Neural Volumes: Learning Dynamic Renderable Volumes from Images

STEPHEN LOMBARDI, Facebook Reality Labs
TOMAS SIMON, Facebook Reality Labs
JASON SARAGIH, Facebook Reality Labs
GABRIEL SCHWARTZ, Facebook Reality Labs
ANDREAS LEHRMANN, Facebook Reality Labs
YASER SHEIKH, Facebook Reality Labs

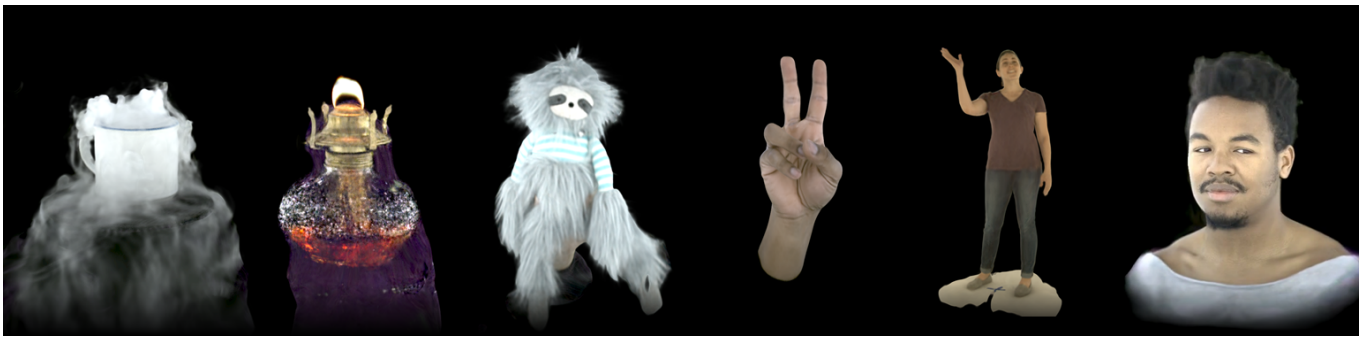


Fig. 1. Renderings of objects captured and modeled by our system. The input to our method consists of synchronized and calibrated multi-view video. We build a dynamic, volumetric representation of the scene by training an encoder-decoder network end-to-end using a differentiable ray marching algorithm.

Modeling and rendering of dynamic scenes is challenging, as natural scenes often contain complex phenomena such as thin structures, evolving topology, translucency, scattering, occlusion, and biological motion. Mesh-based reconstruction and tracking often fail in these cases, and other approaches (e.g., light field video) typically rely on constrained viewing conditions, which limit interactivity. We circumvent these difficulties by presenting a learning-based approach to representing dynamic objects inspired by the integral projection model used in tomographic imaging. The approach is supervised directly from 2D images in a multi-view capture setting and does not require explicit reconstruction or tracking of the object. Our method has two primary components: an encoder-decoder network that transforms input images into a 3D volume representation, and a differentiable ray-marching operation that enables end-to-end training. By virtue of its 3D representation, our construction extrapolates better to novel viewpoints compared to screen-space rendering techniques. The encoder-decoder architecture learns a latent representation of a dynamic scene that enables us to produce novel content sequences not seen during training. To overcome memory limitations of voxel-based representations, we learn a dynamic irregular grid structure implemented with a warp field during ray-marching. This structure greatly improves the apparent resolution and reduces grid-like artifacts and jagged motion. Finally, we demonstrate how to incorporate surface-based representations into our volumetric-learning framework for applications where the highest resolution is required, using facial performance capture as a case in point.

CCS Concepts: • **Computing methodologies** → **Neural networks**; **Rendering**; **Volumetric models**.

Additional Key Words and Phrases: Volumetric Rendering, Volume Warping, Ray Potentials, Differentiable Ray Marching

ACM Reference Format:
Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Trans. Graph.* 38, 4, Article 65 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323020

1 INTRODUCTION

Polygon meshes are an extremely popular representation for 3D geometry in photo-realistic scenes. Mesh-based representations efficiently model solid surfaces and can be paired with sophisticated reflectance functions to generate compelling renderings of natural scenes. In addition, there has been significant progress recently in optimization techniques to support real-time ray-tracing, allowing for interactivity and immersion in demanding applications such as Virtual Reality (VR). However, little of the interactive photo-real content available today is data-driven because many real-world phenomena are challenging to reconstruct and track with high fidelity. State-of-the-art motion capture systems struggle to handle complex occlusions (e.g., running hands through one's hair), to account for reflectance variability (e.g., specularities in the sheen of a moving object), or to track topological evolution in dynamic participating media (e.g., smoke as it billows upward). Where solutions exist,

Authors' addresses: Stephen Lombardi, Facebook Reality Labs, stephen.lombardi@fb.com; Tomas Simon, Facebook Reality Labs, tomas.simon@fb.com; Jason Saragih, Facebook Reality Labs, jason.saragih@fb.com; Gabriel Schwartz, Facebook Reality Labs, gabe.schwartz@oculus.com; Andreas Lehrmann, Facebook Reality Labs, asml@fb.com; Yaser Sheikh, Facebook Reality Labs, yasers@fb.com.

they are typically specialized to an individual phenomenon [Atcheson et al. 2008; Hawkins et al. 2005; Roth and Black 2006; Xu et al. 2014], and are often aimed at either image generation [Buehler et al. 2001; Kalantari et al. 2016] or 3D reconstruction [Goesele et al. 2007; Nießner et al. 2013], but not both. Since mesh-based representations rely heavily on the quality of reconstruction to produce compelling renderings, they are ill-suited to handle such cases. Nonetheless, these kinds of phenomena are necessary to create compelling renderings of much of our natural world.

To address limitations posed by inaccurate geometric reconstructions, great progress has been made in recent years by relaxing the physics-inspired representation of light transport, and instead leveraging machine learning to bridge the gap between the representation and observed images of the scene. In Lombardi et al. [2018], this technique was used to great effect in modeling the human face, where it was demonstrated that a neural network can be trained to compensate for geometric reconstruction and tracking inaccuracies through a view-dependent texture. Similar approaches have also been shown to be effective for modeling general far-field scenes [Overbeck et al. 2018]. An extreme variant of this technique is screen-space rendering, where no geometry of the scene is used at all [Karras et al. 2018; Wang et al. 2018]. Although these approaches have been shown to produce high quality renderings of complex scenes, they are limited to viewpoints available to the system at training time. Since their neural architectures are not *3D aware*, the methods do not extrapolate to novel viewpoints in a way that is consistent with the real world. The problem is exacerbated when modeling near-field scenes, where variation in viewpoint is more common as a user interacts with objects in the scene, compared with far-field captures where there is less interactivity and the viewer is mainly stationary.

An important insight in this work is that if *both* geometry and appearance variations can be learned simultaneously, phenomena explainable by geometric variations may be modeled as such, leading to better generalization across viewpoints. The challenge, then, is to formulate this joint learning such that good solutions can be found. Directly optimizing over a mesh-representation using gradient-based optimization is prone to terminating in poor local minima. This is the case even when a model of both appearance and geometry are known *a priori*, and is exacerbated when these models are also unknown. One of the main reasons for this difficulty is the local support of the gradients of mesh-based representations. To address this, we propose using a volumetric representation consisting of opacity and color at each position in 3D space, where rendering is realized through integral projection. During optimization, this semi-transparent representation of geometry disperses gradient information along the ray of integration, effectively widening the basin of convergence, enabling the discovery of good solutions.

Although the volumetric representation has the ability to represent 3D geometric phenomena in a geometrically faithful way, it can easily over-fit from image-supervision for a typical density of viewpoints. As such, additional regularization is necessary to achieve good results. In this work, we show that a neural-network decoder is sufficient to encourage discovery of solutions that generalize across viewpoints. At first glance, this may appear surprising given the decoder network typically has enough capacity to reproduce solutions found through a direct solve of the volume's entries. We conjecture that the decoder network introduces spatial regularity into the gradients of the volume's entries (i.e., opacity and color), leading to more generalizable solutions without diminishing the volumetric representation's capacity. Additionally, the decoder network is paired with an encoder network that produces a low-dimensional latent space that encodes the state of the scene at each frame, enabling joint reconstruction of sequences rather than just individual frames. Analogous to non-rigid structure from motion [Torresani et al. 2008], this architecture can leverage a scene's regularity across time to improve viewpoint generalization further. This latent code can be used to generate novel renderings of the scene's content by traversing the latent space, enabling realistic modifications of the recording, or even completely new sequence animation, without requiring object/scene/content specific solutions.

Despite these advantages of the volumetric representation, its main drawback is limited resolution. Using voxel-based data structures to represent a scene typically requires an order of magnitude more memory than its mesh-based counterpart to achieve similar levels of resolution. Furthermore, much of this memory is dedicated to modeling empty space or the inside of objects; neither of which have an impact on the rendered result. This limitation stems from the regular grid structure this representation exhibits. To overcome this limitation, we employ a warping technique that indirectly escapes the restrictions imposed by a regular grid structure, allowing the learning algorithm to make the best use of available memory. Using this technique, we demonstrate significantly higher fidelity than using only a conventional voxel data structure. Furthermore, as our representation is 3D based, we can naturally combine it with surface-based reconstruction and tracking methods when appropriate. This allows us to reach the highest levels of fidelity on objects in the scene for which state-of-the-art reconstruction and tracking work well, while also maintaining a complete model of the scene.

