views with direct line-of-sight. As such, it is difficult to recover the unseen regions without prior knowledge of the scene. However, temporal consistency in these areas is easily achievable because inputs remain relatively unchanged throughout the video. Hallucination of the disoccluded areas can also remain the same for this case.

(b) Dynamic occluder and static background. Another type of event happens when a dynamic object is moving across the scene. For example, the person in Fig. 3a and Fig. 3e walks in front of the static background. In these scenes, the camera has line-of-sight on the background behind the person at some point in the video. Therefore, it is relatively easy to acquire static background information as the occluder does not block the line-of-sight in all video frames. Combining information from multiple frames throughout the video provides an accurate rendering of what is behind the dynamic occluder. Temporal consistency in this case can also be maintained by substituting the static background for the dynamically-occluded regions. In other words, we can perform hole-filling based on the observations from other video frames. Our proposed method takes advantage of this prior knowledge to generate temporally-stable view synthesis results, as opposed to previous methods.

(c) Static occluder and dynamic background. This case happens when an object moves behind a static occluder and thus the camera does not have full visuals on it. For example, the person walks behind a traffic sign in Fig. 3f. In this case as it is only a short-term occlusion, the person's appearance can be interpolated between different frames. However, in the case of a larger wall, this becomes difficult to solve as extrapolating the movement is complicated and the ambiguity could lead to different outcomes. In general, it is difficult to accurately predict the trajectory of the occluded object without assuming it is moving at constant velocity. For temporal consistency, the movement of dynamic objects can lead to instability of the novel view prediction. Our method learns to detect the dynamic movements and treat the static part of the scene as (a) such that flickering artifacts are kept at a minimal level.

(d) Dynamic occluder and dynamic background. The last case happens when the occluder and the background object are both moving or the background appearance is changing. For instance, two people walk towards each other parallel to the camera's image plane in Fig. 3g. Similar to (c), how the occluded object is moving remains ambiguous and hard to resolve deterministically. Although we do not have a clear idea of the occluded parts, we can still ensure it is temporally stable when shown. We can reduce this case to (b) with the ambiguity that the occluded object can move anywhere. And as a result, the occluded regions look more or less similar to the static background.

Our dataset contains diverse occlusion interactions and we show results in Sec. 5.2 and provide an analysis in Fig. 6.

4 Deep 3D Mask Volume

Our goal is to synthesize temporally consistent novel view videos given stereo video inputs. Consequently, we build our algorithm upon prior work on multiplane images [1], [2] and propose a novel mask volume structure to fully utilize the temporal background information and the layered representation. In this section, we start with a brief review of the multiplane images in Sec. 4.1. Then we describe our 3D mask volume in Sec. 4.2. Finally we

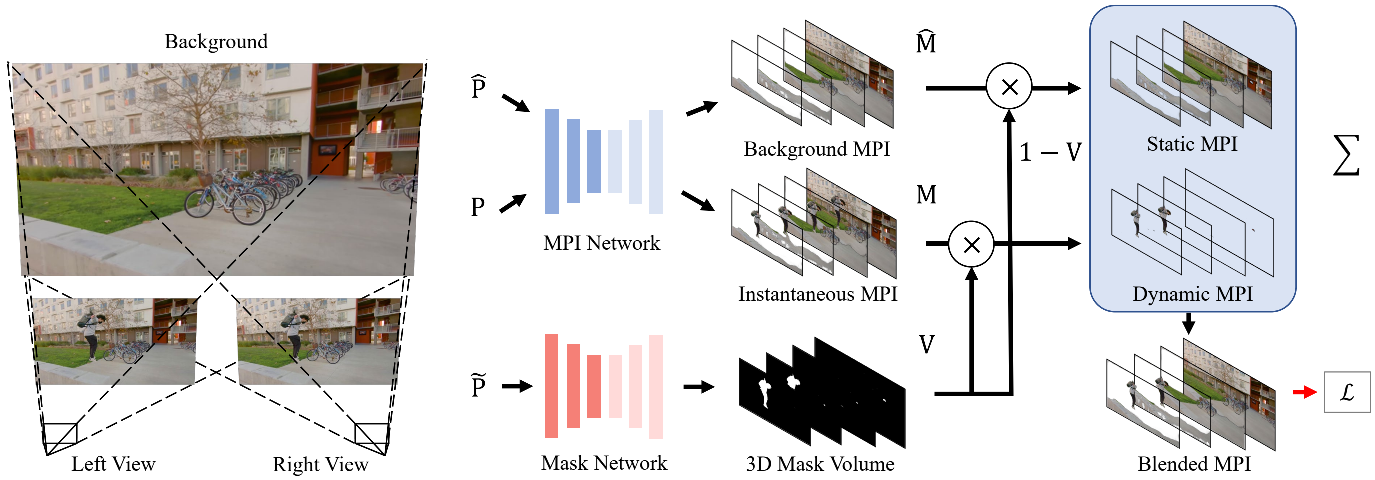


Fig. 4. Overview of our pipeline. Given binocular input videos, our MPI network promotes the 2D multiview images to two 3D MPI representations; one encodes the instantaneous information and the other encodes the background information. The mask network produces a 3D mask volume **V** to modulate the MPIs and blend them together, producing the final output. Please see Sec. 4.2 for more details.

discuss our loss function design in Sec. 4.4. Please refer to Fig. 4 for an overview of our algorithm pipeline.

4.1 Multiplane images

Our approach takes inspiration from recent advancements in multiplane image representation [2], [45]. Multiplane images (MPI) are a layered representation of the 3D scene. They consist of D layers of RGBα images, representing the viewing frustum from the perspective of a virtual reference camera. The planes partition the viewing frustum according to equally-spaced disparity (inverse depth) values $d_0, d_1, ..., d_{D-1}$. Each layer of the MPI encodes color C and transparency information α at a specified plane depth d. We denote the MPI layer at disparity d as a tuple of (C_d, α_d). To construct such a volume, we warp input views to the reference camera position (in our case, the center left camera, numbered 4 in the camera rig diagram in Fig. 1) to construct a plane sweep volume (PSV). The PSV is then used as the input to a 3D CNN similar to the one used by Mildenhall et al. [1] and it generates the corresponding MPI volume. To render a novel viewpoint j from camera i, the MPI layers are warped using planar homography as follows:

$$\mathcal{W}^d_{i\to j}(C_d, \alpha_d), \tag{1}$$

where $\mathcal{W}$ is the warping operator. The warped MPIs are then composited with the over operation. To be more specific, we calculate the per-pixel transmittance t from the alpha value at location (x, y) on plane d by

$$t(x, y, d) = \alpha(x, y, d) \prod_{d'>d} [1 - \alpha(x, y, d')]. \tag{2}$$

The final rendering at each pixel C_{final} is computed as

$$C_{final}(x, y) = \sum_d C(x, y, d)t(x, y, d) \tag{3}$$

These computations are parallelizable and their efficiency during rendering makes the MPI a good representation for fast view synthesis.

