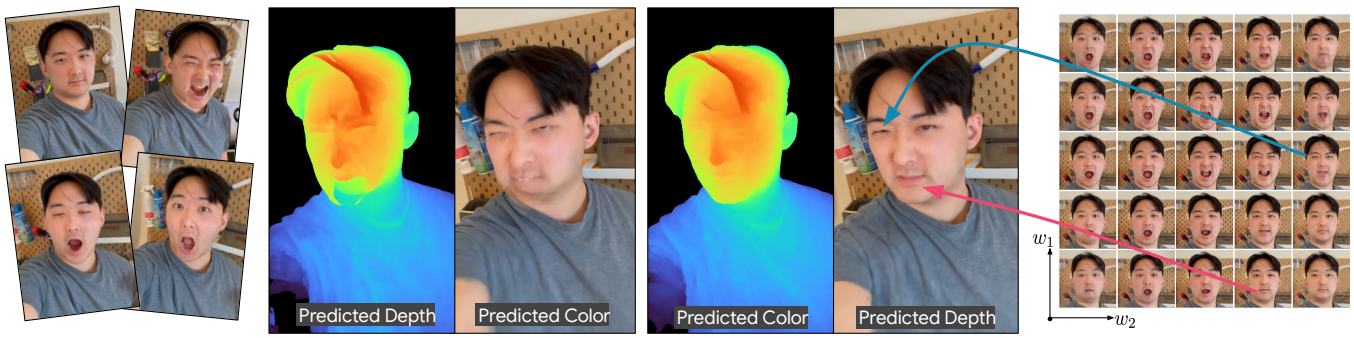
HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields

KEUNHONG PARK*, University of Washington, USA
UTKARSH SINHA, Google Research, USA
PETER HEDMAN, Google Research, USA
JONATHAN T. BARRON†, Google Research, USA
SOFIEN BOUAZIZ†‡, Facebook Reality Labs, USA
DAN B GOLDMAN†, Google Research, USA
RICARDO MARTIN-BRUALLA†, Google Research, USA
STEVEN M. SEITZ†, University of Washington, Google Research, USA



(a) Input Images (b) Nerfies $(\mathbf{c}, \sigma) = F(x, y, z)$ (c) HyperNeRF $(\mathbf{c}, \sigma) = F(x, y, z, w_1, w_2)$ (d) Hyper-Space

Fig. 1. Neural Radiance Fields (NeRF) [Mildenhall et al. 2020] when endowed with the ability to handle deformations [Park et al. 2020] are able to capture non-static human subjects, but often struggle in the presence of significant deformation or topological variation, as evidenced in (b). By modeling a family of shapes in a high dimensional space shown in (d), our Hyper-NeRF model is able to handle topological variation and thereby produce more realistic renderings and more accurate geometric reconstructions, as can be seen in (c).

Neural Radiance Fields (NeRF) are able to reconstruct scenes with unprecedented fidelity, and various recent works have extended NeRF to handle dynamic scenes. A common approach to reconstruct such non-rigid scenes is through the use of a learned deformation field mapping from coordinates in each input image into a canonical template coordinate space. However, these deformation-based approaches struggle to model changes in topology, as topological changes require a discontinuity in the deformation field, but these deformation fields are necessarily continuous. We address this limitation by lifting NeRFs into a higher dimensional space, and by representing the 5D radiance field corresponding to each individual input image as a slice through this "hyper-space". Our method is inspired by level set methods, which model the evolution of surfaces as slices through a higher dimensional surface. We evaluate our method on two tasks: (i) interpolating smoothly between "moments", i.e., configurations of the scene, seen in the input images while maintaining visual plausibility, and (ii) novel-view synthesis at fixed moments. We show that our method, which we dub *HyperNeRF*, outperforms existing methods on both tasks. Compared to Nerfies, HyperNeRF reduces average error rates by 4.1% for interpolation and 8.6% for novel-view synthesis, as measured by LPIPS. Additional videos, results, and visualizations are available at hypernerf.github.io.

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

Additional Key Words and Phrases: Neural Radiance Fields, Novel View Synthesis, 3D Synthesis, Dynamic Scenes, Neural Rendering

1 INTRODUCTION

Many real-world motions involve changes in topology. Examples include cutting a lemon, or tearing a piece of paper. While not strictly a genus change, motions like closing your mouth cause changes in surface connectivity that can also be considered "topological". Such changes in topology often cause problems for algorithms that seek to reconstruct moving three dimensional scenes, as they cause motion discontinuities or singularities.

A clever approach for addressing topology changes is to represent the 3D scene as a *level set* in a 4D volume. Pioneered in the late 1980s [Osher and Sethian 1988], level set methods model moving scenes as static objects in a higher dimensional ambient space, and topological changes as *smooth* (rather than discontinuous) transformations.

* Work done while the author was an intern at Google.
‡Work done while the author was at Google.
†Sorted alphabetically.
Authors' addresses: Keunhong Park, University of Washington, USA; Utkarsh Sinha, Google Research, USA; Peter Hedman, Google Research, USA; Jonathan T. Barron, Google Research, USA; Sofien Bouaziz, Facebook Reality Labs, USA; Dan B Goldman, Google Research, USA; Ricardo Martin-Brualla, Google Research, USA; Steven M. Seitz, University of Washington, Google Research, USA.

In this paper, we adapt the level set framework for deformable neural radiance fields [Park et al. 2020], to generate photorealistic, free-viewpoint renderings of objects undergoing changes in topology. In doing so, we use modern tools like MLPs to significantly generalize the classical level set framework in key ways. First, whereas classical level sets add a single ambient dimension, we can add any number of ambient dimensions to provide more degrees of freedom. Second, rather than restrict level sets to hyperplanes, as is traditional, we allow general, curved slicing manifolds, represented through MLPs.

Our approach models each observation frame as a nonplanar slice through a hyperdimensional NeRF — a HyperNeRF. Previous methods using higher dimensional inputs require either substantial regularization or additional supervision. In contrast, our method retains a deformation field, which has previously demonstrated strong ability to fuse information across observations, and instead of regularizers, we use an optimization strategy that encourages smooth behavior in the higher dimensions. This enables our method to reconstruct high-quality geometry even when some poses are observed from only a small range of angles.

Our method enables users to capture photorealistic free-viewpoint reconstructions of a wide range of challenging deforming scenes from *monocular video*, e.g., waving a mobile phone in front of a moving scene. We demonstrate the quality of our method on two tasks: (i) interpolating smoothly between "moments" while maintaining visual plausibility, and (ii) novel-view synthesis with fixed moments. Our method, *HyperNeRF*, produces sharper, higher quality results with fewer artifacts on both tasks.

