Mixture of Volumetric Primitives for Efficient Neural Rendering

STEPHEN LOMBARDI, Facebook Reality Labs, USA
TOMAS SIMON, Facebook Reality Labs, USA
GABRIEL SCHWARTZ, Facebook Reality Labs, USA
MICHAEL ZOLLHOEFER, Facebook Reality Labs, USA
YASER SHEIKH, Facebook Reality Labs, USA
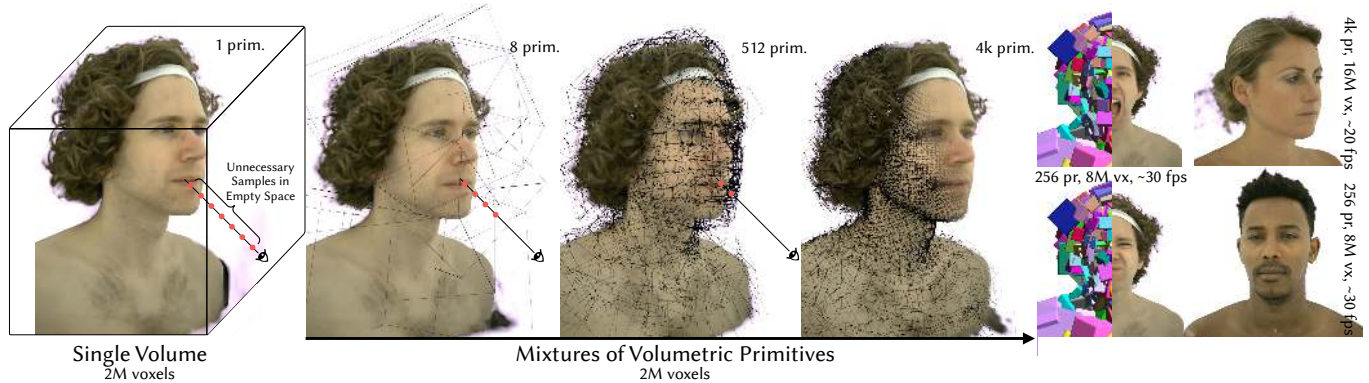JASON SARAGIH, Facebook Reality Labs, USA

Fig. 1. Mixture of Volumetric Primitives (MVP) inherits the strengths of both volumetric as well as primitive-based approaches, while avoiding many of their pitfalls. It enables state-of-the-art results for rendering dynamic objects in terms of rendering quality and runtime performance.

Real-time rendering and animation of humans is a core function in games, movies, and telepresence applications. Existing methods have a number of drawbacks we aim to address with our work. Triangle meshes have difficulty modeling thin structures like hair, volumetric representations like Neural Volumes are too low-resolution given a reasonable memory budget, and high-resolution implicit representations like Neural Radiance Fields are too slow for use in real-time applications. We present Mixture of Volumetric Primitives (MVP), a representation for rendering dynamic 3D content that combines the completeness of volumetric representations with the efficiency of primitive-based rendering, e.g., point-based or mesh-based methods. Our approach achieves this by leveraging spatially shared computation with a convolutional architecture and by minimizing computation in empty regions of space with volumetric primitives that can move to cover only occupied regions. Our parameterization supports the integration of correspondence and tracking constraints, while being robust to areas where classical tracking fails, such as around thin or translucent structures and areas with large topological variability. MVP is a hybrid that generalizes both volumetric and primitive-based representations. Through a series of extensive experiments we demonstrate that it inherits the strengths of each, while avoiding many

of their limitations. We also compare our approach to several state-of-the-art methods and demonstrate that MVP produces superior results in terms of quality and runtime performance.

CCS Concepts: • **Computing methodologies → Rendering**.

Additional Key Words and Phrases: Neural Rendering

1 INTRODUCTION

Photo-realistic rendering of dynamic 3D objects and scenes from 2D image data is a central focus of research in computer vision and graphics. Volumetric representations have seen a resurgence of interest in the graphics community in recent years, driven by impressive empirical results attained using learning-based methods [Lombardi et al. 2019; Mildenhall et al. 2020]. Through the use of generic function approximators, such as deep neural networks, these methods achieve compelling results by supervising directly on raw image pixels. Thus, they avoid the often difficult task of assigning geometric and radiometric properties, which is typically required by classical physics-inspired representations. Leveraging the inherent simplicity of volumetric models, much work has been dedicated to extending the approach for modeling small motions [Park et al. 2020], illumination variation [Srinivasan et al. 2020], reducing data requirements [Trevithick and Yang 2020; Yu et al. 2020], and learning efficiency [Tancik et al. 2020]. All these methods employ a soft

Authors' addresses: Stephen Lombardi, Facebook Reality Labs, USA, stephen.lombardi@fb.com; Tomas Simon, Facebook Reality Labs, USA, tsimon@fb.com; Gabriel Schwartz, Facebook Reality Labs, USA, gbschwartz@fb.com; Michael Zollhoefer, Facebook Reality Labs, USA, zollhoefer@fb.com; Yaser Sheikh, Facebook Reality Labs, USA, yasers@fb.com; Jason Saragih, Facebook Reality Labs, USA, jsaragih@fb.com.

volumetric representation of 3D space that helps them model thin structures and semi-transparent objects realistically.

Despite the aforementioned advances in volumetric models, they still have to make a trade-off; either they have a large memory footprint or they are computationally expensive to render. The large memory footprint drastically limits the resolution at which these approaches can operate and results in a lack of high-frequency detail. In addition, their high computational cost limits applicability to real-time applications, such as VR telepresence [Orts-Escolano et al. 2016; Wei et al. 2019]. The ideal representation would be memory efficient, can be rendered fast, is drivable, and has high rendering quality.

Neural Volumes [Lombardi et al. 2019] is a method for learning, rendering, and driving dynamic objects captured using an outside-in camera rig. The method is suited to objects rather than scenes as a uniform voxel grid is used to model the scene. This grid's $O(n^3)$ memory requirement prevents the use of high resolutions, even on high-end graphics cards. Since much of the scene is often comprised of empty space, Neural Volumes employs a warp field to maximize the utility of its available resolution. The efficacy of this, however, is limited by the resolution of the warp and the ability of the network to learn complex inverse-warps in an unsupervised fashion.

Neural Radiance Fields (NeRF) [Mildenhall et al. 2020] addresses the issue of resolution using a compact representation. NeRF only handles static scenes. Another challenge is runtime, since a multi-layer perceptron (MLP) has to be evaluated at every sample point along the camera rays. This leads to billions of MLP evaluations to synthesize a single high-resolution image, resulting in extremely slow render times around thirty seconds per frame. Efforts to mitigate this rely on coarse-to-fine greedy selection that can miss small structures [Liu et al. 2020]. This approach can not easily be extended to dynamics, since it relies on a static acceleration structure.

In this work, we present Mixture of Volumetric Primitives (MVP), an approach designed to directly address the memory and compute limitations of existing volumetric methods, while maintaining their desirable properties of completeness and direct image-based supervision. It is comprised of a mixture of jointly-generated overlapping volumetric primitives that are selectively ray-marched, see Fig. 1. MVP leverages the conditional computation of ray-tracing to eliminate computation in empty regions of space. The generation of the volumetric primitives that occupy non-empty space leverages the shared computation properties of convolutional deep neural networks, which avoids the wasteful re-computation of common intermediate features for nearby areas, a common limitation of recent methods [Liu et al. 2020; Mildenhall et al. 2020]. Our approach can naturally leverage correspondences or tracking results defined previously by opportunistically linking the estimated placement of these primitives to the tracking results. This results in good motion interpolation. Moreover, through a user-defined granularity parameter, MVP generalizes volumetric [Lombardi et al. 2019] on one end, and primitive-based methods [Aliev et al. 2019; Lombardi et al. 2018] on the other, enabling the practitioner to trade-off resolution for completeness in a straightforward manner. We demonstrate that our approach produces higher quality, more driveable models, and can be evaluated more quickly than the state of the art. Our key technical contributions are:

- A novel volumetric representation based on a mixture of volumetric primitives that combines the advantages of volumetric and primitive-based approaches, thus leading to high performance decoding and efficient rendering.

- A novel motion model for voxel grids that better captures scene motion, minimization of primitive overlap to increase the representational power, and minimization of primitive size to better model and exploit free space.

- A highly efficient, data-parallel implementation that enables faster training and real-time rendering of the learned models.

2 RELATED WORK

In the following, we discuss different scene representations for neural rendering. For an extensive discussion of neural rendering applications, we refer to Tewari et al. [2020].

Point-based Representations. The simplest geometric primitive are points. Point-based representations can handle topological changes well, since no connectivity has to be enforced between the points. Differentiable point-based rendering has been extensively employed in the deep learning community to model the geometry of objects [Insafutdinov and Dosovitskiy 2018; Lin et al. 2018; Roveri et al. 2018; Yifan et al. 2019]. Differentiable Surface Splatting [Yifan et al. 2019] represents the points as discs with a position and orientation. Lin et al. [Lin et al. 2018] learns efficient point cloud generation for dense 3D object reconstruction. Besides geometry, point-based representations have also been employed extensively to model scene appearance [Aliev et al. 2019; Kolos et al. 2020; Lassner and Zollhöfer 2020; Meshry et al. 2019; Wiles et al. 2020]. One of the drawbacks of point-based representations is that there might be holes between the points after projection to screen space. Thus, all of these techniques often employ a network in screen-space, e.g., a U-Net [Ronneberger et al. 2015], to in-paint the gaps. SynSin [Wiles et al. 2020] lifts per-pixel features from a source image onto a 3D point cloud that can be explicitly projected to the target view. The resulting feature map is converted to a realistic image using a screen-space network. While the screen-space network is able to plausibly fill in the holes, point-based methods often suffer from temporal instabilities due to this screen space processing. One approach to remove holes by design is to switch to geometry proxies with explicit topology, i.e., use a mesh-based model.