In summary, we propose a novel volumetric representation that is object/scene agnostic, can generalize well to novel viewpoints, reconstructs dynamic scenes jointly, facilitates novel content generation, requires only image-level supervision and is end-to-end trainable. The resulting models afford real-time rendering and support on-the-fly adjustments, suitable for interactive applications in VR. In §2 we cover related work, followed by an overview of our approach in §3. Details of the encoder and decoder architectures are covered in sections §4 and §5 respectively. Rendering through integral projection is discussed in §6 and details of the learning problem are covered in §7. We evaluate this architecture on a number of challenging scenes and present ablation study on various design choices in our construction in §8. We conclude in §9 with a discussion and directions of future work.

2 RELATED WORK

Our approach is driven by learning and rendering techniques spanning multiple domains, from volumetric reconstruction and deformable volumes to neural rendering and novel view synthesis. The following paragraphs discuss similarities and differences to previous works in these areas.

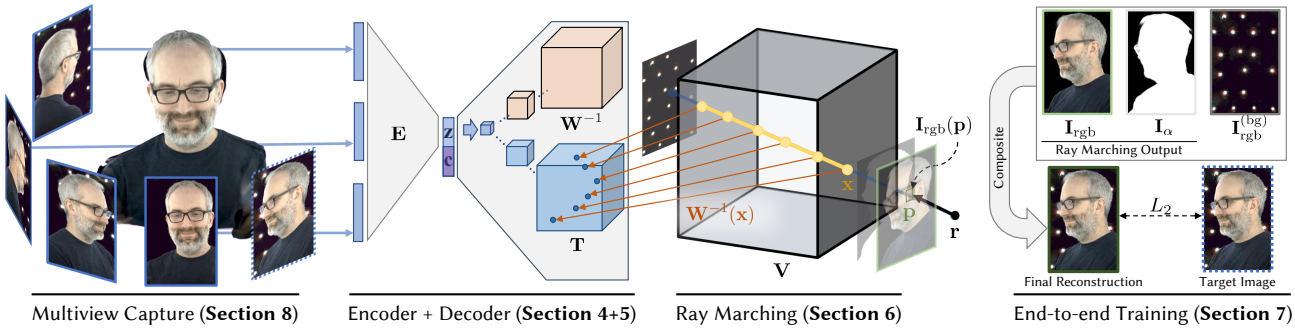


Fig. 2. Pipeline of our method. We begin with a multi-view capture system, from which we choose a subset of the cameras as input to our encoder. The encoder produces a latent code $\mathbf{z}$ which is decoded into a volume that gives an RGB and α value for each point in space, as well as a warp field used to index into the RGBα volume. The decoder may optionally use an additional control signal (e.g., head pose) $\mathbf{c}$. We then use an accumulative ray marching algorithm to render the volume. The final output in image space is an RGB image with associated alpha mask and estimated background. We composite these components together and minimize the L_2 loss between the rendered and target images w.r.t. network parameters.

2.1 Classical Surface and Volumetric Reconstruction

Point- and surface-based reconstruction techniques have a long history in computer vision (see [Furukawa and Hernández 2015] for a review). Following successful efforts in stereo matching [Scharstein and Szeliski 2002], most of the subsequent literature has focused on the multi-view case, including extensions of photometric consistency [Furukawa and Ponce 2010] and depth map fusion [Merrell et al. 2007; Zach et al. 2007]. Despite some attempts at handling more complex materials or semi-transparent surfaces [Fitzgibbon and Zisserman 2005; Szeliski and Golland 1999], many popular multi-view stereo (MVS) methods, such as COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016], still struggle with thin structures and dense semi-transparent materials (e.g., hair and smoke).

Volumetric reconstruction methods side-step this problem of explicit correspondence matching, beginning with Voxel Coloring [Prock and Dyer 1998; Seitz and Dyer 1997, 1999] and Space Carving [Bonet and Viola 1999; Broadhurst et al. 2001; Kutulakos and Seitz 2000]. These methods recover occupancy and color in a voxel grid from multi-view images by evaluating the photo-consistency of each voxel in a particular order. Our method is similar to these classical voxel-based techniques in spirit, but rather than using a strict photometric consistency criterion, we learn a generative model that tries to best match the input images. Since we do not assume that objects in a scene are composed of flat surfaces, this approach also allows us to overcome the typical limitations of MVS methods and capture rich materials and fine geometry.

2.2 Volumetric Reconstruction with Ray Potentials

A number of more recent works on volumetric reconstruction have explored the concept of *ray potentials*, i.e., cost functions between the first surface struck by a ray and the color (or other property) of the corresponding pixel. Ulusoy et al. [2015], Savinov et al. [2016], and Paschalidou et al. [2018] formulate graph-based energy or inference objectives using ray potentials as constraints. A differentiable ray consistency criterion that inspired our work is developed by Tulsiani et al. [2018, 2017], who use an encoder-decoder architecture to predict voxel occupancy probabilities from RGB images. A loss on ray potentials evaluated in this volume is then backpropagated to the underlying convolutional architecture.

A fundamental difference between our work and previous works using ray potentials is that we model voxel transparency rather than occupancy probability, as we are focused on rendering rather than reconstruction. This explicit image formation process allows us to reconstruct and render dynamic scenes of semi-transparent materials, such as smoke.

2.3 Deformable Volumes

Non-rigidly deforming objects pose challenges to both optimization- and learning-based approaches. Over the years, significant efforts have been spent on their reconstruction and tracking from RGB-D sensors: the DynamicFusion method of Newcombe et al. [2015] produces a base 3D template surface using a Truncated Signed Distance Function (TSDF) representation, and a time-dependent warp to fuse a sequence of depth frames. Zollhöfer et al. [2014] and Innmann et al. [2016] also deform a 3D template surface with an as-rigid-as-possible regularizer.

Our rendering model is based on classical volume ray marching [Ikits et al. 2004; Levoy 1988], but we introduce the concept of a *warp field* to it: instead of directly sampling color and opacity along the ray, we first sample a 3D warp field encoding the location (within a template voxel grid) from which color and opacity are sampled. In 2D, similar techniques have been used to learn warps that align images [Dai et al. 2017; Jaderberg et al. 2015; Shu et al. 2018]. In our work, the warp field is not only used to model motion but also increases the effective resolution of the voxel grid by simulating a dynamic non-uniform sampling grid.

2.4 Neural Rendering

Deep learning-based rendering has become an active area of exploration, with some methods relying on volumetric representations. Nguyen-Phuoc et al. [2018] use a convolutional neural network applied on a voxel grid to produce an image, but their method requires correspondence between the images and the voxel reconstruction.

DeepVoxels [Sitzmann et al. 2018] automatically learns a 3D feature representation for novel view synthesis but is limited to static objects and scenes, and the employed architecture does not lend itself to real-time inference. Martin-Brualla et al. [2018] use neural networks to fill holes and generally improve the quality of a textured geometric representation. Kim et al. [2018] use a U-net architecture to convert an image of rasterized attributes to realistic images, similar to pix-to-pix [Isola et al. 2017]. Our method shares many similarities with these approaches but has one important difference: machine learning is only used to generate an RGBA volume, which is then rendered with a ray marching algorithm with no learned parameters. This is important as it gives us an interpretable volume, which may lead to better viewpoint generalization.

In our evaluation of hybrid rendering approaches, we employ the Deep Appearance Model of Lombardi et al. [2018] that provides a textured mesh representation. The method learns a Variational Autoencoder (VAE) model representing the dynamic mesh and view-dependent texture of a specific person's face. While we also evaluate our model on human faces, our method does not require precise mesh tracking or other forms of pre-processing. Instead, it is trained end-to-end, using only raw images as supervision. Volumetric representations such as ours are also able to better represent complex surfaces like hair that are difficult to model using meshes.

2.5 Novel View Synthesis

Novel view synthesis aims to produce RGB images of novel views given a set of input RGB images. Typically, these methods use a geometric proxy to assist in reprojecting 3D points back into the input images and some blending is performed to produce a final pixel color. Buehler et al. [2001] and Davis et al. [2012] use heuristics to blend contributions of different images based on the rays from the geometric proxy to each camera. Hedman et al. [2018] uses neural networks to determine blending weights, which can overcome inaccurate geometric proxies. Zhou et al. [2018] skip the geometric proxy altogether and use a neural network to compute blending weights of each image projected along a set of planes. Penner and Zhang [2017] compute a soft volumetric representation using MVS depth maps to perform novel view synthesis. Unlike most novel view synthesis techniques, our method operates on sequences and creates an animatable model.