One observation of MPIs is that the unseen parts in the volume are often merely repeated texture of the foreground objects [3]. This happens when the input camera baseline is not large enough and the resulting PSV cannot provide further information about the background. In addition, these areas typically present different estimations between frames. Therefore, the unseen areas produce visible artifacts, especially in video view synthesis (see Fig. 1). On the other hand, visible parts usually provide temporally stable results as can be seen in Broxton et al. [10]

4.2 3D mask volume generation

From Sec. 4.1, we observe that most artifacts are introduced by the disocclusion of moving objects. In order to address this issue, we seek to find a 3D mask volume that identifies the dynamic components and removes the flickering artifacts behind them accordingly. To be more specific, given a pair of stereo image sequences of length n, $\{\mathbf{I}^L_0, \mathbf{I}^L_1, ..., \mathbf{I}^L_{n-1}\}$ and $\{\mathbf{I}^R_0, \mathbf{I}^R_1, ..., \mathbf{I}^R_{n-1}\}$, we wish to derive a 3D mask $\mathbf{V}(x, y, d)$, such that

$$\mathbf{V}(x, y, d) = \begin{cases} 1, & \text{if } \mathbf{I}(x, y) \neq \hat{\mathbf{I}}(x, y), d > \mathbf{D}(x, y) \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

where $\mathbf{I}$ is the instantaneous frame, $\hat{\mathbf{I}}$ denotes the background image, and $\mathbf{D}$ is the scene disparity observed by the camera. We drop the frame subscript as a shorthand for instantaneous frame in the following discussion. In addition, we represent the instantaneous MPI of the scene as $\mathbf{M}(x, y, d)$, and the background MPI as $\hat{\mathbf{M}}(x, y, d)$.

The main purpose of the 3D mask volume $\mathbf{V}(x, y, d)$ is to partition the scene $\mathbf{M}(x, y, d)$ into two parts: static and dynamic. The static portion of the MPI does not change for the whole video duration, and thus $\mathbf{M}(x, y, d) = \hat{\mathbf{M}}(x, y, d)$, when $\mathbf{V}(x, y, d) = 0$. The synthesized novel view of these parts is temporally stable and requires no further modification to the algorithm. On the contrary, the dynamic objects ($\mathbf{V}(x, y, d) = 1$) could move in different directions. The disoccluded areas, given mathematically by $\mathbf{M}(x, y, d)$ if $\mathbf{I}(x, y) \neq \hat{\mathbf{I}}(x, y), d > \mathbf{D}(x, y)$, often change with them, producing "stack of card" artifacts and flickering when viewed from another angle (see Fig. 1). However these areas in fact usually resemble the background $\hat{\mathbf{I}}$. With this knowledge, a clear separation between the static and dynamic scene components allows us to identify the disocclusion and minimize the temporal inconsistency by

$$\mathbf{M}(x,y,d) \leftarrow \hat{\mathbf{M}}(x,y,d) \text{ if } \mathbf{I}(x,y) \neq \hat{\mathbf{I}}(x,y), d < \mathbf{D}(x,y). \tag{5}$$

Essentially, we are using the temporally-stable static background to replace the unknown disoccluded areas. An illustration of the mask is given in Fig. 4.

In order to perform the operation in Eq. 5, our network is composed of two networks: **MPI network** generates 2 layered representations of the 3D scene, namely $\mathbf{M}(x,y,d)$ and $\hat{\mathbf{M}}(x,y,d)$; **Mask network** produces the 3D mask volume $\mathbf{V}(x,y,d)$ satisfying Eq. 4. We show each network in Fig.4 and discuss them in details as follows:

MPI network. It is necessary to acquire 3D information from both the instantaneous frame and throughout the whole video, so we can then obtain the needed information behind the dynamic occluder. To this end, we first apply a median filter $\mathcal{A}$ on the image sequences

$$\hat{\mathbf{I}} = \mathcal{A}(\{\mathbf{I}_0, \mathbf{I}_1, ..., \mathbf{I}_{n-1}\}). \tag{6}$$

It is applied to both views to generate the corresponding background images.

Then, we can inversely warp $\mathbf{I}^R$ and $\hat{\mathbf{I}}^R$ to the left camera and construct a PSV. The PSV from the instantaneous frame is generated as

$$\mathbf{P} = \{\mathbf{I}^L, \mathcal{W}^{d_0}_{R\to L}(\mathbf{I}^R), \mathcal{W}^{d_1}_{R\to L}(\mathbf{I}^R), ..., \mathcal{W}^{d_{D-1}}_{R\to L}(\mathbf{I}^R)\}. \tag{7}$$

It is then used as an input to a 3D CNN $\mathcal{F}_\theta$ to produce the instantaneous MPI, $\mathbf{M} = \mathcal{F}_\theta(\mathbf{P})$. Similarly, we construct the background MPI, $\hat{\mathbf{M}}$, using another PSV, $\hat{\mathbf{P}}$, generated from $\hat{\mathbf{I}}^L$ and $\hat{\mathbf{I}}^R$. The two MPIs, $\mathbf{M}$ and $\hat{\mathbf{M}}$, now contain the information of the dynamic occluder and the static background.

Mask network. We utilize another 3D CNN $\mathcal{G}_\theta$ to reason about the relationship between the MPIs and generate a mask volume $\mathbf{V}$ to satisfy Eq. 4. Inspired by background matting [46] on 2D images, our mask network takes a similar approach but in 3D space. From Eq. 6, we define a temporal plane sweep volume (TPSV) as follows

$$\tilde{\mathbf{P}} = \{\mathbf{I}^L, \mathcal{W}^{d_0}_{R\to L}(\mathbf{I}^R), ..., \mathcal{W}^{d_{D-1}}_{R\to L}(\mathbf{I}^R), \hat{\mathbf{I}}^L, \mathcal{W}^{d_0}_{R\to L}(\hat{\mathbf{I}}^R), ..., \mathcal{W}^{d_{D-1}}_{R\to L}(\hat{\mathbf{I}}^R)\}. \tag{8}$$

The TPSV helps the network to distinguish the dynamically-occluded parts in the 3D scene. Then, we acquire the 3D mask volume by $\mathbf{V} = \mathcal{G}_\theta(\tilde{\mathbf{P}})$.