Mesh-based Representations. Mesh-based representations explicitly model the geometry of an objects based on a set of connected geometric primitives and their appearance based on texture maps. They have been employed, for example, to learn personalized avatars from multi-view imagery based on dynamic texture maps [Lombardi et al. 2018]. Differentiable rasterization approaches [Chen et al. 2019; Kato et al. 2018; Laine et al. 2020; Liu et al. 2019; Loper and Black 2014; Petersen et al. 2019; Ravi et al. 2020; Valentin et al. 2019] enable the end-to-end integration of deep neural networks with this classical computer graphics representation. Recently, of-the-shelf tools for differentiable rendering have been developed, e.g., TensorFlow3D [Valentin et al. 2019], Pytorch3D [Ravi et al. 2020], and Nvidia's nvdiffrast [Laine et al. 2020]. Differentiable rendering strategies have for example been employed for learning 3D face

models [Genova et al. 2018; Luan Tran 2019; Tewari et al. 2018] from 2D photo and video collections. There are also techniques that store a feature map in the texture map and employ a screen-space network to compute the final image [Thies et al. 2019]. If accurate surface geometry can be obtained a-priori, mesh-based approaches are able to produce impressive results, but they often struggle if the object can not be well reconstructed. Unfortunately, accurate 3D reconstruction is notoriously hard to acquire for humans, especially for hair, eyes, and the mouth interior. Since such approaches require a template with fixed topology they also struggle to model topological changes and it is challenging to model occlusions in a differentiable manner.

Multi-Layer Representations. One example of a mixture-based representation are Multi-Plane Images (MPIs) [Srinivasan et al. 2019; Tucker and Snavely 2020; Zhou et al. 2018]. MPIs employ a set of depth-aligned textured planes to store color and alpha information. Novel views are synthesized by rendering the planes in back-to-front order using hardware-supported alpha blending. These approaches are normally limited to a small restricted view-box, since specular surfaces have to be modeled via the alpha blending operator. Local Light Field Fusion (LLFF) [Mildenhall et al. 2019] enlarges the view-box by maintaining and smartly blending multiple local MPI-based reconstructions. Multi-sphere images (MSIs) [Attal et al. 2020; Broxton et al. 2020] replace the planar and textured geometry proxies with a set of textured concentric sphere proxies. This enables 360° inside-out view synthesis for VR video applications. MatryODShka [Attal et al. 2020] enables real-time 6DoF video view synthesis for VR by converting omnidirectional stereo images to MSIs.

Grid-based Representations. Grid-based representations are similar to the multi-layer representation, but are based on a dense uniform grid of voxels. They have been extensively used to model the 3D shape of objects [Choy et al. 2016; Kar et al. 2017; Mescheder et al. 2019; Peng et al. 2020; Tulsiani et al. 2017; Wu et al. 2016]. Grid-based representations have also been used as the basis for neural rendering techniques to model object appearance [Lombardi et al. 2019; Sitzmann et al. 2019a]. DeepVoxels [Sitzmann et al. 2019a] learns a persistent 3D feature volume for view synthesis and employs learned ray marching. Neural Volumes [Lombardi et al. 2019] is an approach for learning dynamic volumetric reconstructions from multi-view data. One big advantage of such representations is that they do not have to be initialized based on a fixed template mesh and are easy to optimize with gradient-based optimization techniques. The main limiting factor for all grid-based techniques is the required cubic memory footprint. The sparser the scene, the more voxels are actually empty, which wastes model capacity and limits resolution. Neural Volumes employs a warping field to maximize occupancy of the template volume, but empty space is still evaluated while raymarching. We propose to model deformable objects with a set of rigidly-moving volumetric primitives.

MLP-based Representations. Multi-Layer Perceptrons (MLPs) have first been employed for modeling 3D shapes based on signed distance [Chabra et al. 2020; Jiang et al. 2020; Park et al. 2019; Saito et al. 2019a,b] and occupancy fields [Genova et al. 2020; Mescheder et al. 2019; Peng et al. 2020]. DeepSDF [Park et al. 2019] is one of

the first works that learns the 3D shape variation of an entire object category based on MLPs. ConvOccNet [Peng et al. 2020] enables fitting of larger scenes by combining an MLP-based scene representation with a convolutional decoder network that regresses a grid of spatially-varying conditioning codes. Afterwards, researchers started to also model object appearance using similar scene representations. Neural Radiance Fields (NeRF) [Mildenhall et al. 2020] proposes a volumetric scene representaiton based on MLPs. One of its challenges is that the MLP has to be evaluated at a large number of sample points along each camera ray. This makes rendering a full image with NeRF extremely slow. Furthermore, NeRF can not model dynamic scenes or objects. Scene Representation Networks (SRNs) [Sitzmann et al. 2019b] can be evaluated more quickly as they model space with a signed-distance field. However, using a surface representation means that it can not represent thin structures or transparency well. Neural Sparse Voxel Fields (NSVF) [Liu et al. 2020] culls empty space based on an Octree acceleration structure, but it is extremely difficult to extend to dynamic scenes. There exists also a large number of not-yet peer-reviewed, but impressive extensions of NeRF [Du et al. 2020; Gafni et al. 2020; Gao et al. 2020; Li et al. 2020; Martin-Brualla et al. 2020; Park et al. 2020; Rebain et al. 2020; Schwarz et al. 2020; Tretschk et al. 2020; Xian et al. 2020]. While the results of MLP-based models are often visually pleasing, their main drawbacks are limited or no ability to be driven as well as their high computation cost for evaluation.

Summary. Our approach is a hybrid that finds the best trade-off between volumetric- and primitive-based neural scene representations. Thus, it produces high-quality results with fine-scale detail, is fast to render, drivable, and reduces memory constraints.

3 METHOD

Our approach is based on a novel volumetric representation for dynamic scenes that combines the advantages of volumetric and primitive-based approaches to achieve high performance decoding and efficient rendering. In the following, we describe our scene representation and how it can be trained end-to-end based on 2D multi-view image observations.

3.1 Neural Scene Representation

Our neural scene representation is inspired by primitive-based methods, such as triangular meshes, that can efficiently render high resolution models of 3D space by focusing representation capacity on occupied regions of space and ignoring those that are empty. At the core of our method is a set of minimally-overlapping and dynamically moving volumetric primitives that together parameterize the color and opacity distribution in space over time. Each primitive models a local region of space based on a uniform voxel grid. This provides two main advantages that together lead to a scene representation that is highly efficient in terms of memory consumption and is fast to render: 1) fast sampling within each primitive owing to its uniform grid structure, and 2) conditional sampling during ray marching to avoid empty space and fully occluded regions. The primitives are linked to an underlying coarse *guide* mesh (see next section) through soft constraints, but can deviate away from the mesh if this leads to improved reconstruction. Both, the primitives'

motion as well as their color and opacity distribution are parameterized by a convolutional network that enables the sharing of computation amongst them, leading to highly efficient decoding.

3.1.1 Guide Mesh.

We employ a coarse estimate of the scene's geometry $\{\mathcal{M}_i\}_{i=1}^{N_{\text{frames}}}$ of every frame as basis for our scene representation. For static scenes, it can be obtained via off-the-shelf reconstruction packages such as COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016]. For dynamic scenes, we employ multi-view non-rigid tracking to obtain a temporally corresponded estimate of the scenes geometry over time [Wu et al. 2018]. These meshes guide the initialization of our volumetric primitives, regularize the results, and avoid the optimization terminating in a poor local minimum. Our model generates both the guide mesh as well as its weakly attached volumetric primitives, enabling the direct supervision of large motion using results from explicit tracking. This is in contrast to existing volumetric methods that parameterize explicit motion via an inverse warp [Lombardi et al. 2019; Park et al. 2020], where supervision is more challenging to employ.

3.1.2 Mixture of Volumetric Primitives.

The purpose of each of the N_{prim} volumetric primitives is to model a small local region of 3D space. Each volumetric primitive $\mathcal{V}_k = \{\mathbf{t}_k, \mathbf{R}_k, \mathbf{s}_k, \mathbf{V}_k\}$ is defined by a position $\mathbf{t}_k \in \mathbb{R}^3$ in 3D space, an orientation given by a rotation matrix $\mathbf{R}_k \in \text{SO}(3)$ (computed from an axis-angle parameterization), and per-axis scale factors $\mathbf{s}_k \in \mathbb{R}^3$. Together, these parameters uniquely describe the model-to-world transformation of each individual primitive. In addition, each primitive contains a payload that describes the appearance and geometry of the associated region in space. The payload is defined by a dense voxel grid $\mathbf{V}_k \in \mathbb{R}^{4 \times M_x \times M_y \times M_z}$ that stores the color (3 channels) and opacity (1 channel) for the $M_x \times M_y \times M_z$ voxels, with M_* being the number of voxels along each spatial dimension. Below, we will assume our volumes are cubes with $M_* = M$ unless stated otherwise.