To compute geometric proxies, a multi-view stereo method is often employed. While free-viewpoint video methods [Collet et al. 2015; Prada et al. 2016] rely on a sophisticated combination of multi-view stereo techniques (including silhouettes and MVS) to reconstruct many kinds of objects, our method creates novel views of dynamic scenes with a single generative framework. In addition, our model's latent embedding of the scene allows us to generate novel animations more flexibly by producing new embedding sequences.

3 OVERVIEW

We present an end-to-end pipeline for rendering images from novel views with only image supervision that leverages an internal 3D volumetric representation. There are two main parts to the method: an encoder-decoder network that converts input images into a 3D volume $\mathbf{V}(\mathbf{x})$, and a differentiable raymarching step that renders an image from the volume $\mathbf{V}$ given a set of camera parameters. The method can be thought of as an autoencoder whose final layer is a fixed-function (i.e., no free parameters) volume rendering operation.

Formally, we model a volume that maps 3D positions, $\mathbf{x} \in \mathbb{R}^3$, to a local RGB color and differential opacity at that point,

$$\mathbf{V} : \mathbb{R}^3 \rightarrow \mathbb{R}^4, \qquad \mathbf{V}(\mathbf{x}) = \left(\mathbf{V}_{\text{rgb}}(\mathbf{x}), \mathbf{V}_{\alpha}(\mathbf{x})\right), \tag{1}$$

where $\mathbf{V}_{\text{rgb}}(\mathbf{x}) \in \mathbb{R}^3$ is the color at $\mathbf{x}$ and $\mathbf{V}_{\alpha}(\mathbf{x}) \in \mathbb{R}$ is its differential opacity in the range $[0, \infty]$, with 0 representing full transparency. The purpose of a semi-transparent volume is two-fold: first, it is a softening of a discrete volume representation which enables gradients to flow for learning; second, it allows us to model semi-transparent objects or bundles of thin structures that appear translucent at limited resolutions, like hair.

Fig. 2 shows a visual representation of the pipeline. We first capture a set of synchronized and calibrated video streams of a performance from different viewpoints. Next, an encoder network takes images from a subset of the cameras for each time instant and produces a latent code $\mathbf{z}$ that represents the state of the scene at that time. A volume decoder network produces a 3D volume given this latent code, $\mathbf{V}(\mathbf{x}; \mathbf{z})$, which yields an RGB$\alpha$ value at each point $\mathbf{x}$. Finally, an accumulative ray marching algorithm renders the volume from a particular point-of-view. We train this system end-to-end by reconstructing each of the input images and minimizing the squared pixel reconstruction loss over the entire training set. At training time, we run through the entire pipeline to train the weights of the encoder-decoder network. At inference time, we produce a stream of latent codes $\mathbf{z}$ (either the sequence of latent codes produced by the training images or a novel generated sequence) and decode and render in real time.

4 ENCODER NETWORK

The main component of our system that enables novel sequence generation is the encoder-decoder architecture, where the scene's state is encoded using a consistent latent representation $\mathbf{z} \in \mathbb{R}^{256}$. A traversal in this latent space can be decoded into a novel sequence of volumes that can then be rendered from any viewpoint (see §5 for details). This is in contrast to methods that rely on specialized mesh constructions per frame which only allow for playback [Collet et al. 2015] or limited control over the generative process [Prada et al. 2016]. Moreover, this representation allows for conditional decoding, where only part of the scene's state is modified on playback (i.e., expression during speech, view-dependent appearance effects, etc.). The encoder-decoder architecture naturally supports this capability without requiring specialized treatment on the decoder side so long as paired samples of the conditioning variable are available during training.

To build the latent space, the information state of the scene at any given time is codified by encoding a subset of views from the multi-camera capture system using a convolutional neural network (CNN). The architecture of the encoder is shown in Fig. 3. Each camera view is passed through a dedicated branch before being concatenated with those from other views and further encoded down to the final shape. Although using all camera views as input is optimal from an information-theoretic perspective, we found that

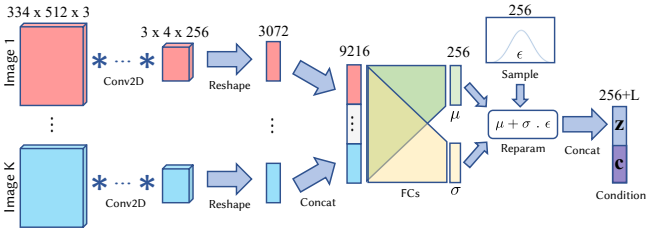


Fig. 3. Encoder architecture. Inputs are images from a subset of K-cameras, passed through camera-specific CNNs. The latent variable $\mathbf{z}$ is concatenated with the L-dimensional conditioning variable $\mathbf{c}$ before being passed to the decoder (see §5).

using K = 3 views worked well in practice, while being much more memory and computationally efficient. To maximize coverage, we select a subset of cameras that are roughly orthogonal, although our system is not especially sensitive to the specific choice of views. In practice, we used the frontal, left and right-most camera views and downsampled the images by a factor of 8 to size 334×512 pixels.

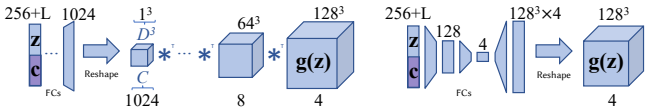
To generate plausible samples during a traversal through the latent space, the generative model needs to generalize well between training samples. This is typically achieved by learning a smooth latent space. To encourage smoothness, we use a variational architecture [Kingma and Welling 2013]. The encoder outputs parameters of a diagonal 256-dimensional Gaussian (i.e., μ and σ), whose KL-divergence from a standard Normal distribution is used as regularization. Generating an instance involves sampling from this distribution using the reparameterization trick, and decoding into the volumetric components described in §5.

In addition to encouraging latent space smoothness, the variational architecture also ensures that the decoder makes use of the conditioning variable when it is trained jointly with the encoder, as described in §7. Specifically, since the variational bottleneck maximizes the non-informative latent dimensions [Higgins et al. 2017], information pertaining to the conditioning variable is projected out of the latent space, leaving the decoder no choice but to use the conditioning variable in its reconstruction.

This method of conditioning can be applied using any auxiliary information available to the user for controlling the rendered output. In §8 we show experiments demonstrating this for a few types of conditioning information. Of particular importance is view-conditioning, which allows view-dependent effects, such as specularity, to be rendered correctly. When viewed in VR, the auxiliary information in the form of a view-vector can be obtained from the relative orientation of the headset in the virtual scene.

5 VOLUME DECODERS

In this section we discuss parameterizing the RGBα volume function $\mathbf{V}$ using different neural network architectures which we call *volume decoders*. We discuss possible representations for the volumes (voxel grids and multi-layer perceptrons) and a method for increasing the effective resolution by using warping fields. Finally, we discuss view-conditioning for modeling view-dependent appearance.



(a) Convolutional Decoder (b) Linear Basis Decoder

Fig. 4. Voxel grid decoders. (a) Convolutional and (b) Linear basis decoders. L denotes the size of the conditioning variable.

5.1 MLP Decoders

One possible model for the volume function $\mathbf{V}(\mathbf{x}; \mathbf{z})$ at point $\mathbf{x}$ with state $\mathbf{z}$, is an implicit one with a series of fully connected layers with non-linearities. A benefit of this approach is that we're not restricted by voxel grid resolution or storage space. Unfortunately, in practice an MLP requires prohibitive size to produce high-quality reconstructions. We must also evaluate the MLP at every step along each ray in the ray-marching process (see §6), imposing an equally restrictive upper bound on the MLP complexity for real-time applications.

5.2 Voxel Grid Decoders

Rather than trying to model the entire volume implicitly with an MLP, we may instead assume that the volume function can be modeled as a discrete 3D grid of voxels. We produce this explicit 3D voxel grid as the output tensor of a neural network. Let the tensor $\mathbf{Y} \in \mathbb{R}^{C \times D \times D \times D}$ represent a $D \times D \times D$ grid of values in $\mathbb{R}^C$, with C the *channel* dimensions. Define $S(\mathbf{x}; \mathbf{Y}) : \mathbb{R}^3 \rightarrow \mathbb{R}^C$ to be an interpolation function that samples from the grid $\mathbf{Y}$ by scaling continuous values in the range $[-1, 1]$ to the grid $[1, D]$ along each dimension, followed by trilinear interpolation. We can define a volume decoder for a 3D cube with center at $\mathbf{x}_o$ and sides of size W,

$$\mathbf{V}(\mathbf{x}; \mathbf{z}) = S\left(\frac{\mathbf{x} - \mathbf{x}_o}{W/2}; \mathbf{g}(\mathbf{z})\right), \tag{2}$$

where $\mathbf{g}(\mathbf{z})$ is a neural network that produces a tensor of size $4 \times D \times D \times D$. Note that we only evaluate the decoder function inside the volume it covers by computing intersections with its bounding volume.