Finally, we can calculate the final MPI $\mathbf{M}_o$ by:

$$\mathbf{M}_o(x,y,d) = \mathbf{M}(x,y,d)\mathbf{V}(x,y,d) + \hat{\mathbf{M}}(x,y,d)(1 - \mathbf{V}(x,y,d)), \tag{9}$$

for all (x,y,d). We define a shorthand version as

$$\mathbf{M}_o = \mathbf{M} \odot \mathbf{V} + \hat{\mathbf{M}} \odot (1 - \mathbf{V}), \tag{10}$$

where $\odot$ means element-wise multiplication. $\mathbf{M}_o$ achieves Eq. 5 as our learnable mask volume $\mathbf{V}$ satisfies Eq. 4 and we can then render the output image $\mathbf{I}_o$ using planar homography and the over composite operation described in Sec. 4.1. Please refer to Fig.4 for illustrations.

One major difference between using a 3D mask volume $\mathbf{V}(x,y,d)$ and a 2D mask $\mathbf{V}'(x,y)$ is that the former is able to segment out the dynamic objects in the 3D space, namely Eq. 4 and subsequently do Eq. 5. In Fig. 4, notice that the mask volume only contains the dynamic object (jumping person in this case). In contrast, a 2D mask $\mathbf{V}'(x,y)$ does not vary with respect to the disparity d, making it impossible to manipulate the areas behind dynamic objects.

TABLE 3
Details of each layer in our 3D mask network.

Layer	kernel size	stride	dilation	in	out	activation	input
conv1_1	7	1	1	12	8	ReLU	PSVs
conv1_2	7	2	1	8	16	ReLU	conv1_1
conv2_1	3	1	1	16	16	ReLU	conv1_2
conv2_2	3	2	1	16	32	ReLU	conv2_1
conv3_1	3	1	1	32	32	ReLU	conv2_2
conv3_2	3	2	1	32	64	ReLU	conv3_1
conv4_1	3	1	1	64	64	ReLU	conv3_2
conv4_2	3	1	1	64	64	ReLU	conv4_1
up5	-	2	-	128	128	-	conv3_2 + conv4_2
conv5_1	3	1	1	128	32	ReLU	nnup5
conv5_2	3	1	1	32	32	ReLU	conv5_1
up6	-	2	-	64	64	-	conv2_2 + conv5_2
conv6_1	3	1	1	64	16	ReLU	nnup6
conv6_2	3	1	1	16	16	ReLU	conv6_1
up7	-	2	-	32	32	-	conv1_1 + conv6_2
conv7_1	3	1	1	32	16	ReLU	nnup7
conv7_2	3	1	1	16	8	ReLU	conv7_1
conv7_3	3	1	1	8	1	Sigmoid	conv7_2

4.3 Network Architecture

Our view synthesis pipeline utilizes two different 3D CNNs to predict the MPI volumes and the 3D mask volume as described in Sec. 4.2. Both networks have similar structures as the one in Mildenhall et al. [1]. However, we made some adjustments to keep the network light for faster training and less memory consumption. We show detailed layers for the mask network in Table 3. The MPI network has the same structure except for some changes in the overall input and output channels to account for different view counts.

4.4 Loss function

We implement our loss function as a rendering loss, similar to previous work on MPIs [1]–[3]. For the rendering loss, we use view synthesis as the supervision task and let the algorithm render a held-out view from the final MPI $\mathbf{M}_o$ (see Fig.4). The rendering loss is as follows:

$$\mathcal{L} = \frac{||\mathcal{F}_{VGG}(\mathbf{I}_o) - \mathcal{F}_{VGG}(\mathbf{I}_{gt})||_1}{N}, \tag{11}$$

where $\mathcal{F}_{VGG}$ is the VGG-19 network [47], N is the number of elements in the image $\mathbf{I}_o$, and $\mathbf{I}_{gt}$ is the held-out ground truth view. This perceptual loss is similar to the implementation of Chen et al. [48]. We also considered a mask supervision loss $\mathcal{L}_m$ and a mask sparsity constraint $\mathcal{L}_s$. However, we did not find them to be useful for temporal consistency. Ablation studies on these two losses can be found later in Table 6, and details are in Sec. 5.4.

5 Results

In this section, we discuss implementation details for our network in Sec. 5.1. Then we show comparisons to other methods on our dataset in Sec. 5.2. We include comparisons on our synthetic dataset in Sec. 5.3. To explore the effects of different loss functions, we show the ablation studies in Sec. 5.4. We show that our method is able to degrade gracefully even doing view extrapolation

Fig. 5. We show visual results on 4 different scenes. These scenes include both fast and slow movements, such as waving, jumping and walking. The novel viewpoint is an extrapolation from the input camera views. In the above images, each row is rendered using one method from (a) MPI/LLFF [1], (b) 2D Mask, (c) NeRFlow [6], (d) Non-rigid NeRF [8], (e) Neural Scene Flow Field [7], (f) IBRNet [49], and (g) ours. Last row (h) is the ground truth. Our proposed method produces results with fewer artifacts and more temporal stability.

far outside the viewing volume in Sec. 5.5. Our method can also be extended to incorporate more input views in Sec. 5.6. Finally we discuss limitations of our current setup and method in Sec. 5.7. Result videos can be found in the supplementary materials.

5.1 Implementation details

Due to GPU memory constraints, we choose a two-step training scheme to train our network. We first train the MPI network on the RealEstate10K dataset [2], and then train only the mask network on our own video dataset. This training scheme can keep the memory usage within a reasonable range and the speed fast enough.

The MPI generation network is trained by predicting a held-out novel view and applying the rendering loss $\mathcal{L}$ as supervision. This stage is trained for 800K steps. After the previous pretraining stage, we freeze the weights of the MPI network and train only the mask network using the loss $\mathcal{L}$. The network takes 2 random views from the 10 views as input and we randomly choose a target camera position from the rest of the views at each step. We select 86 out of the 96 scenes as our training dataset and images are rescaled to 640×360. This second stage is trained for 100K steps. The learning rate is set to $5e-4$ for both stages. Our training pipeline is implemented in PyTorch [50] and training takes around 5 days on a single RTX 2080Ti GPU. With resolution in 640×360, inferencing $\mathbf{M}_o$ using our full pipeline takes around 1.75s, while rendering takes another 0.28s. Note that the rendering pipeline is implemented in PyTorch without further optimization. In practice, it could be significantly faster with OpenGL or other rasterizer.