As mentioned earlier, the volumetric primitives are weakly constrained to the surface of the guide mesh and are allowed to deviate from it if that improves reconstruction quality. Specifically, their position $\mathbf{t}_k = \hat{\mathbf{t}}_k + \boldsymbol{\delta}_{\mathbf{t}_k}$, rotation $\mathbf{R}_k = \boldsymbol{\delta}_{\mathbf{R}_k} \cdot \hat{\mathbf{R}}_k$, and scale $\mathbf{s}_k = \hat{\mathbf{s}}_k + \boldsymbol{\delta}_{\mathbf{s}_k}$ are modeled relative to the guide mesh base transformation $(\hat{\mathbf{t}}_k, \hat{\mathbf{R}}_k, \hat{\mathbf{s}}_k)$ using the regressed values $(\boldsymbol{\delta}_{\mathbf{t}_k}, \boldsymbol{\delta}_{\mathbf{R}_k}, \boldsymbol{\delta}_{\mathbf{s}_k})$. To compute the mesh-based initialization, we generate a 2D grid in the mesh's texture space and generate the primitives at the 3D locations on the mesh that correspond to the *uv*-coordinates of the grid points. The orientation of the primitives is initialized based on the local tangent frame of the 3D surface point they are attached to. The scale of the boxes is initialized based on the local gradient of the *uv*-coordinates at the corresponding grid point position. Thus, the primitives are initialized with a scale in proportion to distances to their neighbours.

3.1.3 Opacity Fade Factor.

Allowing the volumetric primitives to deviate from the guide mesh is important to account for deficiencies in the initialization strategy, low guide mesh quality, and insufficient coverage of objects in the scene. However, allowing for motion is not enough; during training the model can only receive gradients from regions of space that the primitives cover, resulting in a limited ability to self assemble and expand to attain more coverage of the scene's content. Furthermore, it is easier for the model to

reduce opacity in empty regions than to move the primitives away. This wastes primitives that would be better-utilized in regions with higher occupancy. To mitigate this behavior, we apply a windowing function $\mathbf{W} \in \mathbb{R}^{M^3}$ to the opacity of the payload that takes the form:

$$\mathbf{W}(x, y, z) = \exp\left\{-\alpha\left(x^\beta + y^\beta + z^\beta\right)\right\}, \quad (1)$$

where $(x, y, z) \in [-1, 1]^3$ are normalized coordinates within the primitive's payload volume. Here, α and β are hyperparameters that control the rate of opacity decay towards the edges of the volume. This windowing function adds an inductive bias to explain the scene's contents via motion instead of payload since the magnitude of gradients that are propagated through opacity values at the edges of the payload are downscaled. We note that this does not prevent the edges of the opacity payload from being able to take on large values, rather, our construction forces them to learn more slowly [Karras et al. 2020], thus favoring motion of the primitives whose gradients are not similarly impeded. We found $\alpha = 8$ and $\beta = 8$ was a good balance between scene coverage and reconstruction accuracy and keep them fixed for all experiments.

3.1.4 Network Architecture.

We employ an encoder-decoder network to parameterize the coarse tracked proxy mesh as well as the weakly linked mixture of volumetric primitives. Our approach is based on Variational Autoencoders (VAEs) [Kingma and Welling 2013] to encode the dynamics of the scene using a low-dimensional latent code $\mathbf{z} \in \mathbb{R}^{256}$. Note that the goal of our construction is only to produce the decoder. The role of the encoder during training is to encourage a well structured latent space. It can be discarded upon training completion and replaced with an application specific encoder [Wei et al. 2019] or simply with latent space traversal [Abdal et al. 2019]. In the following, we provide details of the encoder for different training settings as well as our four decoder modules.

Encoder. The architecture of our encoder is specialized to the data available for modeling. When coarse mesh tracking is available, we follow the architecture in [Lombardi et al. 2018], which takes as input the tracked geometry and view-averaged unwarped texture for each frame. Geometry is passed through a fully connected layer, and texture through a convolutional branch, before being fused and further processed to predict the parameters of a normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$, where $\boldsymbol{\mu} \in \mathbb{R}^{256}$ is the mean and $\boldsymbol{\sigma} \in \mathbb{R}^{256}$ is the standard deviation. When tracking is not available we follow the encoder architecture in [Lombardi et al. 2019], where images from $K = 1$ fixed view is taken as input. To learn a smooth latent space with good interpolation properties, we regularize using a KL-divergence loss that forces the predicted distribution to stay close to a standard normal distribution. The latent vector $\mathbf{z} \in \mathbb{R}^{256}$ is obtained by sampling from the predicted distribution using the reparameterization trick [Kingma and Welling 2013].

Decoder. Our decoder is comprised of four modules; two geometry decoders and two payload decoders. The geometry decoders determine the primitives' model-to-world transformations. $\mathcal{D}_{\text{mesh}} : \mathbb{R}^{256} \rightarrow \mathbb{R}^{3 \times N_{\text{mesh}}}$ predicts the guide mesh $\mathcal{M}$ used to initialize the transformations. It is comprised of a sequence of fully connected layers. $\mathcal{D}_{\text{pRs}} : \mathbb{R}^{256} \rightarrow \mathbb{R}^{9 \times N_{\text{prim}}}$ is responsible for predicting the deviations in position, rotation (as a Rodrigues vector),
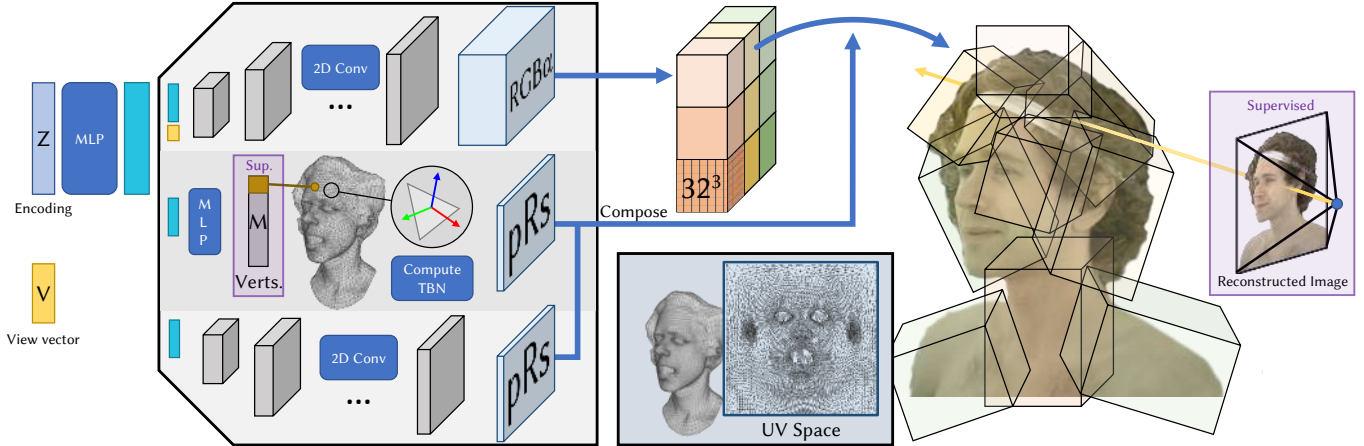
Fig. 2. Decoder architecture. We decode latent code **z** into 1) a *slab* of volumetric primitives, 2) vertex positions of a guide mesh, which is used to compute a base transformation for each primitive $(\hat{\mathbf{t}}_k, \hat{\mathbf{R}}_k, \hat{\mathbf{s}}_k)$ by computing a rotation matrix from the tangent, bitangent, normal vectors at points evenly distributed in the UV space of the mesh, and 3) a residual transformation $(\delta_{\mathbf{t}_k}, \delta_{\mathbf{R}_k}, \delta_{\mathbf{s}_k})$. The 3D slab is divided into individual primitives, and transformed into world space by composing the two transformations. We raymarch through the volumetric primitives, accumulating color and opacity, to form the reconstructed image. Both the reconstructed image and the mesh vertex positions are supervised during training from ground truth data.

and scale $(\delta_{\mathbf{t}_k}, \delta_{\mathbf{R}_k}, \delta_{\mathbf{s}_k})$ from the guide mesh initialization. It uses a 2D convolutional architecture to produces the motion parameters as channels of a 2D grid following the primitive's ordering in the texture's uv-space described in §3.1.2. The payload decoders determine the color and opacity stored in the primitives' voxel grid $\mathbf{V}_k$. $\mathcal{D}_\alpha : \mathbb{R}^{256} \rightarrow \mathbb{R}^{1 \times M^3 \times N_{\text{prim}}}$ computes opacity based on a 2D convolutional architecture. $\mathcal{D}_{\text{rgb}} : \mathbb{R}^{256+3} \rightarrow \mathbb{R}^{3 \times M^3 \times N_{\text{prim}}}$ computes view-dependent RGB color. It is also based on 2D transposed convolutions and uses an object-centric view-vector $\mathbf{d}_{\text{view}} \in \mathbb{R}^3$, i.e., a vector pointing to the center of the object/scene. The view vector allows the decoder to model view-dependent phenomena, like specularities. Unlike the geometry decoders, which employ small networks and are efficient to compute, payload decoders present a significant computational challenge due to the total number of elements they have to generate. Our architecture, shown in Fig. 2, addresses this by avoiding redundant computation through the use of a convolutional architecture. Nearby locations in the output *slab* of $\mathbf{V}_k$'s leverage shared features from earlier layers of the network. This is in contrast to MLP-based methods, such as [Mildenhall et al. 2020], where each position requires independent computation of all features in all layers, without any sharing. Since our texture space is the result of a mesh-atlasing algorithm that tends to preserve the locality structure of the underlying 3D mesh, the regular grid ordering of our payload $\mathbf{V}_k$ within the decoded slab (see §3.1.2) well leverages the spatially coherent structures afforded by devonvolution. The result is an efficient architecture with good reconstruction capacity.