In practice, we use either a convolutional architecture or a series of fully-connected layers to implement $\mathbf{g}(\mathbf{z})$. In the former case, we first apply a fully-connected layer and non-linearity to transform $\mathbf{z}$ into a 1024-dimensional representation and reinterpret the resulting vector as a $1 \times 1 \times 1$ cube with 1024 channels. We experiment with two convolutional architectures, one with a final size of 32^3 (achieved with 5 transposed convolutions and running at 90Hz) and one with a final size of 128^3 (achieved with 7 transposed convolutions and running at 22Hz). As an alternative, we consider a bottleneck architecture capable running at 90Hz consisting of 3 fully-connected layers with output sizes of 128, 4, and $128^3 \times 4$, respectively, with the last layer representing the decoded volume. After decoding the volume, we apply a softplus function to the RGBα values to ensure they are non-negative. Both decoder variants are illustrated in Fig. 4.

5.3 Warping Fields

On their own, voxel grids are limited because they can only represent details as small as a single voxel and are computationally

expensive to evaluate and store at high resolution. Additionally, they are wasteful in typical scenes where much of the scene consists of empty space. Common solutions to these problems are spatial acceleration structures like octrees (e.g., Riegler et al. [2017]), but it's difficult to modify these to work in a learning setting. They also typically require that the distribution of objects within the structure is known *a priori*, which is not true in our case as we do not use any 3D training data.

To solve these problems, we propose to use warping fields to both alter the effective resolution of the voxel volumes as well as model motion more naturally. In our warping formulation, we produce a template RGBα volume $\mathbf{T}(\mathbf{x})$ and a warp volume $\mathbf{W}^{-1}(\mathbf{x})$. Each point in the warp volume gives a corresponding location in the template volume from which to sample, making this an inverse warp as it maps from output positions to template sample locations. As before, both template and warp are decoded from a dynamic (per-frame) latent code $\mathbf{z}$, which we drop from the notation for conciseness.

The choice of inverse warps rather than forward warps allows representing resolution-increasing transformations by mapping a small area of voxels in the output space to a larger area in the template space without requiring additional memory. Thus, the inverse warp can represent details in the output space with higher resolution than uniform grid sampling, but remains well-defined everywhere in the output space, which is necessary for providing usable gradients during learning.

Formally, we define the inverse warp field,

$$\mathbf{W}^{-1}(\mathbf{x}) \rightarrow \mathbf{y} \qquad \mathbf{x}, \mathbf{y} \in \mathbb{R}^3, \tag{3}$$

where $\mathbf{x}$ is a 3D point in the output (rendering) space and $\mathbf{y}$ a 3D point in the RGBα template space. To generate the final volume value, we first evaluate the value of the inverse warp and then sample the template volume $\mathbf{T}$ at the warped point,

$$\mathbf{V}_{\text{RGB}\alpha}(\mathbf{x}) = \mathbf{T}_{\text{RGB}\alpha}(\mathbf{W}^{-1}(\mathbf{x})). \tag{4}$$

5.4 Mixture of Affine Warps

An important piece of including warp fields is determining how they're produced. As we show later, the architecture of the warp field decoder makes a large difference on the quality of the model.

A straightforward approach to decoding warp fields would be to use deconvolutions to produce a warp field with freely-varying template sample points at each output point. This parameterization, however, is too flexible, resulting in overfitting and poor generalization to novel views as we show in our experimental results. We instead take the approach that the basic building block of a warp field should be an affine warp. Since a single affine warp can't model non-linear bending, we use a spatial mixture of affine warps to produce an inverse warp field.

We write the affine mixture as,

$$\mathbf{W}^{-1}(\mathbf{x}) = \sum_i \mathbf{A}_i(\mathbf{x})\, a_i(\mathbf{x}), \tag{5}$$

$$\text{with} \quad \mathbf{A}_i(\mathbf{x}) = \mathbf{R}_i\left(\mathbf{s}_i \circ (\mathbf{x} - \mathbf{t}_i)\right), \qquad a_i(\mathbf{x}) = \frac{w_i(\mathbf{A}_i(\mathbf{x}))}{\sum_j w_j(\mathbf{A}_j(\mathbf{x}))}, \tag{6}$$

where $\mathbf{A}_i(\mathbf{x})$ is the i^{th} affine transformation, $\{\mathbf{R}_i, \mathbf{s}_i, \mathbf{t}_i\}$ define the rotation, scaling, and translation of the i^{th} affine transformation parameters, $\circ$ is element-wise multiplication, and $w_i(\mathbf{x})$ is the weight volume of the i^{th} warp. Note that we sample the spatial mixture weight w_i after warping ("warped weights"). The intuition behind this is that the warped space represents different parts of the scene, and the weighting function should be in that space as well. This can be viewed as an extension of linear blend skinning [Lewis et al. 2000] applied to a volumetric space.

To compute the transformation parameters $\{\mathbf{R}_i, \mathbf{s}_i, \mathbf{t}_i\}$ and the weighting volume w_i we use 2 fully-connected layers after the encoding $\mathbf{z}$. For rotation, we produce a rotation quaternion vector which is normalized and transformed into a rotation matrix. Before outputting the values of the weighting volume, we apply $\exp(\cdot)$ to ensure the weights are non-negative. Unlike the voxel volume $\mathbf{V}$, we clamp samples outside the weighting volume w_i to the surface otherwise the warps can get "stuck" early in training if they land outside the volume. In practice, we found that a mixture of 16 warps provides sufficient expressiveness. In all experiments where warping is used, we learn an additional global warp $\{\mathbf{R}_g, \mathbf{s}_g, \mathbf{t}_g\}$ (with parameters produced by MLP) that is applied to $\mathbf{x}$ before the warping field, and we use a 32^3 voxel grid to represent the warp field as we found that the low resolution provides smoothness from the trilinear interpolation that helps learning.

5.5 View Conditioning

In order to model view-dependent appearance, we opt to condition the RGB decoder network on the viewpoint. This allows us to model specularities in a data-driven way, without specifying a particular functional form. To do this, we input the normalized direction of the camera to the decoder alongside the encoding. Note that for view-conditioned models, we use separate convolutional branches to produce the RGB values and α values as we only want to condition the RGB values on the viewpoint.

6 ACCUMULATIVE RAY MARCHING

We formulate a rendering process for semi-transparent volumes that mimics front-to-back additive blending: As a camera ray traverses a volume of inhomogeneous material, it accumulates color in proportion to the local color and density of the material at each point along its path.

6.1 Semi-Transparent Volume Rendering

We generate images from a volume function $\mathbf{V}(\mathbf{x})$ by marching rays through the volume it models. To model occlusions, the ray accumulates not only color but also opacity. If the accumulated opacity reaches 1 (for example, when the ray traverses an opaque region), then no further color can be accumulated on the ray. In particular, the color $\mathbf{I}_{\text{rgb}}(\mathbf{p})$ at a pixel $\mathbf{p}$ in the focal plane of a camera with center $\mathbf{r}_o \in \mathbb{R}^3$ and image-to-world transformation $\mathbf{P}^{-1}$ is given by raymarching in the unit direction $\mathbf{r}_d = (\mathbf{P}^{-1}\mathbf{p} - \mathbf{r}_o)/\|\mathbf{P}^{-1}\mathbf{p} - \mathbf{r}_o\| \in \mathbb{R}^3$. This leads to the rendering process

$$\mathbf{I}_{\text{rgb}}(\mathbf{p}) = \int_{t_{\text{min}}}^{t_{\text{max}}} \mathbf{V}_{\text{rgb}}(\mathbf{r}_o + t\mathbf{r}_d)\, \frac{\mathrm{d}\alpha(t)}{\mathrm{d}t}\, \mathrm{d}t, \tag{7}$$

$$\text{with} \quad \alpha(t) = \min\left(\int_{t_{\text{min}}}^{t} \mathbf{V}_\alpha(\mathbf{r}_o + s\mathbf{r}_d)\ \mathrm{d}s, 1\right), \tag{8}$$

ALGORITHM 1: Accumulative rendering of ray $\mathbf{r}_o+t\mathbf{r}_d$ with segment $t\in[t_{\min}, t_{\max}]$ intersecting $\mathbf{V}$. (Integration of Eq. (7) with stepsize Δ.)

$\mathbf{I}_{\text{rgb}}=\mathbf{0}$; $\mathbf{I}_\alpha=0$; $t=t_{\min}$;
repeat
  $\mathrm{d}\alpha(t) = \min\left(\mathbf{I}_\alpha + \Delta\mathbf{V}_\alpha(\mathbf{r}_o + t\mathbf{r}_d), 1\right) - \mathbf{I}_\alpha$
  $\mathbf{I}_{\text{rgb}} = \mathbf{I}_{\text{rgb}} + \mathbf{V}_{\text{rgb}}(\mathbf{r}_o + t\mathbf{r}_d)\,\mathrm{d}\alpha(t)$
  $\mathbf{I}_\alpha = \mathbf{I}_\alpha + \mathrm{d}\alpha(t)$
  $t = t + \Delta$
until $\mathbf{I}_\alpha=1$ *or* $t>t_{\max}$
return $\mathbf{I}_{\text{rgb}}$, $\mathbf{I}_\alpha$

where $t\in[t_{\min}, t_{\max}]$ denotes the segment of the ray that intersects the volume modeled by $\mathbf{V}$, and $\min(\cdot, 1)$ in Eq. (8) ensures that the accumulated ray opacity is clamped at 1. Furthermore, we set the final image opacities to $\mathbf{I}_\alpha(\mathbf{p}) = \alpha(t_{\max})$.