5.2 Comparisons on real data

For comparison, we choose 7 unseen videos from the dataset and subdivide them into 14 clips, focusing on salient movements in the scene. The methods we chose to evaluate includes MPI-based methods like LLFF, and also emerging NeRF-based methods like Nonrigid-NeRF, Neural Scene Flow Field, and NeRFlow We ran all methods on the clips with camera 4 and 5 as input and others as the target output (see Fig. 1). Error metrics are calculated between the output and the ground truth images. For monocular NeRF-based methods [6]–[8], as they assume the input to be monocular, moving camera, and have increasing time steps, we

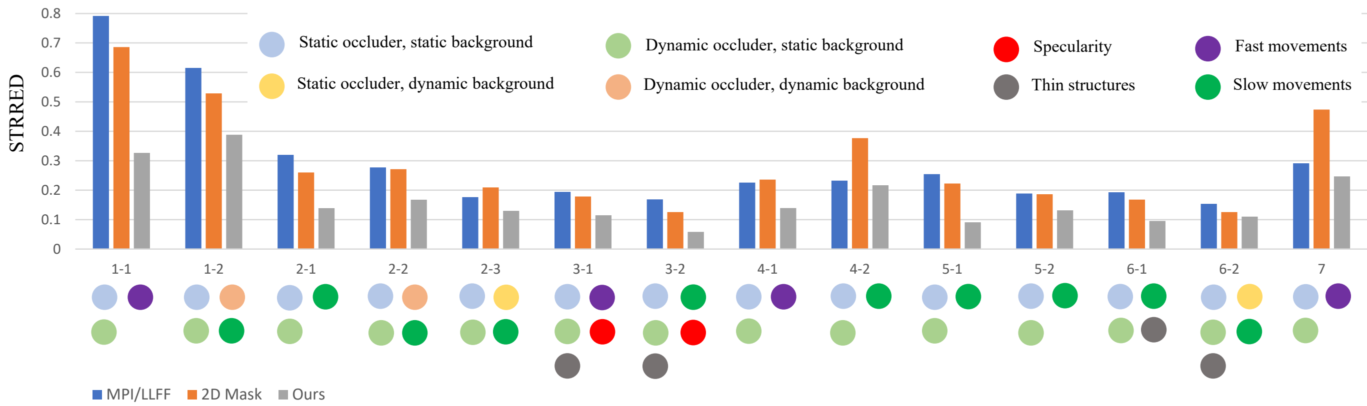


Fig. 6. STRRED comparison on our dataset with baseline methods. We select 14 clips from 7 different scenes. 1-1, 1-2 denotes clip 1 and clip 2 from scene 1.

alternate between left and right views to satisfy this assumption. This allows the algorithm to treat the input as a monocular video with the camera jumping between two viewpoints.

We compare with 6 baseline approaches: (1) MPI/LLFF is our adaptation of Mildenhall et al. [1] to work with only two input views and different camera intrinsics. It processes the stereo input videos and renders the novel view frames on a per-frame basis. We trained it on the same dataset as our method. (2) 2D mask is our naive baseline method, which is similar to our pipeline, except that it uses a foreground mask $\mathbf{V}'(x, y)$ generated by the background matting method [46] with $\mathbf{I}$ and $\hat{\mathbf{I}}$ as inputs. The blended MPI $\mathbf{M}_o'$ for (2) is obtained by

$$\mathbf{M}_o' = \mathbf{V}'(x, y) \odot \mathbf{M} + (1 - \mathbf{V}'(x, y)) \odot \hat{\mathbf{M}},$$

where the 2D mask has been expanded into 3D by repeating its values along the depth dimension. (3) IBRNet [49] uses the official implementation and their pretrained model weights, and it takes 2 views as input on a per-frame basis. (4) NeRFlow [6] uses the official implementation and we slightly modify the necessary parts to allow for two alternating views as input. (5) NSFF [7] is also adapted from the official implementation to take two input views. (6) Non-rigid NeRF [8] uses the released official implementation with modifications to enable two-view inputs. For (4)-(6), we train them for 20,000 steps for each scene and render the corresponding viewpoints. Please refer to our supplementary materials for the video results.

TABLE 4
Comparison on our evaluation dataset. We compare with different baseline methods and the results show that our 3D mask offers much better temporal stability. 2D mask does not improve much because it fails to resolve the ambiguity in disoccluded areas.

Methods	**Mask**	**STRRED↓**	**PSNR↑**	**SSIM↑**
MPI/LLFF [1]	✗	0.2917	25.52	0.8227
2D Mask	2D	0.2892	25.50	0.8242
IBRNet (2-view) [49]	✗	2.2606	21.49	0.6713
NeRFlow [6]	✗	3.2646	16.81	0.4146
NSFF [7]	✗	1.4230	17.04	0.4197
Non-rigid NeRF [8]	✗	2.3941	18.11	0.4997
Ours	3D	**0.1683**	**26.22**	**0.8390**

From Table 4, we see that our method is able to achieve temporally-coherent rendering, while offering better visual quality and fewer distortions. Specifically, we employ the STRRED metric [51] to evaluate stability across time. Our method significantly reduces the temporal artifacts across most scenes while also keeping PSNR and SSIM better than the baseline methods. For MPI/LLFF, since it does not utilize the information across the whole video, it yields more flickering and distorted areas as can be seen in Fig. 5. For example, in the top scene, there is a ghosting artifact around the person's head and it changes frame-by-frame, resulting in flickering video. The 2D mask method is a binary mask that naively selects the dynamic parts in $\mathbf{M}$ and the background in $\hat{\mathbf{M}}$ to produce the final MPI. As a result, it amplifies the stack of cards artifacts (see Fig. 5) and also slightly worsens the visual quality as shown in Table 4. IBRNet [49], does not work well with 2-view input and it produces poor results compared to ours. Concurrent monocular NeRF-based methods [6]–[8] perform similarly in Table 4. With only two input viewpoints, they fail to represent even the static scene components since there are not enough multi-view cues for reconstruction. For dynamic parts of the scene, NSFF provides more stable quality as can be seen from the STRRED metric. In general, our proposed method provides state-of-the-art performance over other previous and concurrent work. We show qualitative results in Fig. 5. Each inset column corresponds to a scene as shown on the leftmost side. We show the MPI baseline method in row (a) and 2D mask baseline in row (b). These two methods suffer from stack-of-card artifacts in particular in the disoccluded regions. 2D mask fails to solve the problem and sometimes makes it more apparent. This is because 2D mask does not reason about the 3D geometry of the scene. For the more recent NeRF-based methods, we show them in row (c-f). NeRFlow [6] provides better static scene reconstruction than other methods. However, it produces blurred results and lacks high-frequency details as can be seen from the second image in row (c). On the other hand, our proposed method is able to make the text on the person more legible and sharper, while suffering little to no disocclusion artifacts. Non-rigid NeRF [8] suffers from significant artifacts when rendering the images. This is possibly due to sparse viewpoints and the network is trying to compensate with deformation fields. NSFF [7] generates sharper images than NeRFlow, but it suffers from blurriness in static parts of the scene. IBRNet [49] produces noisy results given two input views on a frame-by-frame basis. Their method tries to blend different viewpoints with a ray transformer to synthesize disoccluded regions. However, given two input views, this becomes even more difficult

Fig. 7. We show visual results on 4 different synthetic scenes. These scenes include moving characters and dynamic backgrounds. In the above image, each row is rendered using one method from (a) MPI/LLFF [1], (b) 2D Mask, (c) NeRFlow [6], (d) Non-rigid NeRF [8], (e) Neural Scene Flow Field [7], (f) IBRNet [49], and (g) ours. Last row (h) is the ground truth. Our proposed method (g) produces results with fewer artifacts and more temporal stability.

because of the lack of samples.