Background Model. MVP is designed to model objects in a scene from an outside-in camera configuration, but the extent of object coverage is not know a-priori. Thus, we need a mechanism for separating objects from the backgrounds in the scene. However, existing segmentation algorithms can fail to capture fine details around object borders and can be inconsistent in 3D. Instead, we jointly model

the objects as well as the scene's background. Whereas the objects are modeled using MVP, we use a separate neural network to model the background as a modulation of images captured of the empty scene with the objects absent. Specifically, our background model for the ith-camera takes the form:

$$\mathcal{B}_i(\mathbf{x}) = \text{softplus}\left\{\bar{\mathcal{B}}_i + F_\theta(\mathbf{c}_i, \mathbf{d}_i(\mathbf{x}))\right\} \ , \tag{2}$$

where $\bar{\mathcal{B}}_i$ is the image of the empty capture space, $\mathbf{c}_i$ is the camera center and $\mathbf{d}_i$ is the ray direction for pixel $\mathbf{x}$. The function F is an MLP with weights θ that takes position-encoded camera coordinates and ray directions and produces a rgb-color using an architecture similar to NeRF [Mildenhall et al. 2020]. The background images of the empty scene are not sufficient by themselves since objects in the scene can have effects on the background, which, if not accounted for, are absorbed in to the MVP resulting in hazy reconstructions as observed in NV [Lombardi et al. 2019]. Examples of these effects include shadowing and content outside of the modeling volume, like supporting stands and chairs. As we will see in §3.2.2, MVP rendering produces an image with color, $\mathcal{I}$, and alpha, $\mathcal{A}$, channels. These are combined with the background image to produce the final output that is compared to the captured images during training through alpha-compositing: $\tilde{\mathcal{I}}_i = \mathcal{A}_i \, \mathcal{I}_i + (1 - \mathcal{A}_i) \, \mathcal{B}_i$.

3.2 Efficient and Differentiable Image Formation

The proposed scene representation is able to focus the representational power of the encoder-decoder network on the occupied regions of 3D space, thus leading to a high resolution model and efficient decoding. However, we still need to be able to efficiently render images using this representation. For this, we propose an approach that combines an efficient raymarching strategy with a differentiable volumetric aggregation scheme.

3.2.1 Efficient Raymarching. To enable efficient rendering, our algorithm should: 1) skip samples in empty space, and 2) employ

efficient payload sampling. Similar to [Lombardi et al. 2019], the regular grid structure of our payload enables efficient sampling via trilinear interpolation. However, in each step of the ray marching algorithm, we additionally need to find within which primitives the current evaluation point lies. These primitives tend to be highly irregular with positions, rotations, and scales that vary on a per-frame basis. For this, we employ a highly optimized data-parallel BVH implementation [Karras and Aila 2013] that requires less than 0.1 ms for 4096 primitives at construction time. This enables us to rebuild the BVH on a per-frame basis, thus handling dynamic scenes, and provides us with efficient intersection tests. Given this data structure of the scene, we propose a strategy for *limiting* evaluations as much as possible. First, we compute and store the primitives that each ray intersects. We use this to compute $(t_{\min}, t_{\max})$, the domain of integration. While marching along a ray between $t_{\min}$ and $t_{\max}$, we check each sample only against the ray-specific list of intersected primitives. Compared to MLP-based methods, e.g., NeRF [Mildenhall et al. 2020], our approach exhibits very fast sampling. If the number of overlapping primitives is kept low, the total sampling cost is much smaller than a deep MLP evaluation at each step, which is far from real-time even with a good importance sampling strategy.

3.2.2 Differentiable Volumetric Aggregation. We require a differentiable image formation model to enable end-to-end training based on multi-view imagery. Given the sample points in occupied space extracted by the efficient ray marching strategy, we employ an accumulative volume rendering scheme as in [Lombardi et al. 2019] that is motivated by front-to-back additive alpha blending. During this process, the ray accumulates color as well as opacity. Given a ray $r_{\mathbf{p}}(t) = \mathbf{o_p} + t\mathbf{d_p}$ with starting position $\mathbf{o_p}$ and ray direction $\mathbf{d_p}$, we solve the following integral using numerical quadrature:

$$\mathcal{I}_p = \int_{t_{\min}}^{t_{\max}} \mathbf{V}_{\text{col}}(r_{\mathbf{p}}(t)) \cdot \frac{dT(t)}{dt} \cdot dt \ .$$

$$T(t) = \min\left(\int_{t_{\min}}^{t} \mathbf{V}_{\alpha}(r_{\mathbf{p}}(t)) \cdot dt, 1\right) \ .$$

Here, $\mathbf{V}_{\text{col}}$ and $\mathbf{V}_{\alpha}$ are the global color and opacity field computed based on the current instantiation of the volumetric primitives. We set the alpha value associated with the pixel to $\mathcal{A}_p = T(t_{\max})$. For high performance rendering, we employ an early stopping strategy based on the accumulated opacity, i.e., if the accumulated opacity is larger than $1.0 - \epsilon_{\text{early}}$ we terminate ray marching, since the rest of the sample points do not have a significant impact on the final pixel color. If a sample point is contained within multiple volumetric primitives, we combine their values in their BVH order based on the accumulation scheme. Our use of the additive formulation for integration, as opposed to the multiplicative form [Mildenhall et al. 2020], is motivated by its independence to ordering up to the saturation point. This allows for a backward pass implementation that is more memory efficient, since we do not need to keep the full $O(n^2)$ graph of operations. Thus, our implementation requires less memory and allows for larger batch sizes during training. For more details, we refer to the supplemental document.

3.3 End-to-end Training

Next, we discuss how we can train our approach end-to-end based on a set of 2D multi-view images. The trainable parameters of our model are Θ. Given a multi-view video sequence $\{\mathcal{I}^{(i)}\}_{i=1}^{N_{img}}$ with $N_{\text{img}} = N_{\text{frames}} \cdot N_{\text{cams}}$ training images, our goal is to find the optimal parameters Θ^* that best explain the training data. To this end, we solve the following optimization problem:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=0}^{N_{\text{img}}-1} \sum_{p=0}^{N_{\text{pixels}}-1} \mathcal{L}(\Theta; \mathcal{I}_p^{(i)}) \ .$$

We employ ADAM [Kingma and Ba 2015] to solve this optimization problem based on stochastic mini-batch optimization. In each iteration, our training strategy uniformly samples rays from each image in the current batch to define the loss function. We employ a learning rate $lr = 0.0001$ and all other parameters are set to their default values. Our training objective is of the following from:

$$\mathcal{L}(\Theta; \mathcal{I}_p) = \mathcal{L}_{\text{pho}}(\Theta; \mathcal{I}_p) + \mathcal{L}_{\text{geo}}(\Theta) + \mathcal{L}_{\text{vol}}(\Theta) + \mathcal{L}_{\text{del}}(\Theta) + \mathcal{L}_{\text{kld}}(\Theta) \ .$$

It consists of a photometric reconstruction loss $\mathcal{L}_{\text{pho}}$, a coarse geometry reconstruction loss $\mathcal{L}_{\text{geo}}$, a volume minimization prior $\mathcal{L}_{\text{vol}}$, a delta magnitude prior $\mathcal{L}_{\text{del}}$, and a Kullback–Leibler (KL) divergence prior $\mathcal{L}_{\text{kld}}$ to regularize the latent space of our Variational Autoencoder (VAE) [Kingma and Welling 2013]. In the following, we provide more details on the individual energy terms.

Photometric Reconstruction Loss. We want to enforce that the synthesized images look photo-realistic and match the ground truth. To this end, we compare the synthesized pixels $\bar{\mathcal{I}}_p(\Theta)$ to the ground truth $\mathcal{I}_p$ using the following loss function:

$$\mathcal{L}_{\text{pho}} = \lambda_{\text{pho}} \frac{1}{N_{\mathcal{P}}} \sum_{p \in \mathcal{P}} \left\| \mathcal{I}_p - \bar{\mathcal{I}}_p(\Theta) \right\|_2^2 \ .$$

Here, $\mathcal{P}$ is the set of sampled pixels and w_p is a per-pixel weight. We set a relative weight of $\lambda_{\text{photo}} = 1.0$.

Mesh Reconstruction Loss. We also want to enforce that the coarse mesh proxy follows the motion in the scene. To this end, we compare the regressed vertex positions to the available ground truth traced mesh using the following loss function:

$$\mathcal{L}_{\text{geo}} = \lambda_{\text{geo}} \frac{1}{N_{\text{mesh}}} \sum_{i=0}^{N_{\text{mesh}}} \left\| \mathbf{v}_i - \bar{\mathbf{v}}_i(\Theta) \right\|_2^2 \ .$$

Here, we employ an ℓ_2-loss function, $\mathbf{v}_i$ is the ground truth position of the tracked mesh, and $\bar{\mathbf{v}}_i(\Theta)$ is the corresponding regressed vertex position. We employ the coarse mesh-based tracking used in the approach of [Lombardi et al. 2018]. The mesh reconstruction loss pulls the volumetric primitives, which are weakly linked to it, to an approximately correct position. Note, the primitives are only weakly linked to the mesh proxy and can deviate from their initial positions if that improves the photometric reconstruction loss. We set a relative weight of $\lambda_{\text{geo}} = 0.1$.

Volume Minimization Prior. We constrain the volumetric primitives to be as small as possible. The reasons for this are twofold: 1) We want to prevent them from overlapping too much, since this wastes model capacity in already well explained regions, and 2)

Fig. 3. Novel view synthesis example. Our novel approach enables high fidelity novel view synthesis of a complete, dynamic, upper body at real-time rates.

Fig. 4. Latent space interpolation example. Our approach enables animating the reconstructions via latent-space interpolation between two keyframes. Even for extremely challenging facial expressions the interpolated results are highly realistic. The output of our motion model, which produces rotations, translations, and scales, is effectively a forward warp. Such a warp can be sensibly-interpolated whereas inverse warps cannot.