Algorithm 1 shows the computation of the output color for a ray intersecting the volume $\mathbf{V}$, and represents the numerical integration of Eqs. (7–8) using the rectangle rule[1]. Importantly, this image formation model is differentiable, which allows us to optimize the parameters of the volume $\mathbf{V}$ to match target images. In practice, we set the step size to be $\frac{1}{128}^{\text{th}}$ the size of the volume, which provides adequate voxel coverage and allows the rendering process to run at 90Hz in an OpenGL shader at 512×600 resolution, enabling real-time stereo in Virtual Reality.

6.2 Hybrid Rendering

While the semi-transparent volume representation is very versatile, certain kinds of scene content can be more efficiently represented at high resolution using surface-based representations combined with unwrapped texture maps. One example is fine detail in human faces, for which specialized capture systems are available and commonly use texture resolutions larger than 1024^2 [Beeler et al. 2011; Fyffe et al. 2017; Lombardi et al. 2018].

The volume representation described above offers a natural integration with such existing mesh-based representations. Rendering proceeds as described in Algorithm 1 with one modification: we set $t_{\max}$ to the mesh depth for all rays that intersect the mesh. Whenever one of these rays reaches $t_{\max}$, any remaining color throughput is filled with the color of the mesh at the intersection.

This technique can also be used during learning to avoid expending representational power in these parts of the scene. As we show in §8.6, the resulting semi-transparent volume learned using the hybrid rendering process described above naturally avoids occluding the mesh in areas where the mesh provides a higher-fidelity representation.

7 END-TO-END TRAINING

In this section we discuss the details of training our method. Training the system consists of training the weights θ of the encoder-decoder network. We discuss the estimation of a per-camera color calibration matrix and static background image, the construction of our loss function, and reconstruction priors that improve accuracy.

[1]We use the rectangle rule twice with samples on the right of each step interval and backwards differences to discretize the derivative of $\alpha(t)$.

7.1 Color Calibration

Although we have geometrically calibrated the cameras in our multi-view system, we have not color calibrated them relative to one-another. We need to ensure that one radiance value will be converted to the same pixel value for each of the cameras. To do this, we introduce a per-camera and per-channel gain g and bias b that is applied to our reconstructed image before comparing to ground truth. This allows our system to account for slight differences in overall intensity in the image.

7.2 Backgrounds

In our training data we often have static backgrounds that the algorithm will try to reconstruct. To ensure that our algorithm only reconstructs the object of interest, we estimate a per-camera background image $\mathbf{I}^{(\text{bg})}_{\text{rgb}}$. The background image is static across the entire sequence, capturing only stationary objects that are generally outside of the reconstruction volume.

We obtain a final image $\widehat{\mathbf{I}}_{\text{rgb}}$ from a specific view by raymarching all pixels $\mathbf{p}$ according to Eq. (7) and merging $\mathbf{I}_{\text{rgb}}(\mathbf{p})$ with its corresponding background pixel $\mathbf{I}^{(\text{bg})}_{\text{rgb}}(\mathbf{p})$ according to the remaining opacity when exiting the volume,

$$\widehat{\mathbf{I}}_{\text{rgb}}(\mathbf{p}) = (1 - \mathbf{I}_\alpha(\mathbf{p}))\,\mathbf{I}^{(\text{bg})}_{\text{rgb}}(\mathbf{p}) + \left(g\mathbf{I}_{\text{rgb}}(\mathbf{p}) + b\right). \tag{9}$$

This background estimation process greatly reduces the amount of artifacts in the reconstruction.

7.3 Reconstruction Priors

Without using priors, the reconstructed volumes tend to include smoke-like artifacts. These artifacts are caused by slight differences in appearance from different viewpoints due to calibration errors or view-dependent effects. The system can learn to compensate for these differences by adding a small amount of opacity that becomes visible from one particular camera. To reduce these artifacts, we introduce two priors.

The first prior regularizes the total variation of the log voxel opacities,

$$J_{\text{tv}}(\mathbf{V}_\alpha) = \frac{1}{N}\sum_{\mathbf{x}} \lambda_{\text{tv}} \left\| \frac{\partial}{\partial \mathbf{x}} \log \mathbf{V}_\alpha(\mathbf{x}) \right\|, \tag{10}$$

where the sum is performed over all the voxel centers $\mathbf{x}$ and N is the number of voxels. This term helps recover sharp boundaries between opaque and transparent regions by enforcing sparse spatial gradients. We apply this prior in log space to increase the sensitivity of the prior to small α values because the artifacts tend to be mostly transparent.

The second prior is a beta distribution ($\text{Beta}(0.5, 0.5)$) on the final image opacities $\mathbf{I}_\alpha(\mathbf{p})$. We write the regularization term using the negative log-likelihood of the beta distribution,

$$J_{\text{B}}\left(\mathbf{I}_\alpha\right) = \frac{1}{P}\sum_{\mathbf{p}} \lambda_{\text{B}} \left[\log\left(\mathbf{I}_\alpha(\mathbf{p})\right) + \log\left(1 - \mathbf{I}_\alpha(\mathbf{p})\right)\right], \tag{11}$$

where $\mathbf{p}$ is an image pixel and P is the number of pixels. This prior reduces the entropy of the exit opacities and is based on the intuition that most of our rays should strike the object or the background;

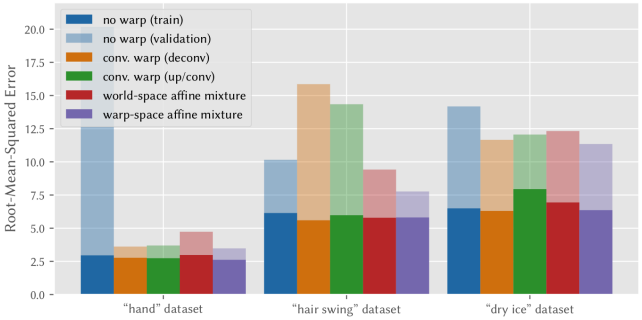


Fig. 5. Warping method evaluation. In this experiment we evaluate different techniques for warping. We evaluate on three different datasets: a moving hand, swinging hair, and dry ice smoke. We evaluate models with: no warping, convolutional warp (using transposed convolutions), convolutional warp (using trilinear upsampling followed by convolution), no warp affine mixture, where the spatial weighting volume w_i is not warped before evaluating (Eq. (13)), and warp-space affine mixture, as described in Eq. (6). In all cases, our warp-space affine mixture outperforms the other models.

fewer rays will graze the surface of the object, picking up some opacity but not saturating.

7.4 Training Hyperparameters

Our full training objective is

$$\ell(\theta) = \frac{1}{P}\sum_{\mathbf{p}} \left\| \widehat{\mathbf{I}}_{\text{rgb}}(\mathbf{p}) - \mathbf{I}^*_{\text{rgb}}(\mathbf{p}) \right\|^2 + \lambda_{\text{KL}} D_{\text{KL}}\left(\mathbf{z} \,\|\, \mathcal{N}(0,1)\right) + J_{\text{tv}}(\mathbf{V}_\alpha) + J_{\text{B}}\left(\mathbf{I}_\alpha\right), \tag{12}$$

where $\mathbf{I}^*_{\text{rgb}}(\mathbf{p})$ is the ground truth image and $D_{\text{KL}}(\mathbf{z} \,\|\, \mathcal{N}(0,1))$ is the KL divergence between the latent encoding $\mathbf{z}$ and a standard normal distribution used in a variational autoencoder [Kingma and Welling 2013].

We use Adam [Kingma and Ba 2015] to optimize the loss function. In our experiments, we set $\lambda_{\text{KL}} = 0.001$, $\lambda_{\text{tv}} = 0.01$, and $\lambda_{\text{B}} = 0.1$ and we use a batch size of 16 with a fixed learning rate of 10^{-4} on the encoder-decoder network weights (the estimated background images $\mathbf{I}^{(\text{bg})}$ and per-camera gain and bias g, b are given separate learning rates of 10^{-1} and 10^{-3}, respectively). We randomly sample 128×128 pixels from the image to reduce memory usage. While raymarching, we compute step sizes in normalized voxel space (i.e., $[-1, 1]^3$) as it makes the end of the ray the expected saturation point at network initialization. We train for about 500,000 iterations, depending on dataset size, which takes 10 days on a single NVIDIA Tesla V100.

8 EXPERIMENTS

We perform a number of quantitative and qualitative experiments to validate our model. For all experiments, we use the convolutional volume decoder with size 128^3. We show that the design choices of our model provide a good compromise between quality and speed, and that the model generalizes to new viewpoints. We show results of our method on objects that are typically hard to reconstruct (e.g., fuzz, smoke, and hair) and demonstrate the method combined with traditional triangle rasterization. Finally, we demonstrate animation of our models by interpolating in the latent space and driving the reconstruction from user input.