2 RELATED WORK

2.1 Non-Rigid Reconstruction

A common approach in non-rigid reconstruction techniques is to decompose a scene into a canonical model of scene geometry (which is fixed across frames) and a deformation model that warps the canonical scene geometry to reproduce each input image. The difficulty of this task depends heavily on the inherent ambiguity of the problem formulation. Using only a monocular video stream is convenient and inexpensive, but also introduces significant ambiguity which must be ameliorated through the use of factorization [Bregler et al. 2000] or regularization [Torresani et al. 2008]. On the opposite end of the spectrum, complicated and expensive capture setups using multiple cameras and depth sensors can be used to overconstrain the problem, thereby allowing 3D scans to be registered and fused to produce high quality results [Collet et al. 2015; Dou et al. 2016]. Machine learning techniques have been used effectively for non-rigid reconstruction, when applied to direct depth sensors [Božič et al. 2020; Schmidt et al. 2015]. Our method requires only monocular RGB images from a conventional smartphone camera as input — no depth sensors or multi-view capture systems are required.

Several works have used learning techniques to solve for deformation based models of shape geometry [Jiang et al. 2020; Niemeyer et al. 2019], though because these works model only geometry and not radiance, they cannot be applied to RGB image inputs and do not directly enable view synthesis. Yoon et al. [Yoon et al. 2020] use a combination of multi-view cues as well as learned semantic priors in the form of monocular depth estimation to recover dynamic scenes from moving camera trajectories. In contrast, our approach requires no training data other than the input sequence being used as input, as is typical in NeRF-like models. Neural Volumes [Lombardi et al. 2019] represents deformable scenes using a volumetric 3D voxel grid and a warp field, which are directly predicted by a convolutional neural network. As we will demonstrate, Neural Volumes's high fidelity output relies on the use of dozens of synchronized cameras, and does not generalize well to monocular image sequences. The Deformable NeRF technique of Park et al. [2020] uses NeRF-like learned distortion fields alongside the radiance fields of Mildenhall et al. [2020] to recover "nerfies" of human subjects, and is capable of generating photorealistic synthesized views of a wide range of non-stationary subjects. We build directly upon this technique, and extend it to better support subjects that not only move and deform, but that also vary topologically.

2.2 Neural Rendering

The nascent field of "neural rendering" aims, broadly, to use neural networks to render images of things. This is an emerging area of study that is changing rapidly, but progress in the field up through 2020 is well-documented in the survey report of Tewari et al. [2020]. The dominant paradigm in this field has, until recently, been framing the task of synthesizing an image as a sort of "image to image translation" task, in which a neural network is trained to map some representation of a scene into an image of that scene [Isola et al. 2017]. This idea has been extended to incorporate tools and principles from the graphics literature, by incorporating reflectance or illumination models into the "translation" process [Meka et al. 2019; Sun et al. 2019], or by using proxy geometries [Aliev et al. 2019; Thies et al. 2019] or conventional renderings [Fried et al. 2019; Kim et al. 2018; Martin-Brualla et al. 2018; Meshry et al. 2019] as a harness with which neural rendering can then be guided. Though these techniques are capable of producing impressive results, they are often hampered by a lack of consistency across output renderings — when rendering is performed independently by a "black box" neural network, there is no guarantee that all renderings of scene will correspond to a single geometrically-consistent 3D world.

Research within neural rendering has recently begun to shift away from this "image to image translation" paradigm and towards a "neural scene representation" paradigm. Instead of "rendering" images using a black box neural network that directly predicts pixel intensities, scene-representation approaches use the weights of a neural network to directly model some aspect of the physical scene itself, such occupancy [Mescheder et al. 2019], distance [Park et al. 2019], surface light field [Yariv et al. 2020], or a latent representation of appearance [Sitzmann et al. 2019a,b]. The most effective approach within this paradigm has been constructing a neural radiance field (NeRF) of a scene, and using a conventional multilayer perceptron (MLP) to parameterize volumetric density and color as a function of spatial scene coordinates [Mildenhall et al. 2020]. NeRF's usage of classical volumetric rendering techniques has many benefits in addition to enabling photorealistic view synthesis: renderings from NeRF must correspond to a single coherent model of geometry, and

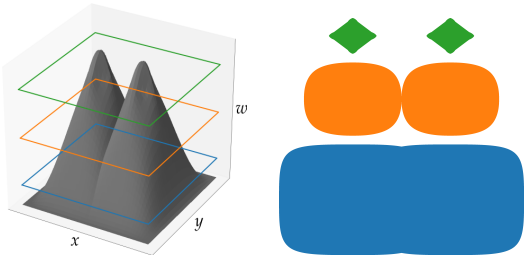
the gradients of volumetric rendering are well-suited to gradient-based optimization. Note that, in NeRF, a neural network is not used to render an image — instead, an analytical physics-based volumetric rendering engine is used to render a scene whose geometry and radiance happen to be parameterized by a neural network.

Though NeRF produces compelling results on scenes in which all content is static, it fails catastrophically in the presence of moving objects. As such, a great deal of recent work has attempted to extend NeRF to support dynamic scenes. We will separate these NeRF variants into two distinct categories: *deformation-based* approaches apply a spatially-varying deformation to some canonical radiance field [Park et al. 2020; Pumarola et al. 2020; Tretschk et al. 2021], and *modulation-based* approaches directly condition the radiance field of the scene on some property of the input image and modify it accordingly [Gafni et al. 2021; Li et al. 2021, 2020; Xian et al. 2020].

Deformation-based NeRF variants follow the tradition established by the significant body of research on non-rigid reconstruction [Newcombe et al. 2015], and map observations of the subject onto a template of that subject. Nerfies [Park et al. 2020], D-NeRF [Pumarola et al. 2020], and NR-NeRF [Tretschk et al. 2021] all define a continuous deformation field which maps observation coordinates to canonical coordinates which are used to query a template NeRF. Because multiple observations of the subject are used to reconstruct a single canonical template, and only the deformation field is able to vary across images, this approach yields a well-constrained optimization problem similar to basic NeRF. Similar to how NeRF parameterizes radiance and density using a coordinate-based MLP, these deformation-based NeRF variants use an MLP to parameterize the deformation field of the scene, and are thereby able to recover detailed and complicated deformation fields. However, this use of a *continuous* deformation field means that these techniques are unable to model any topological variations (e.g., mouth openings) or transient effects (e.g, fire). Topological openings or closings require a discontinuity in the deformation field at the seam of the closing, which MLPs cannot model easily. This is a consequence of the fact that coordinate-based MLPs with positional encoding perform interpolation with a band-limited kernel, when viewed through the lens of neural tangent kernels [Tancik et al. 2020].

Modulation-based or *latent-conditioned* NeRF variants adopt the well established technique of conditioning a neural network with a latent code to modulate its output. 2D generative models such as GANs [Goodfellow et al. 2014] or VAEs [Kingma and Welling 2013] input latent code to a 2D CNN which then outputs a corresponding image. This technique is also used to encode a space of 3D shapes; for example, DeepSDF [Park et al. 2019] and IM-NET [Chen and Zhang 2019] modulate a signed distance field (SDF) based on input latent codes, while Occupancy Networks [Mescheder et al. 2019] modulate an occupancy field. Similarly, SRNs [Sitzmann et al. 2019b] modulate a latent representation which is decoded by an LSTM.