We want to prevent loss of resolution by large primitives overlapping empty space. To this end, we employ the following volume minimization prior:

$$\mathcal{L}_{\text{vol}} = \lambda_{\text{vol}} \sum_{i=1}^{N_{\text{prim}}} \text{Prod}(\mathbf{s}_i) \ .$$

Here, $\mathbf{s}_i = \hat{\mathbf{s}}_i + \boldsymbol{\delta}_{\mathbf{s}_i}$ is the vector of side lengths of the primitive and $\text{Prod}(\bullet)$ is the product of the values of a vector, e.g., in our case the volume of a primitive. We minimize the total volume with a relative weight of $\lambda_{\text{vol}} = 0.01$.

4 RESULTS

In this section we describe our datasets for training and evaluation, present results on several challenging sequences, perform ablation studies over our model's components, and compare to the state of the art. We perform both qualitative and quantitative evaluations.

4.1 Training Data

We evaluate our approach on a large number of sequences captured using a spherically arranged multi-camera capture system with ≈ 100 synchronized color cameras. Each dataset contains roughly 25,000 frames from each camera. The cameras record with a resolution of 2668×4096 at 30Hz, and are equally distributed on the surface of the spherical structure with a radius of 1.2 meters. They are geometrically calibrated with respect to each other with the intrinsic and extrinsic parameters of a pinhole camera model. For training and evaluation, we downsample the images to a resolution of 667×1024 to reduce the time it takes to load images from disk,

keeping images from all but 8 cameras for training, with the remainder used for testing. To handle the different radiometric properties of the cameras, e.g., color response and white balance, we employ per-camera color calibration based on 6 parameters (gain and bias per color channel) similar to [Lombardi et al. 2018], but pre-trained for all cameras once for each dataset. We train each scene for 500,000 iterations, which takes roughly 5 days on a single NVIDIA Tesla V100.

4.2 Qualitative Results

Our approach achieves a high level of fidelity while matching the completeness of volumetric representations, e.g., hair coverage and inner mouth, see Fig. 3, but with an efficiency closer to mesh-based approaches. Our fastest model is able to render binocular stereo views at a resolution of 896×832 at 40Hz, which enables live visualization of our results in a virtual reality setting; please see the supplemental video for these results. Furthermore, our model can represent dynamic scenes, supporting free view-point video applications, see Fig. 3. Our model also enables animation, which we demonstrate via latent-space interpolation, see Fig. 4. Due to the combination of the variational architecture with a forward warp, our approach produces highly realistic animation results even if the facial expressions in the keyframes are extremely different.

4.3 Ablation Studies

We perform a number of ablation studies to support each of our design choices.

| (a) GT | (b) NV | (c) 8 prim. | (d) 512 prim. | (e) 32k prim. | (f) 256 prim.* |

Fig. 5. We evaluate results by comparing models with a varying number of primitives on held out views while keeping the total number of voxels constant (2 million). From left to right: ground truth, neural volumes, 8 primitives, 512 primitives, and 32768 primitives. We find that hundreds of primitives gives the best balance of quality and performance, with higher numbers of primitives exhibiting fine detail, but struggling with elaborate motion, e.g., mouth interior. The last column shows our final model, which uses 256 primitives and 8 million total voxels. (*): only 2D transposed convolutions.

Table 1. Quantitative evaluation of number of volumetric primitives. We evaluate quality (MSE, PSNR, SSIM, LPIPS) and execution time (decode time, raymarch time, and total time in milliseconds) on held out views. Quality generally increases as more primitives are used, as they can more tightly fit the geometry of the surface. The performance drops for extreme numbers of primitives, since this increases the difficulty of the learning problem. Raymarching becomes more costly as more primitives are used, as the overhead of checking if a sample point lies in each primitive dominates the sampling time. Decode time decreases with number of primitives (note the number of voxels remains constant at 2 million) as the "slab" has less thickness and therefore requires fewer 3D convolutions, which tend to be more expensive. The rightmost two columns show our final optimized model, which uses 256 primitives, 8 million voxels, and 2D convolutions exclusively. (*): only 2D convolutions, and (**): smaller raymarching step size.

	NV	8 prim.	64 prim.	512 prim.	4k prim.	32k prim.	262k prim.	256 prim.*	256 prim.**
MSE (↓)	49.1535	46.5067	42.2567	41.9601	40.3211	40.6524	43.4307	39.9031	**37.0805**
PSNR (↑)	31.2153	31.4556	31.8718	31.9024	32.0755	32.0399	31.7528	32.1207	**32.4393**
SSIM (↑)	0.9293	0.9301	0.9336	0.9333	0.9352	0.9349	0.9308	0.9344	**0.9393**
LPIPS (↓)	0.2822	0.3151	0.2879	0.2764	0.2755	0.2670	0.2767	0.2921	**0.2484**
decode (↓)	54.8993	55.3351	57.5364	54.0634	39.3384	39.6899	26.4612	20.5428	**18.3311**
raymarch (↓)	**7.0539**	7.3450	8.7660	10.6198	13.3397	19.8951	35.8098	11.0970	36.6212
total (↓)	61.9532	62.6801	66.3024	64.6831	52.6781	59.5850	62.2710	**31.6398**	54.9523

Number of Primitives. We investigated the influence the number of primitives, N_{prim}, has on rendering quality and runtime performance. A quantitative evaluation can be found in Tab. 1. Here, we compared models with varying number of primitives on held out views, while keeping the total number of voxels constant (∼ 2 million). In total, we compared models with 1, 8, 64, 512, 4096, 32768, and 262144 primitives. Note that the special case of exactly one volumetric primitive corresponds to the setting used in Lombardi et al. [2019]. Our best model uses 256 primitives and 8 million voxels. Fig. 6 shows this data in a scatter plot with PSNR on the x-axis and total execution time on the y-axis. Fig. 5 shows the qualitative results for this evaluation. As the results show, if the number of primitives is too low results appear blurry, and if the number of primitives is too large the model struggles to model elaborate motion, e.g., in the mouth region, leading to artifacts.

Primitive Volume Prior. The influence of our primitive volume prior, $\mathcal{L}_{vol}$ in Section 3.3, is evaluated by training models with different weights, $\lambda_{vol} \in \{0.001, 0.01, 0.1\}$. We used models with 512 and 32,768 primitives in this evaluation (see Fig. 7 and Tab. 2). Larger weights lead to smaller scales and reduced overlap between adjacent primitives. This, in turn, leads to faster runtime performance since less overlap means that there are fewer primitives to evaluate at each marching step, as well as having less overall volume coverage. However, prior weights that are too large can lead to over shrinking, where holes start to appear in the reconstruction and image evidence is not sufficient to force them to expand.

Importance of the Opacity Fade Factor. We trained a model with 512 primitives with and without the opacity fade factor, $\mathbf{W}$, described in Section 3.1.3. As shown in Fig. 8, opacity fade is critical for the primitives to converge to good configurations, as it allows gradients to properly flow from image to primitive position, rotation, and scaling. Without opacity fade, the gradients do not account for the impact of movement in the image plane at primitive silhouette edges. This results in suboptimal box configurations with large overlaps and coverage of empty space. See the supplemental document for a quantitative comparison of the opacity fade factor.
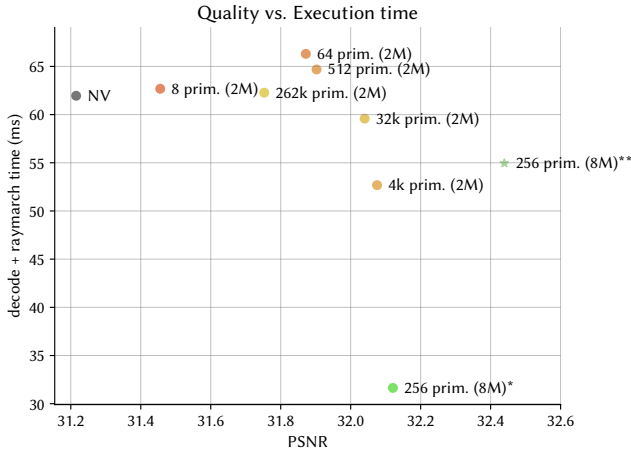
Fig. 6. Quality vs. Execution time for varying number of primitives and model architectures. We show our base model (8-32k primitives with 2 million total voxels) and our optimized model (256 primitives with 8 million total voxels) compared to Neural Volumes. Our optimized model greatly improved performance over Neural Volumes in both quality (PSNR) and total execution time.

Table 2. Ablation of primitive volume prior. The volume prior provides a trade-off between quality and performance, with high values causing primitives to shrink, therefore making raymarching more efficient, at the cost of quality.