Table 1. View conditioning comparison. Conditioning the decoder on viewpoint improves the reconstruction on a set of validation viewpoints.

	Face	
	Train	Val.
No view conditioning	51.1	117.8
View conditioning	**38.7**	**85.7**

To capture data, we used a multi-camera capture system consisting of 34 4096×2668-resolution 30Hz color cameras placed on a hemisphere with a radius of approximately one meter. We calibrate the camera system using an icosahedral checkerboard pattern [Ha et al. 2017]. The raw images and camera calibration are then used as the only input to our method.

In our quantitative experiments, we compare mean-squared error of pixel reconstructions on the training cameras and also on a set of 7 held-out validation cameras. This allows us to test how well our model extrapolates to novel viewpoints.

8.1 Warping Method

To validate our warp representation, we compare against several variants: a model with no warping, a model with a warp produced by a convolutional neural network, a model that does not apply the warp before computing the affine mixture weight, and the proposed affine mixture warp model (i.e., Eq. (5)).

For the model that does not apply the warp to compute the mixture, we modify Eq. (6) as follows:

$$\text{with} \quad A_i(\mathbf{x}) = \mathbf{R}_i\left(\mathbf{s}_i \circ (\mathbf{x} - \mathbf{t}_i)\right), \qquad a_i(\mathbf{x}) = \frac{w_i(\mathbf{x})}{\sum_j w_j(\mathbf{x})}. \tag{13}$$

The main difference is that the mixture is done in "world" space rather than in warped space. Since the mixture weights are not warped, the decoder must match any motion of the template by moving the mixture weights.

Fig. 5 shows the results of our warping evaluation on three datasets: a moving hand, swinging hair, and dry ice smoke, each approximately 20 seconds long. In each dataset, our affine mixture model outperforms models with no warp and outperforms models with a convolutional warp field on the validation cameras. In particular, the convolutional warp fields completely fail on the "hair swing" dataset. The results also show that modeling the weighting volume in warp space is better than in "world" space.

8.2 View-Conditioning

An important part of rendering is being able to model view-dependent effects such as specularities. To do this, we can condition the RGB decoder $\mathbf{V}_{\text{rgb}}$ on the viewpoint of the rendered view. This allows the network to change the color of certain parts of the scene depending on the angle it's viewed from.

Table 1 shows a quantitative experiment comparing a view-conditioned model to a non-view-conditioned model for a human

Fig. 6. Qualitative results. In this figure we show ground truth, reconstructed images, and a visualization of the root-mean-squared error for each pixel for 3 datasets on 3 held-out validation viewpoints.

Table 2. Evaluation of priors and background model. We show training and validation MSE for the "fuzzy toy" object with and without priors and using a known background, learned background, and no background model. Surprisingly, the learned background model with priors outperforms a known background.

	Fuzzy Toy	
Background Model	Train	Val.
Known BG / priors	87.4	197.6
Known BG / no priors	**76.1**	330.5
Learned BG / priors	115.4	**183.7**
Learned BG / no priors	94.1	281.7
No BG / priors	208.8	386.7
No BG / no priors	86.6	727.6

face. The results show that the view-conditioned model is better able to model the scene from novel viewpoints. This happens not only because the view-conditioned model can represent some of the view-dependent appearance of the scene, but also because the non-view-conditioned model incorrectly reconstructs extra semi-transparent voxels near the surface of the object to explain view-dependent phenomena.

8.3 Priors and Background Estimation

We impose several priors on the reconstructed volume to help reduce the occurrence of artifacts in the reconstruction. In this experiment, we evaluate the effectiveness of background estimation and the priors on the reconstructed volume quality in terms of mean-squared-error on the validation viewpoints. We evaluate 3 different scenarios: known background image, learned background image, and no background model, each scenario evaluated with and without priors.

Table 2 shows the results of this experiment. For each background setting, using the priors improves performance on the validation viewpoints. Surprisingly, learning a background image outperformed using a known background image, but the improvement is small.

8.4 Qualitative Results

In this section, we show qualitative results on a number of different types of objects. We compare our renderings to held-out ground truth views and also demonstrate hybrid rendering with a mesh representation.

Fig. 6 shows renderings produced by our method compared to ground truth images for several validation (held-out) viewpoints. The renderings demonstrate that our method is able to model difficult phenomena like fuzz, smoke, and human skin and hair. The figure also shows typical artifacts produced by our reconstructions: typically, a very light smokey pattern is added which may be modeling view-dependent appearance for certain training cameras. Using more camera views tends to reduce all artifacts.

Fig. 7 shows how the template volume changes through time across several frames, compared to the final warped volume that is rendered. Ideally, object motion should be represented entirely by the warp field. However, this does not always happen when representing such changes requires more resolution than is available

Fig. 7. Warped and template volumes through time. This figure depicts several frames from two sequences, showing (a) the final warped volumes, and (b) the learned template volumes before warping. For the face sequence, most changes are explained via warping and the face remains static in template space. However, because tracking is not explicitly enforced, the network may learn to represent motion as different templates if it is more parsimonious to do so, as can be seen for finger motion in the hand sequence. Note also how in the template, space is allocated to regions requiring texture details while flat regions are compressed, like the phalanges and the forearm in the hand example.

(e.g., as would be necessary to cleanly separate the rim of the glasses from the side of the head) or because representing the warp field becomes too complex.

Fig. 11 visualizes the learned RGBα volumes. While there is no explicit surface reconstruction objective in our method, we can visualize isosurfaces of constant opacity, shown in (b). Ideally, fully opaque surfaces such as the hand in the first row would be represented by delta functions in $\mathbf{V}_\alpha(\mathbf{x})$, but we find that the method trades off some opacity to better match reconstruction error in the training views. Note how translucent materials such as the glasses in the second row, or materials which appear translucent at coarse resolution, such as hair in the 3rd and 4th rows, are modeled with lower opacity values but retain a distinct structure particular to the object.

8.5 Single Frame Estimation

We evaluate the quality of our algorithm using only a single frame as input. In some ways this should make the problem easier as there is less information to represent within the model. On the other hand, our model is less able to exploit regularities and redundancies of motion. This experiment helps us disentangle those factors and determine the contribution of each component of the model.

To perform this experiment, we run our model on only a single frame. Although the encoder will produce a constant value, we keep the entire encoder-decoder network intact while training. We also compare to "direct" volume estimation, i.e., we directly estimate the template voxel volume $\mathbf{T}$ and warp values $\mathbf{W}^{-1}$ without the encoder-decoder network.

Fig. 12 shows the results of the experiment on four objects we captured as well as one scene from a publicly available MVS dataset [Aanæs et al. 2016]. As shown, the "direct" voxel/warp estimation contains artifacts and incorrectly reconstructs the object. This experiment shows that the convolutional architecture provides a regularization that, even in the case of a single frame, allows us to accurately recover and re-render objects. Similar observations have been made in a recent work on deep image priors [Ulyanov et al. 2018].

For comparison, we show the same frame reconstructed as a mesh using a commercial multi-view stereo system [Agisoft 2019] and the open-source multi-view stereo system COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016], as well as a comparison to space carving [Kutulakos and Seitz 2000]. Characteristically, the recovered resolution in the MVS texture map is greater than what we can achieve with volume reconstructions on current hardware. However, the mesh also shows artifacts in thin regions, like the frame of the glasses, and translucent regions, like the glass material and the hair at the top of the head, or the smoke in the 4th row. Space carving heavily relies on consistent appearance across views and struggles with translucent materials.

8.6 Animating

With the proposed reconstruction method, we can playback and re-render the captured data from many different angles. We not only want to extrapolate in viewpoint, but also in the content of the performance. Our latent variable model allows us to create new sequences of content by modifying the latent variables.

Fig. 8 shows two examples of content modification by interpolating latent codes and by changing the conditioning variable based on user input. Our latent space interpolation shows that the encoder network learns a compact representation of the scene. In the second example, we condition the decoder on the head pose of the subject, allowing us to create novel sequences in real time.

Fig. 9 shows the results of combining a textured mesh representation with our voxel representation. Textured meshes can efficiently and accurately represent fine detail in regions of the face like the skin and eyes while the voxel representation excels at modeling hair. For a mesh model we use the Deep Appearance Model of Lombardi et al. [2018] trained to reconstruct the same face we used to train our volume encoder/decoder network.