Following these footsteps, instead of modeling the scene as a single NeRF that is warped to explain individual images, modulation-based NeRF techniques provide additional information (e.g., the image's timestamp or a latent code) as input to the MLP, directly changing the radiance field of the scene. As such, these techniques are capable of modeling any deformation, topological change, or even complex phenomena such as fire. However, because these modulated NeRFs may have completely different radiance and density across input images, these techniques result in a severely under-constrained problem and allows for trivial, non-plausible solutions. This issue can be addressed by providing additional supervision such as depth and optical flow (as in Video-NeRF [Xian et al. 2020] and NSFF [Li et al. 2020]) or by using a multi-view input captured from 7 synchronized cameras (as in DyNeRF [Li et al. 2021]). For facial avatars, Gafni et al. [2021] avoids this issue by using a face-centric coordinate frame from a 3D morphable face model [Thies et al. 2016]. NeRF in the Wild also uses a similar modulated approach (albeit for a different task) in which latent codes are optimized and provided as input to an MLP, which also introduces ambiguities that are addressed by allowing those codes to only modify radiance but not density [Martin-Brualla et al. 2021].



(a) 3D Auxiliary Function (b) 2D Sliced Shapes

Fig. 2. Level-set methods provide a means to model a family of topologically-varying shapes (b) as slices of a higher dimensional auxiliary function (a).

Our method can be thought of as a combination of deformation-based and modulation-based approaches: we use deformations to model motion in the scene, resulting in a well-behaved optimization, but we also extend NeRF's 3D input coordinate space to take additional higher-dimension coordinates as input, and allow for deformations along the higher dimensions as well as the spatial dimensions. As we will show, this approach is able to use the higher dimensions to capture changes in object topology, which a strictly deformation-based approach would be unable to model.

3 MODELING TIME-VARYING SHAPES

Our method represents changes in scene topology by providing a NeRF with a higher-dimensional input. To provide an intuition and a justification for our formulation in higher dimensions, this section introduces level set methods which our method draws inspiration from. Note that our method is *not* a level set method, though we will apply intuitions gained from these examples to our NeRF setting. Note that the insights gained from the level set perspective also apply to all existing latent-conditioned neural representations, such as DeepSDF [Park et al. 2019], which are equivalent to the axis-aligned slicing plane formulation described in Sec. 3.2. We believe this provides a good visual intuition for existing methods.

There are two common approaches for mathematically representing the surface of an object: a surface can be defined *explicitly*, perhaps with a polygonal mesh, or *implicitly*[1], perhaps as the level set

[1]Confusingly, coordinate-based approaches such as NeRF are sometimes referred to in the literature as "implicit" models, in reference to the idea that they encode scenes

of a continuous function. Explicit representations of shape, though effective and ubiquitous, are often poorly suited to topological variation of a surface: slicing a polygonal mesh into two halves, for example, likely requires creating new vertices and redefining the edge topology of the mesh. This sort of variation is particularly difficult to express in the context of gradient-based optimization methods such as NeRF, as this transformation is discontinuous and therefore not easily differentiated. In contrast, implicit surfaces provide a natural way to model the evolution of a surface in the presence of topological changes, such as when the surface develops a hole or splits into multiple pieces.

3.1 Level Set Methods

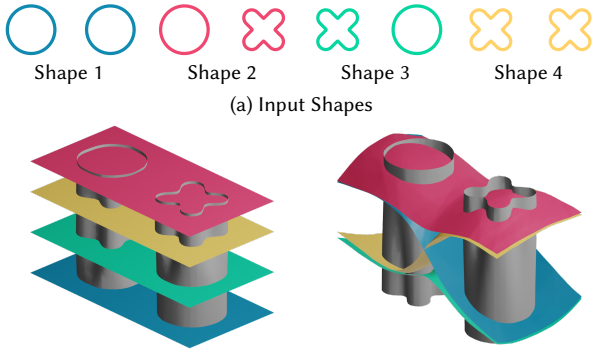
Level set methods model a surface implicitly as the zero-level set of an auxiliary function [Osher and Sethian 1988]. For example, a 2D surface can be defined as $\Gamma = \{(x, y)|S(x, y) = 0\}$, where $S : (x, y) \rightarrow s$ is a signed-distance function with $s > 0$ for points inside the surface, and $s < 0$ for points outside the surface. To model a surface that varies topologically with respect to some additional dimension (such as "time"), one can add an additional dimension w, and thereby define a 3D surface $\Gamma = \{(x, y, w)|S(x, y, w) = 0\}$. We will refer to the space by these additional dimensions as the "ambient" space. The 2D surface at some w_i ambient coordinate can then be expressed as the 2D cross-section of the 3D surface Γ obtained by slicing it with the plane passing through $w = w_i$. See Fig. 2 for an illustration.

This idea can be extended to learn a collection of shapes. Given a set of implicitly-defined 2D shapes, one can learn an SDF Γ which contains all such 2D shapes as individual slices of a canonical 3D surface. Following DeepSDF [Park et al. 2019], let $S : (x, y, w) \rightarrow s$ be an MLP, where the per-shape ambient coordinates $\{w_i\}$ are learned as an embedding layer [Bojanowski et al. 2018]. The weights of the MLP and the hyper coordinates can then be optimized using gradient descent. Fig. 3 shows four different shapes being encoded in a single 3D SDF using this MLP. As with time-varying shapes, this approach gives us a natural way to interpolate between the shapes by interpolating between the learned slicing planes. As shown by DeepSDF, this formulation can be extended to an arbitrary number of spatial and ambient dimensions. For example, by formulating this same problem with 3 spatial dimensions and 256 ambient dimensions, we can learn 3D shapes as 3-dimensional cross sections of a 259-dimensional hyper-surface.

3.2 Deformable Slicing Surfaces

Level set methods work by "slicing" through a function, usually with an axis-aligned plane. The plane of this slice spans the spatial axes (the x and y axes in our 2D/3D example), and occupies a single ambient coordinate (the w axis in our example) at all points. A consequence of using an axis-aligned slice is that every desired output shape must exist as a cross section cut by the slicing plane. In certain circumstances, this can lead to an inefficient use of space. For example, consider the set of shapes shown in Fig. 3a, which are

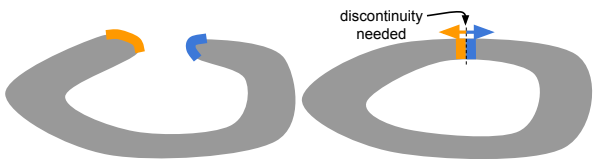
"implicitly" using the weights of a neural network. This new use of "implicit" is distinct from its common use in the literature, so to avoid confusion we will only use "implicit" according to its meaning in the level set literature.