To further analyze how temporal consistency is affected, we characterize the clips with different properties including different types of occlusion discussed in Sec. 3.3 and show the results in Fig. 6. As stated earlier, several clips are selected from the 7 scenes to show salient motions. We only include results from MPI/LLFF [1], 2D mask and Ours, as other methods have significantly higher STRRED. From the results, we observe that faster movements could often result in worse temporal consistency, like the differences between clip 1-1 and 1-2. There is an interesting failure in 4-2 for the 2D mask method. 4-1 is the jumping scene in Fig. 5, and 4-2 shows a person walking in the same scene. Although the movement is slower, the person walks past several areas with large appearance changes in 4-2. As a result, the artifacts in the 2D mask are much more obvious, and the video flickers more than other methods, leading to a worse STRRED score.

5.3 Comparisons on synthetic data

In addition to real data, we also crafted a synthetic dataset and tested different methods on it. The synthetic dataset not only can provide us real ground truth to make proper comparisons, but also can illustrate scenes and movements hard to capture in real life, for example, complex moving backgrounds. The synthetic dataset is constructed using scenes from the Habitat-Matterport 3D dataset [52] and UE4 Sun Temple [53], and the moving characters in the scene are pre-animated characters from Adobe Mixamo. We used Blender [54] to composite the scenes, and replicated the 10-view camera array with parameters similar to our GoPro setup. We deliberately set all the cameras to have the same camera intrinsics in order to reduce unwanted artifacts.

For each scene, we rendered 60 frames of the animation, and produced camera poses for all 10 cameras. As all the camera poses can be directly obtained from Blender, we do not need COLMAP [43], [44] to estimate camera poses anymore. The background images are still obtained using median filter. Similar

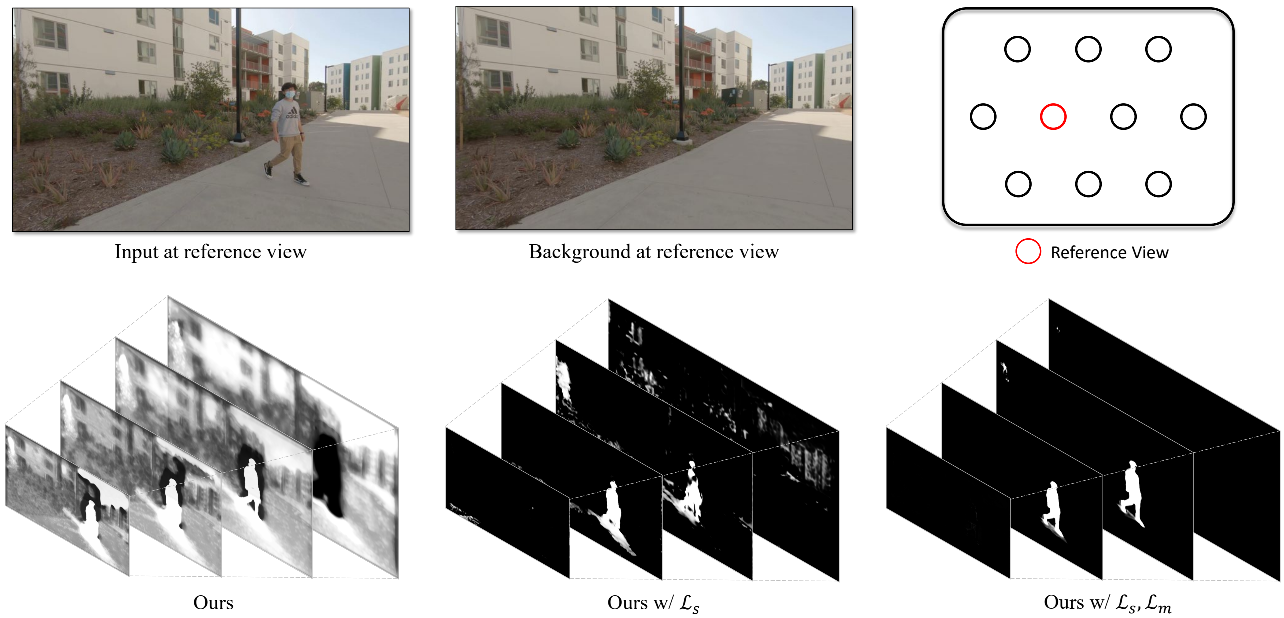


Fig. 8. 3D visualization of the masks from different loss functions. With alpha values from the instantaneous MPI, we collapse the mask volumes using over composite to reduce plane count from 32 to 4 for better visualization. (e.g. plane 1∼8 to the furthest plane, ..., plane 25∼32 to the nearest plane.) Note that there is no supervision on static parts in our final loss function, so the values in those parts are unconstrained, resulting in soft blending between instantaneous frames and the background. In general, the 3D mask achieves better temporal consistency by replacing the erroneous disoccluded parts with correct background observations.

TABLE 5
Comparison on the synthetic evaluation dataset.