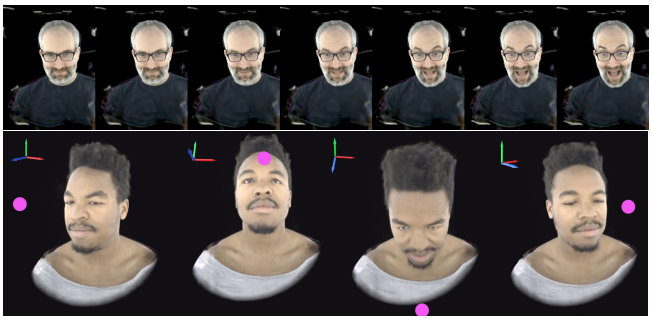


Fig. 8. Novel content generation. First row: Interpolation in latent space. Leftmost and rightmost frames are reconstructed from real image encodings, intermediate frames are reconstructed from linear interpolation between the left and right codes. Second row: real-time avatar driving based on user input. The magenta dot represents the position of the user's hand, which the avatar's head turns to track.

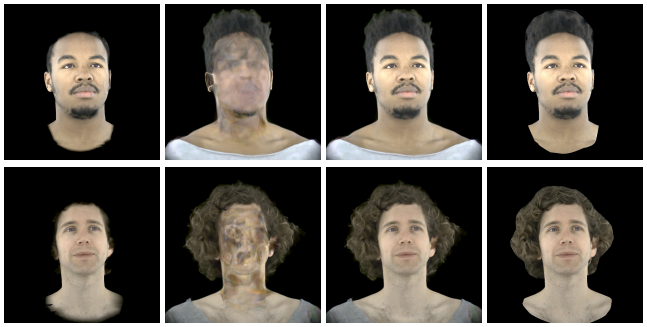


Fig. 9. Combining mesh and volumetric representations (images best viewed at high-resolution). Given an initial textured mesh model, all rays which intersect the mesh have $t_{\max}$ set to the depth of the mesh along each ray. Rays which terminate at $t_{\max}$ accumulate color from the mesh based on remaining opacity. From left to right: masked mesh (placed into voxel volume during learning), voxel representation, hybrid rendering, mesh-only reconstruction. The learned voxel representation avoids occluding the mesh to achieve a higher-quality reconstruction.

9 DISCUSSION

In this paper, we presented a method for modeling objects and scenes with a semi-transparent volume representation that we learn end-to-end from multi-view RGB images. We showed that our method can convincingly reconstruct challenging objects such as moving hair, fuzzy toys, and smoke. Our method requires no explicit tracking, and can be run in real time alongside traditional triangle rasterization.

One limitation of our method is that given a surface with limited texture, our estimated volume may represent that surface as transparent and place its color in the background, so long as doing so does not cause otherwise occluded surfaces to appear. This is a challenge that affects traditional 3D reconstruction methods as well. With our method, however, the reconstruction degrades gracefully and still produces perceptually-pleasing results thanks to our image-space loss function. Fig. 10 shows an example of this via a depth map computed as the distance each ray travels before saturating or hitting the bounding box. In a more practical setting, this could be addressed simply by capturing the sequence with a bright background such as a green screen.

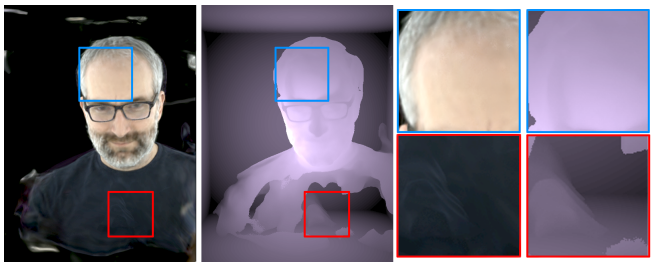


Fig. 10. Depth map showing where raymarching terminates. The hole in the chest indicates that those rays passed through the entire volume and only terminated at the background. Since the chest area has limited texture variation and is a similar color to the background, this artifact does not greatly affect reconstruction error even when viewed from novel viewpoints.

Although our method can handle transparent objects, like plastic bottles, the method doesn't currently consider refractive surfaces. We believe the approach can be extended to model refraction and even reflection, and we leave that to future work. Our model can represent dull specular highlights through view conditioning but high-frequency specular highlights are not correctly represented.

The latent space is the feature enabling us to generate dynamic content, but we do not explicitly model any temporal dynamics. This is not a problem for playback, since the playback sequence implicitly encodes the same temporal dynamics as the recording. It is also not a problem when driving the representation from user input, so long as that user input has reasonable temporal dynamics of its own. However, if we traverse the latent space in some manner not guided by temporal information, we may generate sequences which, while visually accurate, do not represent real behaviors of the object we modeled.

Volumetric representations typically suffer from limited resolution due to the cubic relationship between resolution and memory requirement. In this work, we showed some ways to increase the effective resolution without simply increasing the voxel grid resolution by using warping fields. We believe that we can further improve this approach to achieve a level of fidelity and resolution previously only achievable with traditional textured mesh surfaces.

REFERENCES

Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. 2016. Large-Scale Data for Multiple-View Stereopsis. *International Journal of Computer Vision (IJCV)* (2016).

Agisoft. 2019. Metashape. https://www.agisoft.com/.

Bradley Atcheson, Ivo Ihrke, Wolfgang Heidrich, Art Tevs, Derek Bradley, Marcus Magnor, and Hans-Peter Seidel. 2008. Time-resolved 3D Capture of Non-stationary Gas Flows. *ACM Trans. Graph.* 27, 5 (2008).

Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W. Sumner, and Markus Gross. 2011. High-quality Passive Facial Performance Capture Using Anchor Frames. *ACM Trans. Graph.* 30, 4 (2011).

Jeremy S. De Bonet and Paul A. Viola. 1999. Poxels: Probabilistic Voxelized Volume Reconstruction. In *International Conference on Computer Vision (ICCV)*.

Adrian Broadhurst, Tom Drummond, and Roberto Cipolla. 2001. A Probabilistic Framework for Space Carving. In *International Conference on Computer Vision (ICCV)*.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured Lumigraph Rendering. In *Conference on Computer Graphics and*

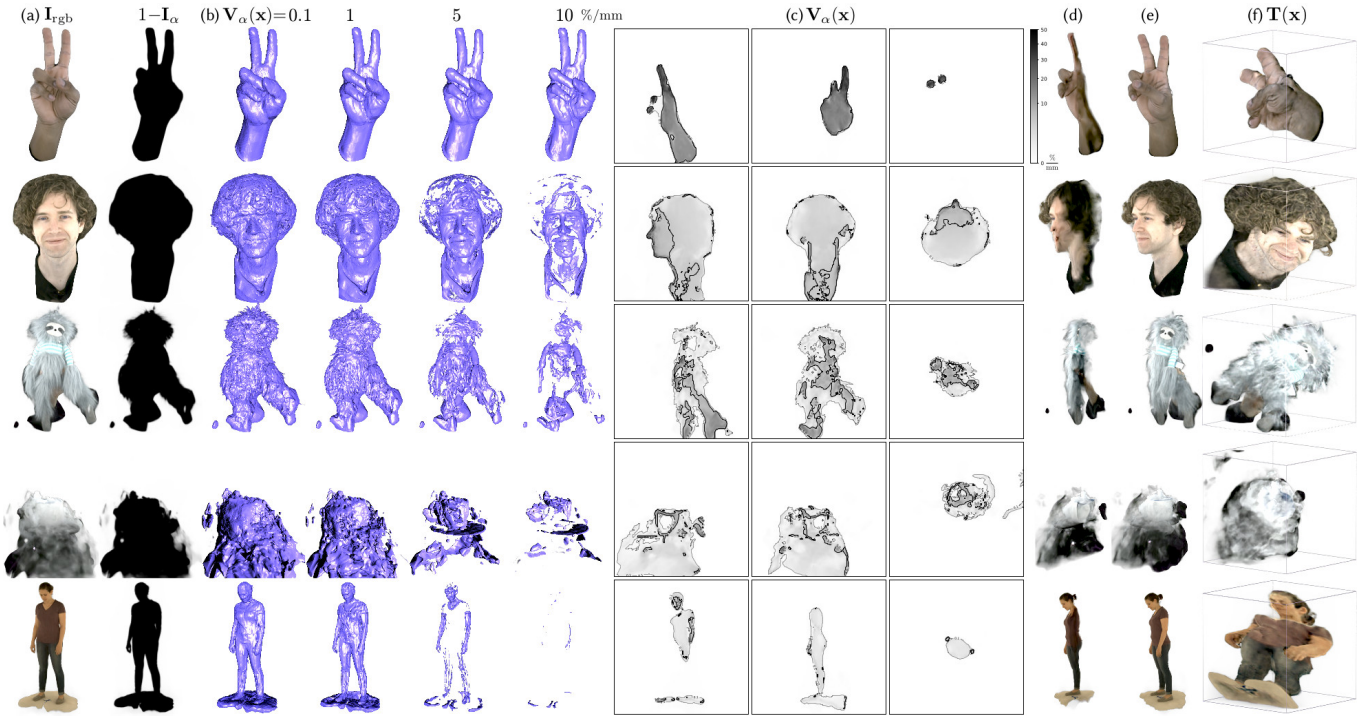


Fig. 11. Learned reconstructions for several objects showing (a) rendered color and opacity, (b) isosurfaces of $\mathbf{V}_{\alpha}(\mathbf{x})$ for different levels of differential opacity, (c) contour plots of the differential opacity, (d) sliced render showing the interior of the volume at the x-y cut, (e) the complete render from the same viewpoint, and (f) the unwarped template.