	512 prim. $\lambda_{vol} = 0.001$	512 prim. $\lambda_{vol} = 0.01$	512 prim. $\lambda_{vol} = 0.1$
MSE ($\downarrow$)	42.3944	**41.9601**	51.9607
PSNR ($\uparrow$)	31.8577	**31.9024**	30.9740
SSIM ($\uparrow$)	**0.9383**	0.9333	0.9253
LPIPS ($\downarrow$)	0.4119	**0.2764**	0.4928
decode ($\downarrow$)	55.0023	**54.0634**	54.3226
raymarch ($\downarrow$)	16.1322	10.6198	**7.8910**
total ($\downarrow$)	71.1345	64.6831	**62.2137**
	32k prim. $\lambda_{vol} = 0.001$	32k prim. $\lambda_{vol} = 0.01$	32k prim. $\lambda_{vol} = 0.1$
MSE ($\downarrow$)	41.9728	**40.6524**	48.9140
PSNR ($\uparrow$)	31.9011	**32.0399**	31.2365
SSIM ($\uparrow$)	**0.9357**	0.9349	0.9238
LPIPS ($\downarrow$)	0.3810	**0.2670**	0.4708
decode ($\downarrow$)	**35.8994**	39.6899	39.1676
raymarch ($\downarrow$)	48.3455	19.8951	**9.6950**
total ($\downarrow$)	84.2449	59.5850	**48.8626**

(a) $\lambda_{vol} = 0.001$ (b) $\lambda_{vol} = 0.01$ (c) $\lambda_{vol} = 0.1$
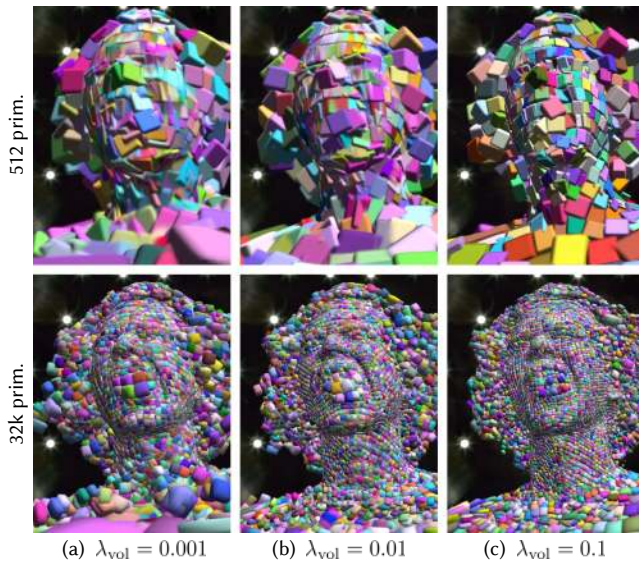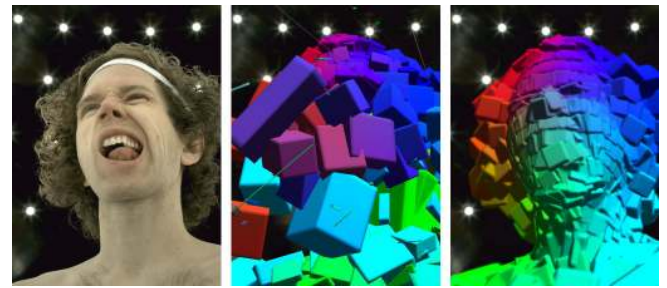
Fig. 7. We evaluate different strengths of the primitive volume prior for 512 primitives and 32768 primitives. A stronger primitive volume prior leads to less overlap and thus speeds up raymarching. If the prior is too strong, however, holes appear and the reconstruction error increases.

(a) GT   (b) No alpha fade   (c) With alpha fade

Fig. 8. The opacity fade factor enables proper learning of primitive positions. We evaluate with and without opacity fade factor applied to the primitives. No opacity fade factor causes poor primitive positioning, drifting very far from the initialization, to a bad configuration with large primitives and high overlap because of poor gradients.
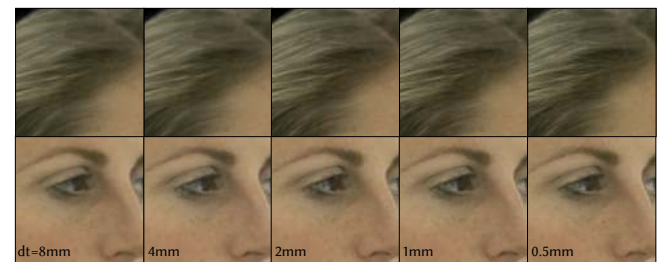

Fig. 9. Here we show single-frame models of 16M voxels. Each model is trained and tested at the stepsize shown in the insets. Small step sizes are able to recover more detail, and reduce noise associated with raymarching at the cost of increased raymarching time.

Impact of Voxel Count and Raymarching Step Size. Fig. 9 and Fig. 10 illustrate the effects of different voxel counts and raymarching step sizes on perceived resolution (see the supplemental document for the quantitative impact of step size). Here, the same step size is used both during training and evaluation. Smaller marching step sizes recover more detail, such as hair and wrinkles, and result in lower reconstruction error on held out views. Likewise, more voxels provide sharper results. These gains in accuracy are, however, attained at the cost of performance. Decoding scales linearly with

the number of voxels decoded, and raymarching scales linearly with step size.
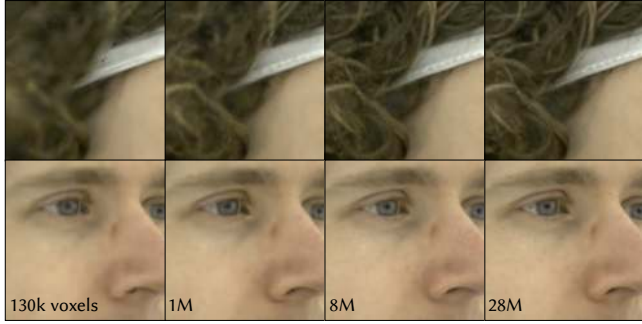
Fig. 10. Effect of varying the number of voxels. Here we show a 256 primitive model at a stepsize of 1mm, trained with 8^3, 16^3, 32^3, and 48^3 voxels per primitive, yielding the total number of voxels shown in the insets. Voxel models of more than 8M voxels are generally too slow to run in realtime on current hardware.

Motion Model Architecture. The motion model, $\mathcal{D}_{pRs}$, regresses position, rotation, and scale deviations of the volumetric primitive from the underlying guide mesh; $\{\boldsymbol{\delta}_{t_k}, \boldsymbol{\delta}_{R_k}, \boldsymbol{\delta}_{\mathbf{s}_k}\}_{k=1}^{N_{prim}}$. We compared the convolutional architecture we proposed in Section 3 against a simpler linear layer that transforms the latent code $\mathbf{z}$ to a $9N_{prim}$-dimensional vector, comprising the stacked motion vectors, i.e., three dimensions each for translation, rotation and scale. Tab. 3 shows that our convolutional architecture outperforms the simple linear model for almost all primitive counts, N_{prim}. A visualization of their differences can be seen in Fig. 11, where our convolutional model produces primitive configurations that follow surfaces in the scene more closely than the linear model, which produces many more "fly away" zero-opacity primitives, wasting resolution.

4.4 Comparisons

We compared MVP against the current state of the art in neural rendering for both static and dynamic scenes. As our qualitative and quantitative results show, we outperform the current state of the art in reconstruction quality as well as runtime performance.

Neural Volumes. We compare to Neural Volumes (NV) [Lombardi et al. 2019] on several challenging dynamic sequences, see Fig. 12. Our approach obtains sharper and more detailed reconstructions, while being much faster to render. We attribute this to our novel mixture of volumetric primitives that can concentrate representation resolution and compute in occupied regions in the scene and skip empty space during raymarching. Quantitative comparisons are presented in Tab. 1. As can be seen, we outperform Neural Volumes (NV) [Lombardi et al. 2019] in terms of SSIM and LPIPS.

Neural Radiance Fields (NeRF). We also compare our approach to a Neural Radiance Field (NeRF) [Mildenhall et al. 2020], see Fig. 13. Since NeRF is an approach for novel view synthesis of static scenes, we trained it using only a single frame of our capture sequence. We compared it against MVP trained on both the same frame and the entire sequence of approximately 20,000-frame. A visualization of the differences between NeRF and MVP on the static frame is shown in Fig. 13. NeRF excels at representing geometric detail, as can be seen in the teeth, but struggles with planar texture detail, like the

texture on the lips or eyebrows. MVP captures both geometric and texture details well. Quantitative results comparing the methods is presented in Tab. 4, where our method outperforms NeRF on all metrics, even when trained using multiple frames. Finally, our approach improves over NeRF in runtime performance by three orders of magnitude.

5 LIMITATIONS

We have demonstrated high quality neural rendering results for dynamic scenes. Nevertheless, our approach is subject to a few limitations that can be addressed in follow-up work. First, we require a coarse tracked mesh to initialize the positions, rotations, and scale of the volumetric primitives. In the future, we'd like to obviate this requirement and allow the primitives to self-organize based only on the camera images. Second, the approach requires a high-end computer and graphics card to achieve real-time performance. One reason for this is the often high overlap between adjacent volumetric primitives. Thus, we have to perform multiple trilinear interpolations per sample point, which negatively impacts runtime. It would be interesting to incorporate regularization strategies to minimize overlap, which could lead to a significant performance boost. Finally, the number of employed primitives is predefined and has to be empirically determined for each scene type. It is an interesting direction for future work to incorporate this selection process into the optimization, such that the best setting can be automatically determined. Despite these limitations, we believe that our approach is already a significant step forward for real-time neural rendering of dynamic scenes at high resolutions.

6 CONCLUSION

We have presented a novel 3D neural scene representation that handles dynamic scenes, is fast to render, drivable, and can represent 3D space at high resolution. At the core of our scene representation is a novel mixture of volumetric primitives that is regressed by an encoder-decoder network. We train our representation based on a combination of 2D and 3D supervision. Our approach generalizes volumetric and primitive-based paradigms under a unified representation and combines their advantages, thus leading to high performance decoding and efficient rendering of dynamic scenes. As our comparisons demonstrate, we obtain higher quality results than the current state of the art. We hope that our approach will be a stepping stone towards highly efficient neural rendering approaches for dynamic scenes and that it will inspire follow-up work.

REFERENCES

Rameen Abdal, Yipeng Qin, and Peter Wonka. 2019. Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. 2019. Neural point-based graphics. *arXiv preprint arXiv:1906.08240* (2019).

Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. 2020. MatryODShka: Real-time 6DoF Video View Synthesis using Multi-Sphere Images. *arXiv preprint arXiv:2008.06534* (2020).

Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. 2020. Immersive Light Field Video with a Layered Mesh Representation. *ACM Trans. Graph.* 39, 4, Article 86 (July 2020), 15 pages. https://doi.org/10.1145/3386569.3392485

Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. 2020. Deep Local Shapes: Learning Local SDF

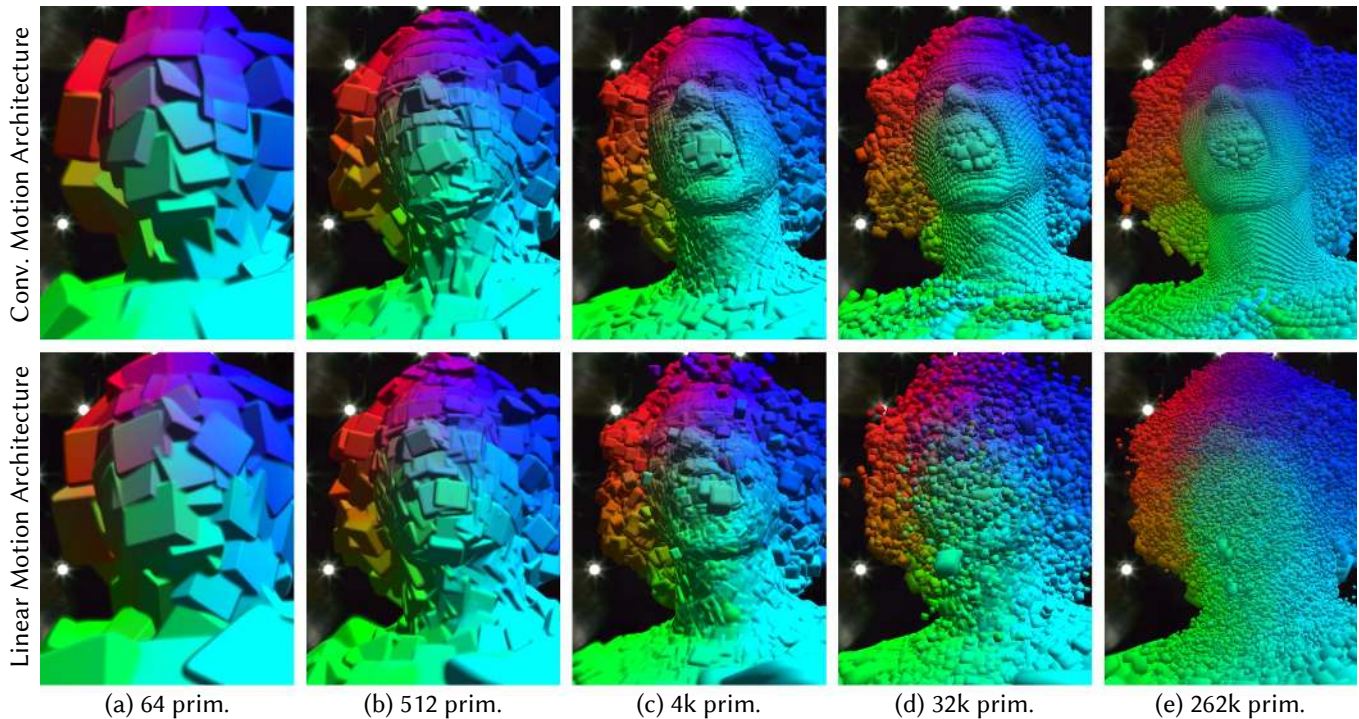| | (a) 64 prim. | (b) 512 prim. | (c) 4k prim. | (d) 32k prim. | (e) 262k prim. |

Fig. 11. We evaluate two different motion models for the primitives. One uses a stack of convolutions, where each output pixel contains 9 channels representing the scaling, rotation, and translation of the corresponding primitive. The other uses a linear layer from the encoding to produce $9N_{boxes}$ channels that encode the scale, rotation, and translation of all primitives. The convolutional motion model produces boxes that closely follow the underlying surface and, as shown in Tab. 3, results in better reconstructions.

Table 3. Quantitative evaluation of convolutional motion model vs. linear motion model. The convolutional architecture for producing primitive location, orientation, and size results in primitives that are more coherent with the mesh surface, which allows it to achieve higher quality and better performance than a linear model.

	64 prim. conv	64 prim linear	512 prim. conv	512 prim. linear	4k prim. conv	4k prim. linear	32k prim. conv	32k prim. linear	262k prim. conv	262k prim. linear
MSE (↓)	42.2567	43.7077	41.9601	44.7273	**40.3211**	41.3797	40.6524	42.5864	43.4307	50.4797
PSNR (↑)	31.8718	31.7252	31.9024	31.6251	**32.0755**	31.9629	32.0399	31.8381	31.7528	31.0996
SSIM (↑)	0.9336	0.9336	0.9333	0.9344	0.9352	**0.9354**	0.9349	0.9311	0.9308	0.9248
LPIPS (↓)	0.2879	0.2953	0.2764	0.2845	0.2755	0.2890	**0.2670**	0.2877	0.2767	0.3059
decode (↓)	57.5364	56.9789	54.0634	53.6730	39.3384	37.9490	39.6899	36.5781	26.4612	**25.0964**
raymarch (↓)	**8.7660**	9.1570	10.6198	14.7531	13.3397	32.1545	19.8951	72.9552	35.8098	184.4986
total (↓)	66.3024	66.1359	64.6831	68.4260	**52.6781**	70.1036	59.5850	109.5333	62.2710	209.5950

Priors for Detailed 3D Reconstruction. *arXiv preprint arXiv:2003.10983* (2020).

Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. 2019. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems*. 9609–9619.

Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 2016. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*. Springer, 628–644.

Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. 2020. Neural Radiance Flow for 4D View Synthesis and Video Processing. *arXiv preprint arXiv:2012.09790* (2020).

Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2020. Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction. *https://arxiv.org/abs/2012.03065* (2020).

Chen Gao, Yichang Shih, Wei-Sheng Lai, Chia-Kai Liang, and Jia-Bin Huang. 2020. Portrait Neural Radiance Fields from a Single Image. *https://arxiv.org/abs/2012.05903* (2020).

Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T Freeman. 2018. Unsupervised training for 3d morphable model regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8377–8386.

Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. 2020. Local Deep Implicit Functions for 3D Shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4857–4866.

Eldar Insafutdinov and Alexey Dosovitskiy. 2018. Unsupervised Learning of Shape and Pose with Differentiable Point Clouds. In *Advances in Neural Information Processing Systems 31 (NIPS 2018)*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Montréal, Canada, 2804–2814.

Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. 2020. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6001–6010.

Abhishek Kar, Christian Häne, and Jitendra Malik. 2017. Learning a multi-view stereo machine. In *Advances in neural information processing systems*. 365–376.

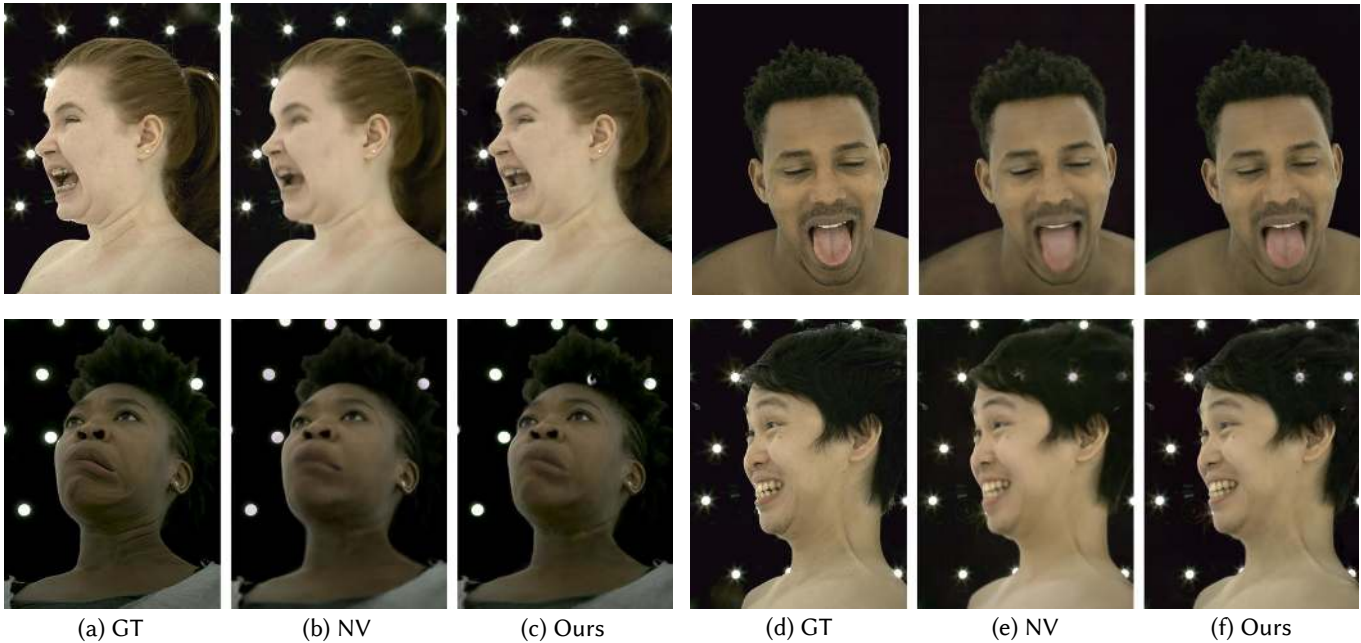(a) GT   (b) NV   (c) Ours   (d) GT   (e) NV   (f) Ours

Fig. 12. We compare with Neural Volumes for several additional subjects. In all cases, our method can better reproduce facial expressions with more accuracy and finer detail.