Methods	Mask	STRRED↓	PSNR↑	SSIM↑
MPI/LLFF [1]	✗	0.2889	26.12	0.8345
2D Mask	2D	0.5428	24.01	0.8082
IBRNet (2-view) [49]	✗	1.7984	21.37	0.6942
NeRFlow [6]	✗	1.8306	19.99	0.5996
NSFF [7]	✗	1.0627	19.72	0.5577
Non-rigid NeRF [8]	✗	3.1401	18.92	0.5947
Ours	3D	**0.2812**	**26.13**	**0.8342**

to our evaluation on the real dataset, we chose cameras 4 and 5 as input. In Table 5, we show the numbers of various methods. The proposed method achieves favorable results compared to other baselines. Additionally, we show qualitative results in Fig. 7. For MPI/LLFF, the numbers are slightly worse than our proposed algorithm, because the main difference is in the disoccluded regions. It can be seen in the row (a) around the moving characters. 2D mask introduces more artifacts and thus results in worse numbers across all metrics. In row (b), 2D mask exacerbates the artifacts and creates more visible repeated texture in the disoccluded regions. NeRF-based methods perform slightly better on the synthetic dataset, as the camera parameters are more precise. However, they still fail to produce sharp imagery. For example, NeRFlow lacks the details on the leftmost character in the third column in row (c). Furthermore, the second column in row (d) shows blurriness and ghosting artifacts for Non-rigid NeRF. NSFF (e) has issues rendering complex static scene texture in the last column. The table to the left shows distorted edges compared to our proposed method. IBRNet (f) still generates renderings with heavy distortions, even though the coarse geometry seemingly matches the ground truth. Our method (g) provides the best visual result and it is able to generalize to unseen synthetic scenes when trained on real data. Please refer to the supplementary video for more results.

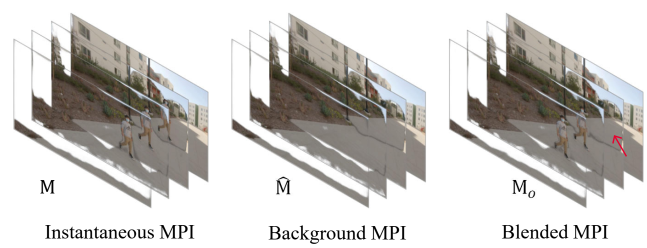


Fig. 9. 3D visualization of the MPI volumes using our loss function $\mathcal{L}$. Note that the person on the furthest plane in $\mathbf{M}$ is replaced by the background in $\mathbf{M}_o$.

5.4 Ablation Studies on Loss Function

In this sub-section, we experiment with different losses to see if we can acquire a 3D mask volume that is more interpretable and possesses physical meaning. Two additional loss functions are described as follows. The first loss is a mask supervision loss $\mathcal{L}_m$, which forces the mask volume to match the shape of the dynamic object in the scene. The second loss is a sparsity loss $\mathcal{L}_s$ applied on the mask volume to encourage the network to reuse $\hat{\mathbf{M}}$ more. To be more specific, for the mask loss, we use the work by Lin et al. [46], which takes the individual frame $\mathbf{I}$ and the background $\hat{\mathbf{I}}$ in the video to generate a dynamic object mask $\mathbf{V}_{gt}$ we later use as supervision. To supervise the mask volume, we directly regularize the over-composited alphas from the warped foreground MPI volume $\mathcal{W}(\mathbf{M} \odot \mathbf{V})$ to be consistent with $\mathbf{V}_{gt}$. We denote the over-composited alpha values as m_1. This mask loss is similar to the mask supervision loss in Lu et al. [55] We calculate the

Fig. 10. Our algorithm is able to provide better visual quality than baseline methods even when the novel viewpoint is far away from the input view. We show results when the baseline is $2.5\times$ and $5\times$ baseline between input views. Note that in the $5\times$ case, our method produces fewer artifacts compared to Mildenhall et al. [1], offering a more graceful degradation.

estimated background mask m_0 by dilating the foreground mask with a kernel of size $(5,5)$ to produce m_1'. The background mask is then $m_0 = 1 - m_1'$. And the mask supervision loss is:

$$\mathcal{L}_m = \frac{||m_1 \odot (1-\mathbf{V}_{gt})||_1}{2||m_1||_1} + \frac{||m_0 \odot \mathbf{V}_{gt}||_1}{2||m_0||_1}. \quad (12)$$

Another loss is a L_1 sparsity constraint on the mask volume to ensure it only covers the necessary portions,

$$\mathcal{L}_s = ||\sum_{(x,y,d)} \mathbf{V}(x,y,d)||_1. \quad (13)$$

We use $\mathcal{L} + 0.1\mathcal{L}_s + 0.25\mathcal{L}_m$ for the full combination and $\mathcal{L} + 0.1\mathcal{L}_s$ for the additional sparsity constraint.

TABLE 6
Effect of different loss functions. Our rendering loss offers better temporal consistency and slightly better visual quality.

$\mathcal{L}_s$	$\mathcal{L}_m$	**STRRED↓**	**PSNR↑**	**SSIM↑**
-	-	**0.1683**	**26.22**	0.8390
✓	-	0.1745	26.18	**0.8393**
✓	✓	0.1900	26.09	0.8374

As shown in Table 6, our rendering loss still offers the most temporally-stable results, whereas the other two losses trade temporal consistency for better interpretability. It is reasonable that the mask supervision loss helps the network to give a sparser and tighter prediction on the dynamic objects. However, it does not take into account the movements of the foliage and the shadows, producing slightly unstable results in those areas. The sparsity constraint is able to achieve marginally better quality than the full $\mathcal{L}_s, \mathcal{L}_m$ combination as it retains some parts of the scene which might cover the slight differences between frames.

Mask visualization can be found in Fig. 8. From the figure, we can observe that our mask volume removes areas around the edges of the dynamic object and the occluded areas behind it. Moreover, the mask softly blends the shadows cast by the moving object. Adding $\mathcal{L}_s$, the mask becomes sparser, ignoring most static areas. However, as shown in Fig. 8, it still contains some areas around the plants on the left and the building in the back. With $\mathcal{L}_s, \mathcal{L}_m$, the mask has more physical meaning and the resulting 3D mask only covers the dynamic object. This might be useful to extract moving objects for other uses such as editing or object insertion.

We further examine the 3D visualization of $\mathbf{M}, \hat{\mathbf{M}}$, and $\mathbf{M}_o$ in Fig. 9. Note that in the blended MPI $\mathbf{M}_o$, the occluded area behind the person is filled with actual background information, unlike in $\mathbf{M}$, which has repeated texture of the dynamic object. Since we do not enforce any constraints on the static parts of the scene, our mask has random values in these areas and softly blends them with the background MPI. *This does not affect temporal consistency too much as the difference is minor and some areas are free space which does not contribute any color to the MPI volume as shown in Fig. 9.*

5.5 Large distance view extrapolation

In Fig. 10, we show results when the target camera is translated far more than the baseline of the input camera pair. When large translational movement is introduced, the conventional method [1] starts to show artifacts in the disoccluded regions. On the contrary, our method still preserves the background details even when the motion is larger, offering a more graceful reduction in quality as the distance is increased.