Interactive Techniques (SIGGRAPH).

Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-quality Streamable Free-viewpoint Video. *ACM Trans. Graph.* 34, 4 (2015).

J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. 2017. Deformable Convolutional Networks. In *International Conference on Computer Vision (ICCV).*

Abe Davis, Marc Levoy, and Fredo Durand. 2012. Unstructured Light Fields. *Computer Graphics Forum* 31, 2pt1 (2012).

Andrew Fitzgibbon and Andrew Zisserman. 2005. Image-based Rendering Using Image-based Priors. In *International Conference on Computer Vision (ICCV).*

Yasutaka Furukawa and Carlos Hernández. 2015. Multi-View Stereo: A Tutorial. *Foundations and Trends in Computer Graphics and Vision* 9, 1-2 (2015).

Yasutaka Furukawa and Jean Ponce. 2010. Accurate, Dense, and Robust Multiview Stereopsis. *Pattern Analysis and Machine Intelligence (PAMI)* 32, 8 (2010).

G. Fyffe, K. Nagano, L. Huynh, S. Saito, J. Busch, A. Jones, H. Li, and P. Debevec. 2017. Multi-View Stereo on Consistent Face Topology. *Computer Graphics Forum* 36, 2 (2017).

Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. 2007. Multi-View Stereo for Community Photo Collections. In *International Conference on Computer Vision (ICCV).*

H. Ha, M. Perdoch, H. Alismail, I. S. Kweon, and Y. Sheikh. 2017. Deltille Grids for Geometric Camera Calibration. In *International Conference on Computer Vision (ICCV).*

Tim Hawkins, Per Einarsson, and Paul Debevec. 2005. Acquisition of Time-varying Participating Media. *ACM Trans. Graph.* 24, 3 (2005).

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep Blending for Free-Viewpoint Image-Based Rendering. *ACM Trans. Graph.* 37, 6 (2018).

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations (ICLR).*

Milan Ikits, Joe Kniss, Aaron Lefohn, and Charles Hansen. 2004. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics (Chapter 39, Volume Rendering Techniques).* Addison Wesley.

Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. 2016. VolumeDeform: Real-Time Volumetric Non-rigid Reconstruction. In *European Conference on Computer Vision (ECCV).*

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *Computer Vision and Pattern Recognition (CVPR)* (2017).

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. 2015. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems (NeurIPS).*

Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. 2016. Learning-based View Synthesis for Light Field Cameras. *ACM Trans. Graph.* 35, 6 (2016).

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations (ICLR).*

Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Nießner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. 2018. Deep Video Portraits. *ACM Trans. Graph.* 37, 4 (2018).

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference for Learning Representations (ICLR).*

Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR).*

Kiriakos N. Kutulakos and Steven M. Seitz. 2000. A Theory of Shape by Space Carving. *International Journal of Computer Vision* 38, 3 (2000).

Marc Levoy. 1988. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications* 8, 3 (1988).

J. P. Lewis, Matt Cordner, and Nickson Fong. 2000. Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-driven Deformation. In *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH).*

Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. 2018. Deep Appearance Models for Face Rendering. *ACM Trans. Graph.* 37, 4 (2018).

Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, and Sean Fanello. 2018. LookinGood: Enhancing Performance Capture with Real-time Neural Re-rendering. *ACM Trans. Graph.* 37, 6 (2018).

Fig. 12. Single frame estimation. In this experiment, we reconstruct only a single frame rather than a sequence. For comparison, from left to right, the first three columns show MVS-based reconstructions: (a) the point cloud recovered by COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016], and (b) the mesh after Poisson reconstruction, and (c) the textured mesh recovered by Agisoft Metashape [2019]. Column (d) shows a colored voxel occupancy reconstruction using space carving [Kutulakos and Seitz 2000], and column (e) shows "direct" estimation of the voxel grid (i.e., the $\mathbf{T}$ and $\mathbf{W}^{-1}$ are directly estimated instead of the result of an encoder-decoder network). Finally, (f) shows the results of our full pipeline trained only on a single frame. The results show that the encoder-decoder architecture facilitates recovering accurate estimates.

Paul Merrell, Amir Akbarzadeh, Liang Wang, Philippos Mordohai, Jan-Michael Frahm, Ruigang Yang, David NistÃĺr, and Marc Pollefeys. 2007. Real-Time Visibility-Based Fusion of Depth Maps. In *International Conference on Computer Vision (ICCV)*.

Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. 2015. DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time. In *Computer Vision and Pattern Recognition (CVPR)*.

Thu H Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yongliang Yang. 2018. RenderNet: A Deep Convolutional Network for Differentiable Rendering from 3D Shapes. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. 2013. Real-time 3D Reconstruction at Scale Using Voxel Hashing. *ACM Trans. Graph.* 32, 6 (2013).

Ryan S. Overbeck, Daniel Erickson, Daniel Evangelakos, Matt Pharr, and Paul Debevec. 2018. A System for Acquiring, Processing, and Rendering Panoramic Light Field Stills for Virtual Reality. *ACM Trans. Graph.* 37, 6 (2018).

Despoina Paschalidou, Ali Osman Ulusoy, Carolin Schmitt, Luc Gool, and Andreas Geiger. 2018. RayNet: Learning Volumetric 3D Reconstruction with Ray Potentials. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Eric Penner and Li Zhang. 2017. Soft 3D Reconstruction for View Synthesis. *ACM Trans. Graph.* 36, 6 (2017).

Fabián Prada, Misha Kazhdan, Ming Chuang, Alvaro Collet, and Hugues Hoppe. 2016. Motion Graphs for Unstructured Textured Meshes. *ACM Trans. Graph.* 35, 4 (2016).

Andrew Prock and Charles R. Dyer. 1998. Towards Real-Time Voxel Coloring. In *Image Understanding Workshop*.

Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. 2017. OctNet: Learning Deep 3D Representations at High Resolutions. In *Computer Vision and Pattern Recognition (CVPR)*.

S. Roth and M. J. Black. 2006. Specular Flow and the Recovery of Surface Structure. In *Computer Vision and Pattern Recognition (CVPR)*.

Nikolay Savinov, Christian Häne, Lubor Ladicky, and Marc Pollefeys. 2016. Semantic 3D Reconstruction with Continuous Regularization and Ray Potentials Using a Visibility Consistency Constraint. In *Computer Vision and Pattern Recognition (CVPR)*.

Daniel Scharstein and Richard Szeliski. 2002. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision (IJCV)* (2002).

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Computer Vision and Pattern Recognition (CVPR)*.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*.

Steven M. Seitz and Charles R. Dyer. 1997. Photorealistic Scene Reconstruction by Voxel Coloring.

Steven M. Seitz and Charles R. Dyer. 1999. Photorealistic Scene Reconstruction by Voxel Coloring. *International Journal of Computer Vision* 35, 2 (1999).

Zhixin Shu, Mihir Sahasrabudhe, Riza Alp Guler, Dimitris Samaras, Nikos Paragios, and Iasonas Kokkinos. 2018. Deforming Autoencoders: Unsupervised Disentangling of Shape and Appearance. In *European Conference on Computer Vision (ECCV)*.

V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhöfer. 2018. DeepVoxels: Learning Persistent 3D Feature Embeddings. *arXiv:1812.01024 [cs.CV]* (2018).

Richard Szeliski and Polina Golland. 1999. Stereo Matching with Transparency and Matting. *International Journal of Computer Vision (IJCV)* 32, 1 (1999).

L. Torresani, A. Hertzmann, and C. Bregler. 2008. Nonrigid Structure-from-Motion: Estimating Shape and Motion with Hierarchical Priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 30, 5 (2008).

Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. 2018. Multi-view Consistency as Supervisory Signal for Learning Shape and Pose Prediction. In *Computer Vision and Pattern Recognition (CVPR)*.

Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. 2017. Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency. In *Computer Vision and Pattern Recognition (CVPR)*.

Ali Osman Ulusoy, Andreas Geiger, and Michael J. Black. 2015. Towards Probabilistic Volumetric Reconstruction Using Ray Potentials. In *3DV*.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2018. Deep Image Prior. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. Video-to-Video Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zexiang Xu, Hsiang-Tao Wu, Lvdi Wang, Changxi Zheng, Xin Tong, and Yue Qi. 2014. Dynamic Hair Capture Using Spacetime Optimization. *ACM Trans. Graph.* 33, 6 (2014).

Christopher Zach, Thomas Pock, and Horst Bischof. 2007. A Globally Optimal Algorithm for Robust TV-L^1 Range Image Integration. In *International Conference on Computer Vision (ICCV)*.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo Magnification: Learning View Synthesis using Multiplane Images. *ACM Trans. Graph.* 37, 4 (2018).

Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger. 2014. Real-time Non-rigid Reconstruction Using an RGB-D Camera. *ACM Trans. Graph.* 33, 4 (2014).