(a) Input Shapes

(b) Axis-aligned Slicing Plane (AP) (c) Deformable Slicing Surface (DS)

Fig. 3. Level-set methods model shapes as slices of a higher-dimensional surface, which we call an "ambient surface". We present two slicing methods: *axis-aligned planes* (AP) slice the surface perpendicular to the higher-dimensional axis; *deformable surfaces* (DS) can deform and thus different parts of the shape can reference varying parts of hyper-space. Axis-aligned planes require a copy of each shape separately, while deformable surfaces can share information and resulting in simpler ambient surfaces. We encode deformable slicing surfaces in the weights of an MLP.



(a) Template (b) Observation

Fig. 4. An example topological change where the ring opens at the marked seam. A deformation field referencing the template for each point in the observation frame would require a discontinuity at the seam, where an infinitesimal step from the orange position towards the blue position results in a big change in the deformation. If the template were the closed ring, the contents of the ring would be inaccessible using a deformation field.

different permutations of the same two shapes. If axis-aligned slices are used, the ambient surface must contain a *copy* of each of the four permutations as shown in Fig. 3b. This is inefficient considering there are only two possible sub-shapes—a circle and a cross.

To address this, we introduce another MLP which encodes a *deformable* slicing surface, such that the output shape is the cross-section sliced by a non-planar surface as in Fig. 3c. This allows different spatial locations to reference different parts of the ambient coordinate space, resulting in a more compact representation in hyperspace. We define the deformable slicing surface as an MLP $H : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow \mathbf{w}$, where $\mathbf{x}$ is a spatial position, $\mathbf{w}$ is a position along the ambient axes, and $\boldsymbol{\omega}_i$ is a per-input latent embedding (whose dimensionality and meaning need not match that of the ambient coordinate). The SDF is then queried at the coordinate obtained by concatenating $\mathbf{x}$ and $H(\mathbf{x}, \boldsymbol{\omega}_i)$. As shown in Fig. 3c, this parameterization is able to slice through the ambient dimensions to model arbitrary mixtures of shapes. See the supplementary materials for details on the experiment for Fig. 3.

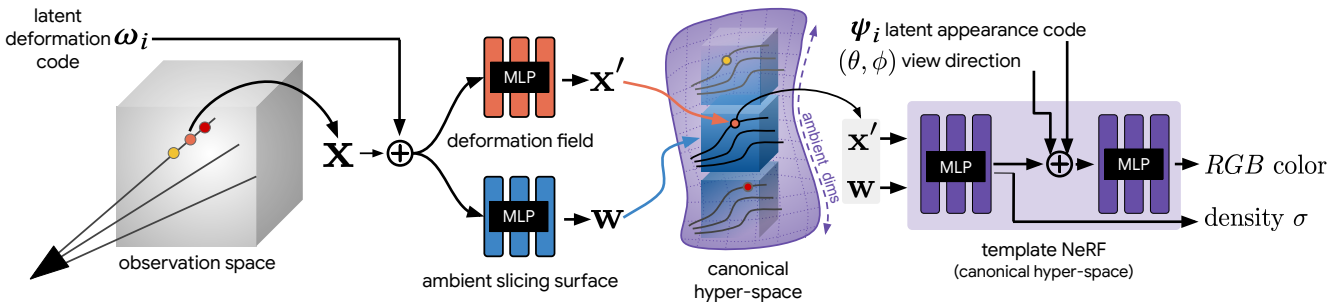


Fig. 5. An overview of our model architecture. We associate a latent deformation code $\boldsymbol{\omega}_i$ and a latent appearance code $\boldsymbol{\psi}_i$ with each image i. Rays are cast from the camera in the space, and samples $\mathbf{x}$ along those rays are then concatenated with the image's latent deformation code $\boldsymbol{\omega}_i$ and provided as input to MLPs parameterizing a deformation field, which yields a warped coordinate $\mathbf{x}'$ and a slicing surface in hyperspace, which yields a coordinate in our ambient space $\mathbf{w}$. Both outputs are concatenated to a mapped point $(\mathbf{x}', \mathbf{w}) = (x, y, z, w_1, w_2, \ldots)$. The concatenation of $(\mathbf{x}', \mathbf{w})$, the viewing direction $\mathbf{d} = (\theta, \phi)$, and the appearance code $\boldsymbol{\psi}_i$ are then used as inputs to the MLP parameterizing the template NeRF. The densities and colors produced by that MLP are then integrated along the ray as per the physics of volumetric rendering, as in Mildenhall et al. [2020].

4 METHOD

Here we describe our method for modeling non-rigidly deforming scenes given a casually captured monocular image sequence. The focus of our method is to be able to accurately model the appearance and geometry of *topologically varying* scenes. Before we introduce HyperNeRF, we first review its foundations: neural radiance fields (NeRF) [Mildenhall et al. 2020] as well as unconstrained and deformable extensions of it [Martin-Brualla et al. 2021; Park et al. 2020], all of which we will build upon.

4.1 Review of NeRF, NeRF-W, and Deformable NeRF

NeRF represents a scene as a continuous, volumetric field of density and radiance, defined as $F : (\mathbf{x}, \mathbf{d}, \boldsymbol{\psi}_i) \rightarrow (\mathbf{c}, \sigma)$. The function F is parameterized by a multilayer perception (MLP) and maps a 3D position $\mathbf{x} = (x, y, z)$ and viewing direction $\mathbf{d} = (\phi, \theta)$ to a color $\mathbf{c} = (r, g, b)$ and density σ. Instead of directly providing input coordinates to the MLP directly, NeRF first maps each input $\mathbf{x}$ (and $\mathbf{d}$) using a sinusoidal positional encoding:

$$\gamma(\mathbf{x}) = \left[\sin(\mathbf{x}), \cos(\mathbf{x}), \sin(2\mathbf{x}), \cos(2\mathbf{x}), \ldots, \sin(2^{m-1}\mathbf{x}), \cos(2^{m-1}\mathbf{x})\right]^{\mathrm{T}}, \quad (1)$$

where m is a hyper-parameter that controls the number of sinusoids used by the encoding. As shown in Tancik et al. [2020], this encoding allows the MLP to model high-frequency signals in low frequency domains, where the parameter m serves to control smoothness of the learned representation by modifying the effective bandwidth of an interpolating kernel.

Appearance Variation. To handle unconstrained "in the wild" images, the MLP in NeRF can be additionally conditioned on an appearance embedding $\boldsymbol{\psi}_i$ for each observed frame $i \in \{1, \ldots, n\}$, as shown in Martin-Brualla et al. [2021]. This allows NeRF to handle appearance variations between input frames, such as those caused by illumination variation or changes in exposure and white balance.