5.6 Extension to more input views

Although our proposed method primarily targets binocular view extrapolation, we also demonstrate that it can be extended to utilize more input views in Fig. 11 and in the supplementary video. With more input views, it can acquire better scene geometry for some cases where there are ambiguities in the plane sweep

Fig. 11. Our proposed method can also be extended to take 4-view input. We feed 4 input views to both the MPI and mask networks to acquire our result. Here the baseline method is also adjusted to use 4 input views instead of 2. Notice that the artifacts around the person do not appear in our result.

Fig. 12. From top to bottom, we show frame 0, frame 966 (last frame) and the extracted background. Since the lighting changes drastically during this scene, the extracted background contains a lot of ghosting artifacts.

volume. For example, some ambiguities might occur when there is straight texture-less structure (beams or handrails) parallel to the camera baseline. Using additional cameras can provide more geometric information and avoid similar situations. In Fig. 11, the main difference is that we modify our network to take 4 input views, which convert to 4 instantaneous images and 4 background images as input to the mask network, and output the 3D mask volume as in the pipeline shown in Fig. 4.

5.7 Limitations

The proposed dataset and algorithm have a few limitations: First, we limit our camera to stay static when capturing. This is mainly due to the limitations of synchronization and pose estimation. Although we can achieve good synchronization with software-based methods, there are still a few milliseconds of error. This error could be magnified when the camera rig is in motion and lead to bad estimates of the camera poses. The camera poses across time would also require more calculations, possibly leading to accumulating errors in the system. These issues could be solved by calibrating the camera trajectory of one of the cameras and utilizing the rigid assumption to infer the trajectories of other cameras. Another limitation is that we require an estimate of the static background. This is easily achievable by applying a median filter. While it works for most of the scenes, this method is sometimes not reliable. We show one example in Fig. 12. In this particular case, the sun light appears after a while in the video, casting hard shadows on the walls. As a result, the background is difficult to determine. Another possible case happens when a static object is moved during the video. It is ambiguous to define the exact background for this case as both states might take up a large portion of the video. Thus, it might require more careful division of different states or using a lighting-agnostic method. There are more advanced approaches [56], [57] that can be used in the future.

6 Conclusions and Future Work

In this paper, we discuss view synthesis of dynamic scenes with stereo input videos. The main challenge is that rendered results are prone to temporal artifacts like flickering in the disoccluded regions. To tackle this issue, we introduce a novel 3D mask volume extension to carefully replace the disoccluded areas with background information acquired from the temporal frames. Additionally, we introduce a high-quality multi-view video dataset, which contains 96 scenes of various human interactions and outdoor environments shot in 120FPS.

In future work, we would like to extend our dataset and method to consider dynamic camera motions, and to operate on even larger baselines. In summary, we believe video view synthesis for dynamic scenes is the next frontier for immersive applications, and this paper has taken a key step in that direction.

7 Acknowledgement

This work was supported in part by a Qualcomm FMA Fellowship, ONR grants N000142012529, N000142312526, NSF grants 2100237 and 2120019, all awarded to the UCSD researchers. We also acknowledge gifts from Adobe, Google, an Amazon Research Award, a Facebook Distinguished Faculty Award, a Sony Research Award, the Ronald L. Graham Chair, and the UC San Diego Center for Visual Computing. Part of the work was done when KEL was an intern at Facebook. Lastly, we thank Jiyang Yu, Yuzhe Qin, Dominique Meyer, Eric Lo, Thomas DeFanti, Jürgen Schulze and Michael Broxton for comments on hardware setup.

References

[1] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (TOG)*, 2019.
[2] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," in *SIGGRAPH*, 2018.
[3] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely, "Pushing the boundaries of view extrapolation with multiplane images," *CVPR*, 2019.
[4] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang, "3d photography using context-aware layered depth inpainting," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[5] K.-E. Lin, L. Xiao, F. Liu, G. Yang, and R. Ramamoorthi, "Deep 3d mask volume for view synthesis of dynamic scenes," in *ICCV*, 2021.
[6] Y. Du, Y. Zhang, H.-X. Yu, J. B. Tenenbaum, and J. Wu, "Neural radiance flow for 4d view synthesis and video processing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[7] Z. Li, S. Niklaus, N. Snavely, and O. Wang, "Neural scene flow fields for space-time view synthesis of dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[8] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt, "Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021.

[9] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker, "Deepview: View synthesis with learned gradient descent," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2367–2376.

[10] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. DuVall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec, "Immersive light field video with a layered mesh representation," vol. 39, no. 4, pp. 86:1–86:15, 2020.

[11] J. S. Yoon, K. Kim, O. Gallo, H. S. Park, and J. Kautz, "Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5336–5345.

[12] Q. Li and N. Khademi Kalantari, "Synthesizing light field from a single image with variable mpi and two network fusion," *ACM Transactions on Graphics*, vol. 39, no. 6, 12 2020.

[13] A. Bansal, M. Vo, Y. Sheikh, D. Ramanan, and S. Narasimhan, "4d visualization of dynamic events from unconstrained multi-view videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5366–5375.

[14] Z. Li, T. Dekel, F. Cole, R. Tucker, N. Snavely, C. Liu, and W. T. Freeman, "Learning the depths of moving people by watching frozen people," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4521–4530.

[15] M. Bemana, K. Myszkowski, H.-P. Seidel, and T. Ritschel, "X-fields: Implicit neural view-, light- and time-image interpolation," *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2020)*, vol. 39, no. 6, 2020.

[16] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 43–54.

[18] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 31–42.

[19] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 425–432.

[20] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 11–20.

[21] J. Shade, S. Gortler, L.-w. He, and R. Szeliski, "Layered depth images," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998, pp. 231–242.

[22] A. Davis, M. Levoy, and F. Durand, "Unstructured light fields," in *Computer Graphics Forum*, vol. 31, no. 2pt1. Wiley Online Library, 2012, pp. 305–314.

[23] E. Penner and L. Zhang, "Soft 3d reconstruction for view synthesis," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–11, 2017.

[24] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–15, 2018.

[25] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "Deepstereo: Learning to predict new views from the world's imagery," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5515–5524.

[26] N. K. Kalantari, T.-C. Wang, and R. Ramamoorthi, "Learning-based view synthesis for light field cameras," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 1–10, 2016.

[27] Z. Xu, S. Bi, K. Sunkavalli, S. Hadap, H. Su, and R. Ramamoorthi, "Deep view synthesis from sparse photometric images," *ACM Trans. Graph.*, vol. 38, no. 4, Jul. 2019. [Online]. Available: https://doi.org/10.1145/3306346.3323007

[28] S. Niklaus, L. Mai, J. Yang, and F. Liu, "3d ken burns effect from a single image," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–15, 2019.

[29] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision*. Springer, 2020, pp. 405–421.

[30] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser *et al.*, "Local implicit grid representations for 3d scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6001–6010.