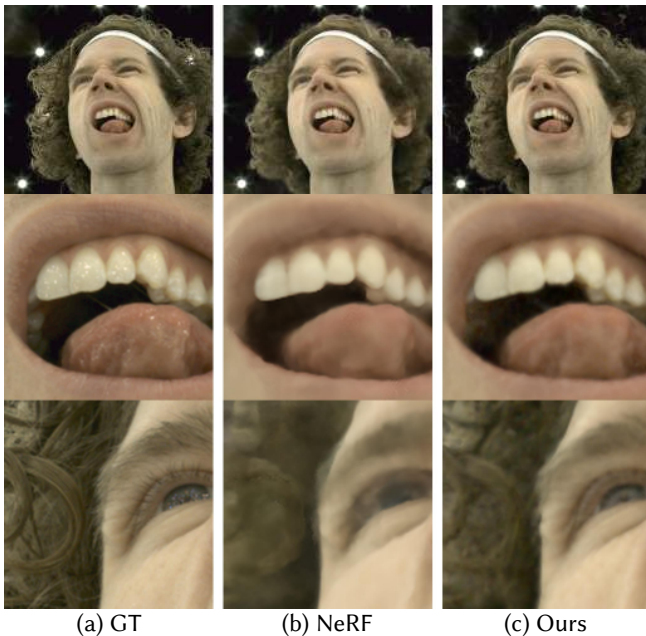(a) GT   (b) NeRF   (c) Ours

Fig. 13. Comparison to Neural Radiance Fields (NeRF) [Mildenhall et al. 2019]. NeRF is a novel view synthesis approach for static scenes. We compare by training both models for a single time step. NeRF excels at representing geometric detail, as can be seen in the teeth, but struggles with planar texture detail. Our approach obtains more detailed results, while being orders of magnitude faster to render.

Table 4. Comparison of our approach to NeRF trained on both a single frame and a sequence (here evaluated only on a static frame). Our approach improves over NeRF in performance by 1000x, and improves quality even when trained on a long sequence.

	NeRF	Ours (single frame)	Ours (multi-frame)
MSE (↓)	47.6849	**37.9261**	38.8223
PSNR (↑)	31.3470	**32.3414**	32.2400
SSIM (↑)	0.9279	**0.9381**	0.9359
LPIPS (↓)	0.3018	**0.2080**	0.2529
decode (↓)	-	**18.8364**	19.4213
raymarch (↓)	-	**31.5097**	37.1573
total (↓)	27910.0000	**50.3462**	56.5786

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and Improving the Image Quality of StyleGAN. In *Proc. CVPR*.

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3d mesh renderer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

Maria Kolos, Artem Sevastopolsky, and Victor Lempitsky. 2020. TRANSPR: Transparency Ray-Accumulating Neural 3D Scene Point Renderer. arXiv:2009.02819

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Transactions on Graphics* 39, 6 (2020).

Christoph Lassner and Michael Zollhöfer. 2020. Pulsar: Efficient Sphere-based Neural Rendering. arXiv:2004.07484 [cs.GR]

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. 2020. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *https://arxiv.org/abs/2011.13084* (2020).

Chen-Hsuan Lin, Chen Kong, and Simon Lucey. 2018. Learning efficient point cloud generation for dense 3d object reconstruction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Tero Karras and Timo Aila. 2013. Fast Parallel Construction of High-Quality Bounding Volume Hierarchies. In *Proceedings of the 5th High-Performance Graphics Conference* (Anaheim, California) *(HPG '13)*. Association for Computing Machinery, New York, NY, USA, 89–99. https://doi.org/10.1145/2492045.2492055

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural Sparse Voxel Fields. arXiv:2007.11571

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*. 7708–7717.

Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. 2018. Deep Appearance Models for Face Rendering. *ACM Trans. Graph.* 37, 4, Article 68 (July 2018), 13 pages. https://doi.org/10.1145/3197517.3201401

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

Matthew M Loper and Michael J Black. 2014. OpenDR: An approximate differentiable renderer. In *European Conference on Computer Vision*. Springer, 154–169.

Feng Liu Xiaoming Liu Luan Tran. 2019. Towards High-fidelity Nonlinear 3D Face Morphoable Model. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

Ricardo Martin-Brualla, Noha Radwan, Mehdi Sajjadi, Jonathan Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2020. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. *https://arxiv.org/abs/2008.02268* (2020).

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. 2019. Neural rerendering in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6878–6887.

Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. 2019. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *ACM Transactions on Graphics (TOG)* (2019).

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934* (2020).

Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L. Davidson, Sameh Khamis, Mingsong Dou, Vladimir Tankovich, Charles Loop, Qin Cai, Philip A. Chou, Sarah Mennicken, Julien Valentin, Vivek Pradeep, Shenlong Wang, Sing Bing Kang, Pushmeet Kohli, Yuliya Lutchyn, Cem Keskin, and Shahram Izadi. 2016. Holoportation: Virtual 3D Teleportation in Real-Time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) (UIST '16). Association for Computing Machinery, New York, NY, USA, 741–754.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 165–174.

Keunhong Park, Utkarsh Sinha, Jonathan Barron, Sofien Bouaziz, Dan Goldman, Steven Seitz, and Ricardo Martin-Brualla. 2020. Deformable Neural Radiance Fields. *https://arxiv.org/abs/2011.12948* (2020).

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. 2020. Convolutional Occupancy Networks. In *European Conference on Computer Vision (ECCV)*.

Felix Petersen, Amit H Bermano, Oliver Deussen, and Daniel Cohen-Or. 2019. Pix2vex: Image-to-geometry reconstruction using a smooth differentiable renderer. *arXiv preprint arXiv:1903.11149* (2019).

Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. 2020. PyTorch3D. https://github.com/facebookresearch/pytorch3d.

Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. 2020. DeRF: Decomposed Radiance Fields. *https://arxiv.org/abs/2011.12490* (2020).

O. Ronneberger, P.Fischer, and T. Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI) (LNCS, Vol. 9351)*. Springer, 234–241.

Riccardo Roveri, Lukas Rahmann, Cengiz Oztireli, and Markus Gross. 2018. A network architecture for point cloud classification via automatic depth images generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4176–4184.

Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. 2019a. PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization. *https://arxiv.org/abs/1905.05172* (2019).

Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. 2019b. PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization. In *Proceedings of the IEEE International Conference on Computer Vision*. 2304–2314.

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on 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)*.

Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. 2020. GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis. arXiv:2007.02442 [cs.CV]

V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhöfer. 2019a. DeepVoxels: Learning Persistent 3D Feature Embeddings. In *Proceedings of Computer Vision and Pattern Recognition (CVPR 2019)*.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. 2019b. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*. 1121–1132.

Pratul Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan Barron. 2020. NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis. *https://arxiv.org/abs/2012.03927* (2020).

Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. 2019. Pushing the Boundaries of View Extrapolation With Multiplane Images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 175–184.

Matthew Tancik, Ben Mildenhall, Terrence Wang, Divi Schmidt, Pratul Srinivasan, Jonathan Barron, and Ren Ng. 2020. Learned Initializations for Optimizing Coordinate-Based Neural Representations. *https://arxiv.org/abs/2012.02189* (2020).

A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. 2020. State of the Art on Neural Rendering. *Computer Graphics Forum (EG STAR 2020)* (2020).

Ayush Tewari, Michael Zollhöfer, Pablo Garrido, Florian Bernard, Hyeongwoo Kim, Patrick Pérez, and Christian Theobalt. 2018. Self-supervised Multi-level Face Model Learning for Monocular Reconstruction at over 250 Hz. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred Neural Rendering: Image Synthesis using Neural Textures. *ACM Transactions on Graphics 2019 (TOG)* (2019).

Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. 2020. Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Deforming Scene from Monocular Video. arXiv:2012.12247 [cs.CV]

Alex Trevithick and Bo Yang. 2020. GRF: Learning a General Radiance Field for 3D Scene Representation and Rendering. *https://arxiv.org/abs/2010.04595* (2020).

Richard Tucker and Noah Snavely. 2020. Single-view View Synthesis with Multiplane Images. In *The IEEE Conference on 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2626–2634.

Julien Valentin, Cem Keskin, Pavel Pidlypenskyi, Ameesh Makadia, Avneesh Sud, and Sofien Bouaziz. 2019. TensorFlow Graphics: Computer Graphics Meets Deep Learning.

Shih-En Wei, Jason Saragih, Tomas Simon, Adam W. Harley, Stephen Lombardi, Michal Perdoch, Alexander Hypes, Dawei Wang, Hernan Badino, and Yaser Sheikh. 2019. VR Facial Animation via Multiview Image Translation. *ACM Trans. Graph.* 38, 4, Article 67 (July 2019), 16 pages.

Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. 2020. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7467–7477.

Chenglei Wu, Takaaki Shiratori, and Yaser Sheikh. 2018. Deep Incremental Learning for Efficient High-Fidelity Face Tracking. *ACM Trans. Graph.* 37, 6, Article 234 (Dec. 2018), 12 pages. https://doi.org/10.1145/3272127.3275101

Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*. 82–90.

Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. 2020. Space-time Neural Irradiance Fields for Free-Viewpoint Video. *https://arxiv.org/abs/2011.12950* (2020).

Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. 2019. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. 2020. pixelNeRF: Neural Radiance Fields from One or Few Images. *https://arxiv.org/abs/2012.02190* (2020).

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817* (2018).