Deformations. Because a NeRF is only able to represent static scenes, we use the deformation field formulation proposed in Nerfies [Park et al. 2020] to model non-rigid motion. Nerfies defines a mapping $T : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow \mathbf{x}'$ that maps all observation-space coordinates $\mathbf{x}$ to canonical-space coordinates $\mathbf{x}'$, conditioned on a per-observation latent deformation code $\boldsymbol{\omega}_i$. The deformation field T is parameterized by an MLP $W : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow (\mathbf{r}, \mathbf{v})$, where $(\mathbf{r}, \mathbf{v}) \in \mathfrak{se}(3)$ encode the rotation and translation. To encourage deformations to be as-rigid-as-possible, Nerfies proposes an elastic regularization loss which penalizes non-unit singular values of the Jacobian of the deformation field. We found that the elastic loss is ill-suited for some of our scenes due to the topological variations which directly violate its assumptions. Please see Park et al. [2020] for details.

Windowed Positional Encoding. Tancik et al. [2020] showed that the number of frequencies m in the positional encoding γ controls the bandwidth of an MLP's Neural Tangent Kernel (NTK) [Jacot et al. 2018]. A small value for m results in a smooth estimator that may under-fit the data, while a large value of m may result in over-fitting. Park et al. [2020] and Hertz et al. [2021] use this property to implement a coarse-to-fine strategy when optimizing an MLP, by slowly narrowing the bandwidth of the NTK by weighting the frequency bands of the positional encoding with a window function. For example, Park et al. [2020] uses the window function

$$w_j(\alpha) = \frac{1 - \cos(\pi \operatorname{clamp}(\alpha - j, 0, 1))}{2}, \quad (2)$$

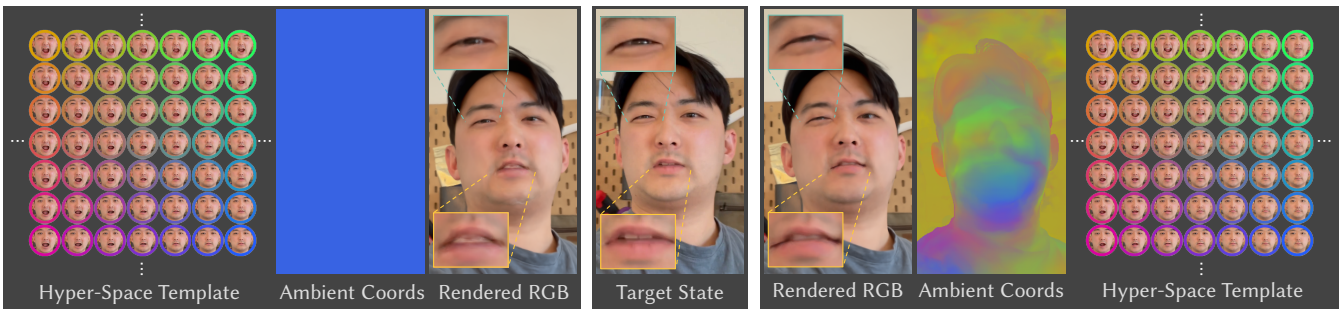
where $j \in \{0, \ldots, m-1\}$ is the index of the frequency band of the positional encoding. Linearly increasing $\alpha \in [0, m]$ is equivalent to sliding a truncated Hann window down the frequency bands of positional encoded features. The windowed positional encoding is then computed as:

$$\gamma_\alpha(\mathbf{x}) = \left[w_0(\alpha)\sin(\mathbf{x}), \ldots, w_{m-1}(\alpha)\cos(2^{m-1}\mathbf{x})\right]^{\mathrm{T}}. \quad (3)$$

Nerfies uses this to optimize the deformation field in a coarse-to-fine manner which prevents getting stuck in sub-optimal local minima while being retaining the ability to represent high-frequency deformations. We do the same for our spatial deformation field.

4.2 Hyper-Space Neural Radiance Fields

Motion in a scene can be divided into two categories: (a) motions that preserve the topology of the scene; and (b) motions which



(a) Axis-Aligned Slicing Plane (AP) (b) (c) Deformable Slicing Surface (DS)

Fig. 6. We render the target state (b) from a novel view using axis-aligned slicing planes (AP, a) and deformable surfaces (DS, c). We show the hyper-space template rendered at ambient coordinates, sampled on a regular grid. We plot the ambient coordinate of each pixel — the color of the coordinates correspond to the outlined color of each template sample. AP must wholly model each scene state and results in artifacts in the eyes, mouth, and chin. DS can more efficiently use the template since each part of the scene can refer to different parts of the template, resulting in sharper details.

change the apparent topology of the scene. Here we use the term "topology" in the sense we defined in Sec. 1. Deformation fields can effectively model topology-preserving motion, but cannot easily model changes in topology. This is because a change in topology necessarily involves a discontinuity in the deformation field. Our deformation fields are continuous, by virtue of being encoded within the weights of an MLP that behaves as a smooth interpolator, and therefore cannot represent such discontinuities. See Fig. 4 for a visual explanation.

Here we present our method, *HyperNeRF*, which extends neural radiance fields into higher dimension to allow topological variations. In Sec. 3.1 we introduced the level set method, which can naturally model topologically changing shapes as cross-sections of a higher-dimensional ambient surface. We use the same idea in the context of neural radiance fields. Our architecture is visualized in Fig. 5.

Hyper-space Template. Deformable NeRFs [Park et al. 2020] represent a scene in a canonical-space *template* NeRF which is indexed by a spatial deformation field to render observation frames. Our idea is to embed the template NeRF in higher dimensions, where a slice taken by an intersecting high-dimensional slicing surface yields a full 3D NeRF, akin to slicing a 3D object to get a 2D shape as in Fig. 2. Note that we are still rendering 3-dimensional scenes, and thus ray casting and volume rendering occur in the same manner as a normal NeRF.

We extend the domain of the template NeRF to a higher dimensional space: $(\mathbf{x}, \mathbf{w}) \in \mathbb{R}^{3+W}$, where W is the number of higher dimensions (visualized in Fig. 6a). The formulation of the template NeRF is otherwise similar, with it being represented as an MLP

$$F : (\mathbf{x}, \mathbf{w}, \mathbf{d}, \boldsymbol{\psi}_i) \rightarrow (\mathbf{c}, \sigma) . \tag{4}$$

Slicing Surfaces. In Sec. 3.2 and Fig. 3 we showed how a 3D auxiliary surface could be cut with a slicing surface to obtain a 2D shape. In the same way, we can take cross-sections of a hyper-space NeRF. While our 2D examples only had a single ambient dimension for visualization purposes, we can have more. Slicing an auxiliary shape with a single ambient dimension involves a single slicing plane; slicing an auxiliary shape with more than one ambient dimension simply requires a slicing plane for each dimension. This is the geometric interpretation of evaluating the auxiliary function at a specific value of $\mathbf{w}$, which leaves the span of the spatial axes as the cross-section along the ambient dimensions.