[31] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural volumes: Learning dynamic renderable volumes from images," *arXiv preprint arXiv:1906.07751*, 2019.

[32] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer, "Deepvoxels: Learning persistent 3d feature embeddings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2437–2446.

[33] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.

[34] K.-E. Lin, Z. Xu, B. Mildenhall, P. P. Srinivasan, Y. Hold-Geoffroy, S. DiVerdi, Q. Sun, K. Sunkavalli, and R. Ramamoorthi, "Deep multi depth panoramas for view synthesis," in *ECCV*, 2020.

[35] R. Szeliski and P. Golland, "Stereo matching with transparency and matting," in *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE, 1998, pp. 517–524.

[36] Z. Xu, K. Sunkavalli, S. Hadap, and R. Ramamoorthi, "Deep image-based relighting from optimal sparse samples," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018. [Online]. Available: https://doi.org/10.1145/3197517.3201313

[37] S. Bi, Z. Xu, K. Sunkavalli, M. Hašan, Y. Hold-Geoffroy, D. Kriegman, and R. Ramamoorthi, "Deep reflectance volumes: Relightable reconstructions from multi-view photometric images," *arXiv preprint arXiv:2007.09892*, 2020.

[38] S. Bi, Z. Xu, P. Srinivasan, B. Mildenhall, K. Sunkavalli, M. Hašan, Y. Hold-Geoffroy, D. Kriegman, and R. Ramamoorthi, "Neural reflectance fields for appearance acquisition," *arXiv preprint arXiv:2008.03824*, 2020.

[39] M. Meshry, D. B. Goldman, S. Khamis, H. Hoppe, R. Pandey, N. Snavely, and R. Martin-Brualla, "Neural rerendering in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6878–6887.

[40] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 600–608, 2004.

[41] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[42] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[43] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[44] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[45] R. Szeliski and P. Golland, "Stereo matching with transparency and matting," *International Journal of Computer Vision*, vol. 32, no. 1, pp. 45–61, 1999.

[46] S. Lin, A. Ryabtsev, S. Sengupta, B. Curless, S. Seitz, and I. Kemelmacher-Shlizerman, "Real-time high-resolution background matting," *arXiv*, pp. arXiv–2012, 2020.

[47] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[48] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1511–1520.

[49] Q. Wang, Z. Wang, K. Genova, P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser, "Ibrnet: Learning multi-view image-based rendering," in *CVPR*, 2021.

[50] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[51] R. Soundararajan and A. C. Bovik, "Video quality assessment by reduced reference spatio-temporal entropic differencing," *IEEE Transactions on*

Circuits and Systems for Video Technology, vol. 23, no. 4, pp. 684–694, 2012.

[52] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. M. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, M. Savva, Y. Zhao, and D. Batra, "Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=-v4OuqNs5P

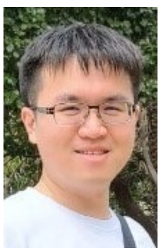
[53] E. Games, "Unreal engine sun temple, open research content archive (orca)," October 2017. [Online]. Available: http://developer.nvidia.com/orca/epic-games-sun-temple

[54] Blender Online Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Blender Institute, Amsterdam, 2020. [Online]. Available: http://www.blender.org

[55] E. Lu, F. Cole, T. Dekel, W. Xie, A. Zisserman, D. Salesin, W. T. Freeman, and M. Rubinstein, "Layered neural rendering for retiming people in video," *ACM Trans. Graph.*, vol. 39, no. 6, Nov. 2020. [Online]. Available: https://doi.org/10.1145/3414685.3417760

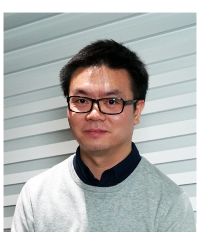
[56] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 1568–1575.

[57] S. Hauberg, A. Feragen, and M. J. Black, "Grassmann averages for scalable robust pca," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3810–3817.



Kai-En Lin received the B.Sc. degree in Electrical Engineering from National Taiwan University, Taiwan. He is currently a Ph.D candidate in the department of Computer Science, University of California, San Diego, United States. His research interests focus on view synthesis, specifically 3D representations from single or multiple images.

Guowei Yang obtained his bachelor's degree in EECS at UC Berkeley in 2019. He then did a 7 month internship at Apple, focusing on visual computing projects. From 2020 to 2022, he studied for his Master's degree at UC San Diego. He was also working as a graduate student researcher in prof. Ravi Ramamoorthi's lab, focusing on view synthesis problems. His research interests are computer vision and graphics. Now he is a research engineer at Apple on the Camera Algorithm team.



Lei Xiao is a Research Scientist at Reality Labs Research, Meta. His recent research focuses on neural rendering and its applications to virtual reality, mixed reality, and computational photography. Before joining Meta, he received his Ph.D. degree in Computer Science from the University of British Columbia.

Feng Liu is an Associate Professor in the Department of Computer Science at Portland State University. His research interests are in the areas of computer vision, computer graphics, and multimedia. He earned his M.S. and Ph.D. in computer science from the University of Wisconsin, Madison in 2006 and 2010, respectively. He received his B.E. and M.E. in computer science from Zhejiang University in 2001 and 2004, respectively. He was a visiting researcher at Google from 2017-2018 and at Facebook from 2020-2021.

Ravi Ramamoorthi received the BS degree in engineering and applied science and MS degrees in computer science and physics from the California Institute of Technology, in 1998, and the PhD degree in computer science from the Stanford University Computer Graphics Laboratory, in 2002, upon which he joined the Columbia University Computer Science Department. He was on the UC Berkeley EECS faculty from 2009-2014. Since July 2014, he is a professor of Computer Science and Engineering at the University of California, San Diego, where he holds the Ronald L. Graham Chair of Computer Science. He is also the founding director of the UC San Diego Center for Visual Computing. His research interests cover many areas of computer vision and graphics, with more than 200 publications. His research has been recognized with a number of awards, including the 2007 ACM SIGGRAPH Significant New Researcher Award in computer graphics, and by the white house with a Presidential Early Career Award for Scientists and Engineers in 2008 for his work on physics-based computer vision. Most recently, he was named an IEEE and ACM Fellow in 2017, and inducted into the SIGGRAPH Academy in 2019. He has advised more than 20 Postdoctoral, PhD and MS students, many of whom have gone on to leading positions in industry and academia; and he has taught the first open online course in computer graphics on the edX platform in fall 2012, with more than 100,000 students enrolled in that and subsequent iterations. He was a finalist for the inaugural 2016 edX Prize for exceptional contributions in online teaching and learning, and again in 2017.