Analogous to the 2D *axis-aligned slicing plane (AP)* formulation described in Sec. 3.2, we can associate a specific slice of the template for each observation i by directly optimizing the ambient coordinates $\{\mathbf{w}_i\}$ as with the deformation and appearance codes. This approach forces all ray samples for an observation to share the same ambient coordinates, requiring all observed states to be explicitly modeled at a single ambient coordinate. This can lead to an inefficient use of the representation capacity of the template NeRF, since topological variations happening in spatially distance parts of the scene must be copied across multiple sub-spaces to represent different combinations of states (as with Fig. 3b). This can decrease reconstruction quality (Fig. 6a) and causes fading artifacts when interpolating between different states (Fig. 7).

We therefore use *deformable slicing surfaces (DS)* — introduced in Sec. 3.2 — which allow different spatial positions to be mapped to different coordinates along the ambient dimensions. This allows a more efficient use of the ambient dimensions (as in Fig. 3c), resulting in better reconstruction quality (Fig. 6c) and smoother interpolations between states compared to axis-aligned planes (Fig. 7).

Similarly to the spatial deformation field, we define a deformable slicing surface field using an MLP. Each observation-space sample point $\mathbf{x}$ is mapped through the mapping $H : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow \mathbf{w}$, where $\boldsymbol{\omega}_i$ is the latent deformation code (shared with the spatial deformation field), and $\mathbf{w}$ is a point in the ambient coordinate space which defines the cross-sectional subspace for the sample. Given the template NeRF F, the spatial deformation field T, and the slicing surface field H, the observation-space radiance field can be evaluated as:

$$\mathbf{x}' = T(\mathbf{x}, \boldsymbol{\omega}_i) , \tag{5}$$

$$\mathbf{w} = H(\mathbf{x}, \boldsymbol{\omega}_i) , \tag{6}$$

$$(\mathbf{c}, \sigma) = F(\mathbf{x}', \mathbf{w}, \mathbf{d}, \boldsymbol{\psi}_i) . \tag{7}$$

where in practice we apply separate positional encodings to $\mathbf{x}$, $\mathbf{d}$, and $\mathbf{w}$. We use a windowed positional encoding γ_α for the deformation field T and γ_β for the slicing surface field H, where α and β are parameters for the windowed positional encoding defined in Eq. (3).

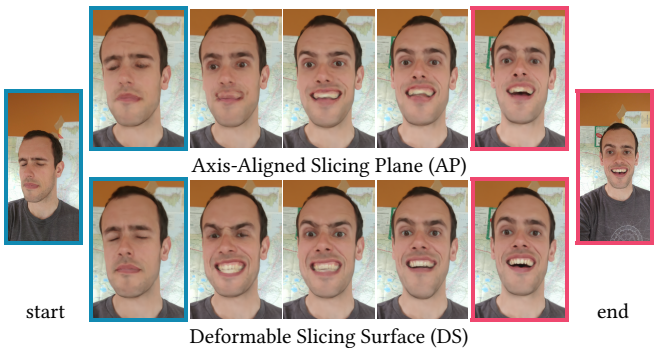


Fig. 7. Novel views synthesized by linearly interpolating the embedding $\boldsymbol{\omega}_i$ of two frames from "Expressions 1". With (AP), all spatial points are rendered at the same ambient coordinate resultin in blurry-cross-fading artifacts during interpolation. With (DS), different spatial positions to reference varying parts of the ambient space, resulting in better interpolations.

We do not use the identity concatenation of the positional encoding for the ambient coordinates. Barron et al. [2021] showed that the identity encoding does not meaningfully affect performance or speed, and omitting it confers a specific advantage: we can collapse ambient dimensions during the initial phases of the optimization by setting $\beta = 0$, as described in the previous section.

Delayed use of Ambient Dimensions. Motion can be encoded by deforming points using the spatial deformation field T, or by moving along the ambient dimensions using the slicing surface H. If a motion does not involve a change in topology, we prefer using a spatial deformation over moving through the ambient dimensions. This is because spatial deformations result in a more constrained optimization, since several observations get mapped to the same point on the template. In addition, spatial deformations result in better interpolations and they only involve movement and cannot directly change the density of the template. We therefore delay the use of the ambient dimensions by using the windowed positional encoding (Sec. 4.1) on the ambient coordinates $\mathbf{w}$. We disable the identity concatenation for the positional encoding so that the input can be completely turned off when $\beta = 0$. We fix $\beta = 0$ for a number of iterations and then linearly ease in the rest of the frequencies. This windowing is only applied to the ambient dimensions. The spatial coordinate $\mathbf{x}$ is not windowed, and the spatial deformation field uses its own windowing schedule from Park et al. [2020].

5 EXPERIMENTS

5.1 Implementation Details

Our implementation of NeRF and the deformation field closely follows Nerfies [Park et al. 2020]. As in NeRF [Mildenhall et al. 2020], we only use an L2 photometric loss. We use 8 dimensions for the latent appearance and deformations codes. We exponentially decay our learning rate from 10^{-3} to 10^{-4}. For the windowed positional encoding parameter α of the deformation field, we follow the same easing schedule as in Park et al. [2020]. We implement the deformable slicing surface as an MLP with depth 6 and width 64 with a skip connection at the 5th layer. The last layer is initialized with weights sampled from $\mathcal{N}(0, 10^{-5})$. We use $m = 1$ for the hyper-coordinate input $\mathbf{w}$ to the template T, fixing $\beta = 0$ for 1000 iterations and then linearly increase it to 1 over 10k iterations. We use $m = 6$ for the spatial position input $\mathbf{x}$ to the slicing surface field H. We use 2 ambient dimensions ($W = 2$) for all experiments, as increasing W did not improve performance for our sequences.

We implement our method on top of JaxNeRF [Deng et al. 2020], a JAX [Bradbury et al. 2018] implementation of NeRF. For our evaluation metrics, we train at half of 1080p resolution (960x540) for 250k iterations, using 128 samples per ray with a batch size of 6,144, which takes roughly 8 hours on 4 TPU v4s. For qualitative results, we train at full-HD (roughly 1920x1080) with 256 samples per ray for 1M iterations, which takes roughly 64 hours.

5.2 Evaluation

Here we analyze the performance of our method both quantitatively and qualitatively. To best judge quality, we urge the reader to view the supplementary video which contains many visual results.

5.2.1 Quantitative Evaluation. We evaluate our method on two tasks: (i) how well it interpolates between different moments seen during training while maintaining visual plausibility; and (ii) its ability to perform novel-view synthesis. We collected our own sequences for these tasks, as existing datasets do not focus on topologically varying scenes. While the dataset of Yoon et al. [2020] contains two sequences exhibiting topological changes, these sequences have short capture baselines (~1.1m) and exaggerated frame-to-frame motion due to aggressive temporal sub-sampling.

We measure visual quality with LPIPS [Zhang et al. 2018], MS-SSIM [Wang et al. 2003], and PSNR. Note that because we are reconstructing dynamic scenes with a single moving camera, there can be ambiguities which cause slight differences from the true geometry or appearance of the scene. Because of this, we find that quantitative metrics often do not reflect perceptual quality, as we show in Fig. 10. In particular, PSNR is incredibly sensitive to small shifts, and will penalize sharp images over blurry results, while MS-SSIM may not pick up artifacts which are obvious to humans. Out of the three metrics, we find that LPIPS best reflects perceptual quality.

Ablations. We also compare with a couple variants of HyperNeRF: "HyperNeRF (DS)" uses the deformable slicing surface (Sec. 3.2), "HyperNeRF (DS, w/ elastic)" uses the elastic regularization loss from Park et al. [2020], "HyperNeRF (AP)" uses the axis-aligned slicing plane, and "HyperNeRF (w/o deform)" removes the deformation field and only uses the hyper-space formulation.

Interpolation. Like Nerfies [Park et al. 2020], our method can interpolate between moments in an input video by interpolating the input embeddings $\boldsymbol{\omega}_i$ and $\boldsymbol{\psi}_i$. Here, we evaluate quality by leaving out intermediate frames from the input video and comparing them with corresponding interpolated frames. Specifically, we collect a set of videos, each 30-60s long, sub-sampled to 15fps and register the frames using COLMAP [Schönberger and Frahm 2016].

We build our dataset by using every 4th frame as a training frame, and taking the middle frame between each pair of training frames as a validation frame. Since certain frames may have been dropped due failed registration in COLMAP, we use the timestamp

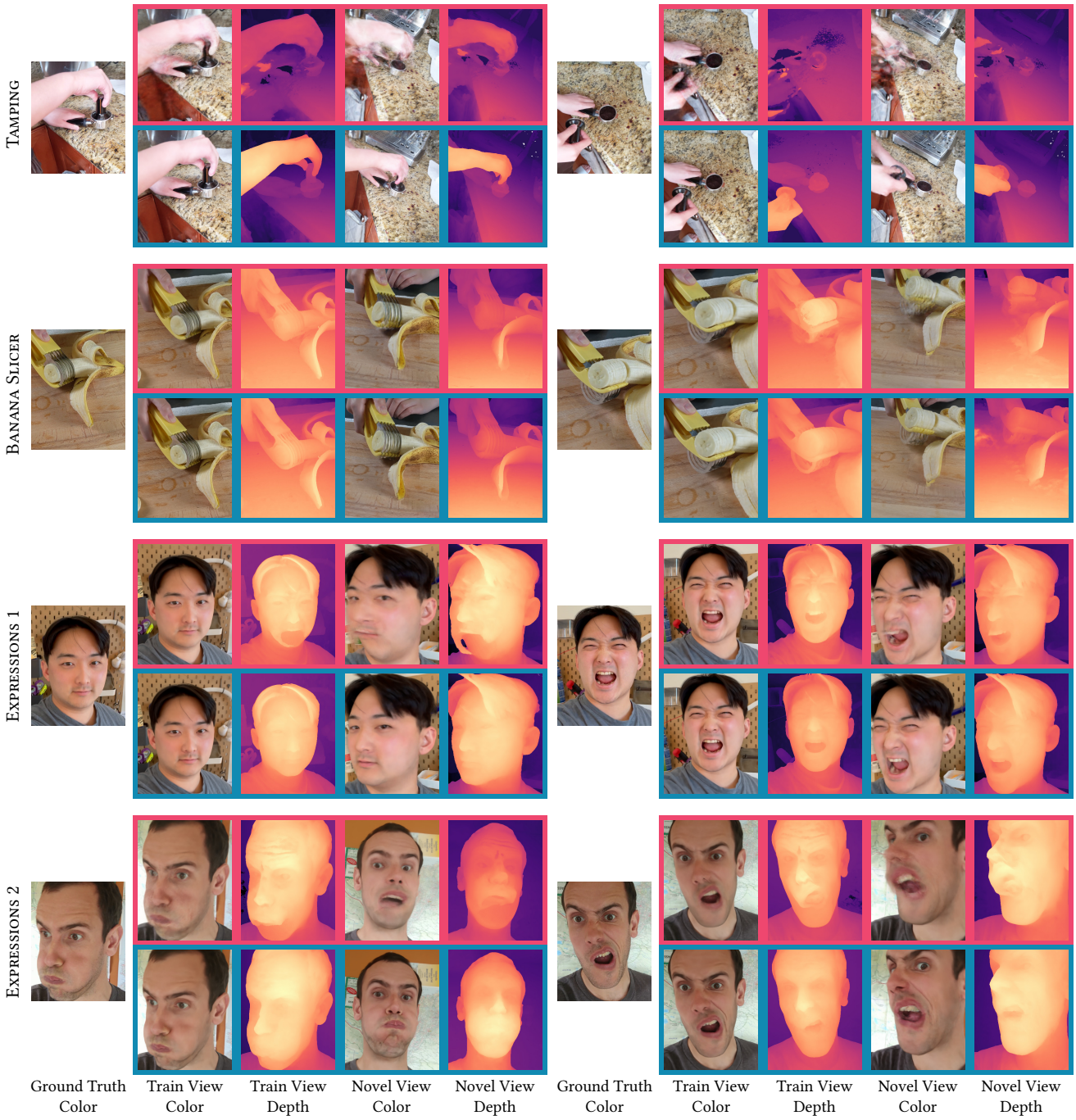
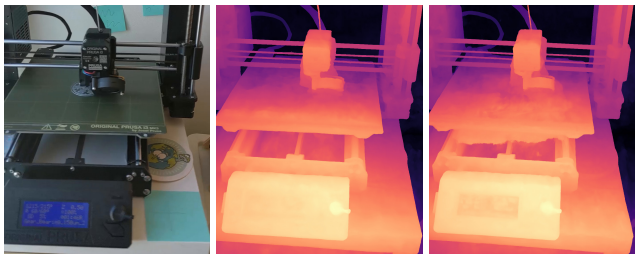


Fig. 8. Here we show qualitative comparisons of Nerfies [Park et al. 2020] (top rows) with HyperNeRF (ours) (bottom rows) on four different image sequences. For each sequence we visualize two different moments, each with a different apparent topology. Nerfies is unable to model the topological variation with its deformation field and therefore distorts the geometry in implausible ways in order to explain the training data, while HyperNeRF produces more plausible geometry estimates and more accurate renderings on novel views not seen during training.



(a) Input Color (b) HyperNeRF (ours) (c) [Park et al. 2020]

Fig. 9. Although not technically a change in topology, the bed of the 3D printer moving relative to its base requires sharp changes in the deformation field, resulting in artifacts in the geometry. *HyperNeRF* alleviates these.

Table 1. Metrics on the interpolation task, averaged across all sequences. We report PSNR, MS-SSIM, LPIPS, and the "average" metric of Barron et al. [2021]. We color each row as **best**, **second best**, and **third best**. See Sec. 5.2.1 for details, and the supplement for metrics for each sequence.

	PSNR↑	MS-SSIM↑	LPIPS↓	Avg↓
NeRF [Mildenhall et al. 2020]	21.3	.745	.490	.126
NV [Lombardi et al. 2019]	25.3	.880	.214	.0630
NSFF [Li et al. 2020]	25.5	.863	.242	.0663
Nerfies [Park et al. 2020]	27.7	.908	.193	.0487
Nerfies (w/o elastic)	28.0	.909	.193	.0476
Hyper-NeRF (DS)	28.3	.914	.185	.0456
Hyper-NeRF (DS, w/ elastic)	28.1	.912	.195	.0471
Hyper-NeRF (AP)	28.3	.912	.208	.0476
Hyper-NeRF (w/o deform)	27.4	.895	.253	.0557

of the validation frame to compute its relative position between its corresponding training frames.

We compare our method with Nerfies [Park et al. 2020], Neural Volumes [Lombardi et al. 2019], and NSFF [Li et al. 2020]. For Neural Volumes, we render the interpolated frame by encoding the two reference frames with the encoder and linearly interpolating the latent codes. For NSFF, we linearly interpolate the input time variable between the two reference frames. Tab. 1 shows that our method outperforms all three baselines in most cases. We show qualitative results for the interpolation sequences in Fig. 11 and Fig. 12, with additional results in the supplementary materials.

Novel-view Synthesis. We evaluate the novel-view synthesis quality of our method on topologically varying scenes. We render the scene at fixed moments from unseen viewpoints and compare how well the predicted images match the corresponding ground truth image. To allow for a fair comparison with prior work, we closely follow the evaluation protocol of Park et al. [2020]. We create a capture rig comprised of a pole with two Pixel 3 phones rigidly attached roughly 16cm apart. Please see Park et al. [2020] for details on dataset processing. We evaluate on the Broom sequence from Park et al. [2020] which exhibits a topological change when the broom contacts the ground, and augment the sequences of Park et al. [2020] with 4 additional sequences exhibiting topological changes: 3D Printer, Chicken, Expressions, and Peel Banana. Tab. 2 reports the metrics computed on unseen validation views.

Baselines. For view-synthesis, we compare with NeRF [Mildenhall et al. 2020], Neural Volumes [Lombardi et al. 2019] and two recent dynamic NeRF methods: NSFF [Li et al. 2020] and Nerfies [Park et al. 2020]. For interpolation, we compare with NeRF, Nerfies, and Neural Volumes. To evaluate Neural Volumes on the interpolation task, we compute the latent codes of the two training images corresponding to each validation image with the image encoder, and linearly interpolate the encoding vector using the identical procedure mentioned in Sec. 5.2.1. As Neural Volumes works with fixed size inputs, we only evaluate error metrics on a central crop, resulting in slightly improved results for this baseline. We do not compare with Yoon et al. [2020] as their code was not available. Instead, we compare with NSFF [Li et al. 2020] which achieves higher quality on the dataset of Yoon et al. [2020]. Note that the default hyper-parameters for NSFF [Li et al. 2020] provided with the code release performs poorly on our sequences — we therefore contacted the authors to help us tune the hyper-parameters (see appendix).

5.2.2 Qualitative Results. Fig. 8 shows qualitative results, comparing our method with Park et al. [2020]. We also show visual results from the validation rig dataset in Fig. 10. Note how some image quality metrics sometimes prefer blurry results (e.g. Expressions).

Like our method, Nerfies [Park et al. 2020] is able to produce sharp results with few artifacts. However, our method better reconstructs the poses of objects in scenes such as Peel Banana and Expressions in Fig. 10. This is also shown by the warped faces in Fig. 8, where Nerfies often does not model geometry for the chin.

6 LIMITATIONS AND CONCLUSION

As with all NeRF-like methods, camera registration affects the quality of the reconstruction. And, since our model uses only color images as input, imposing no domain-specific priors, we can only reconstruct what is observed. So, moments which are not captured well in the training data — such as when there is rapid motion — cannot be reconstructed by our method. We consider these limitations as fruitful avenues for future work.

We have presented HyperNeRF, an extension to NeRF that reconstructs topologically-varying scenes with discontinuous deformations. Deformation-based dynamic NeRF models cannot model such topological variations due to their use of continuous deformation fields, encoded within the weights of an MLP. HyperNeRF models these variations as slices through a higher-dimensional space. To keep the space compact and avoid overfitting, we delay the use of these ambient dimensions, and then use deformable hyperplanes to extract these slices. Thus, we have combined insights from level-set methods and deformation-based dynamic NeRF models to reconstruct both large motions and topological variations.

ACKNOWLEDGMENTS

We thank Xuan Luo, Aleksander Holynski, and Hyunjeong Cho for their help with collecting data. We thank Pratul Srinivasan and John Flynn for their insightful discussions. This work was supported in part by the UW Reality Lab, Amazon, Facebook, Futurewei, and Google.

Table 2. Quantitative evaluation on validation rig captures. We compare with baselines and ablations of our method. We color each row as **best**, **second best**, and **third best**. Note that traditional metrics like PSNR and SSIM are sensitive to small shifts, penalizing sharp images over blurry results (see Fig. 10 below). See Sec. 5.2.1 for more details on these experiments.

	Broom (197 images)			3D printer (207 images)			Chicken (164 images)			Expressions (259 images)			Peel Banana (513 images)			Mean		
	PSNR↑	MS-SSIM↑	LPIPS↓	PSNR↑	MS-SSIM↑	LPIPS↓	PSNR↑	MS-SSIM↑	LPIPS↓	PSNR↑	MS-SSIM↑	LPIPS↓	PSNR↑	MS-SSIM↑	LPIPS↓	PSNR↑	MS-SSIM↑	LPIPS↓
NeRF [Mildenhall et al. 2020]	19.9	.653	.692	20.7	.780	.357	19.9	.777	.325	20.1	.697	.394	20.0	.769	.352	20.1	.735	.424
NV [Lombardi et al. 2019]	17.7	.623	.360	16.2	.665	.330	17.6	.615	.336	14.6	.672	.276	15.9	.380	.413	16.4	.591	.343
NSFF [Li et al. 2020]†	26.1	.871	.284	27.7	.947	.125	26.9	.944	.106	26.7	.922	.157	24.6	.902	.198	26.4	.917	.174
Nerfies [Park et al. 2020]	19.2	.567	.325	20.6	.830	.108	26.7	.943	.0777	21.8	.802	.150	22.4	.872	.147	22.1	.803	.162
Nerfies (w/o elastic)	19.4	.581	.323	20.2	.820	.115	26.0	.935	.0837	21.8	.800	.149	21.7	.852	.157	21.8	.798	.165
Hyper-NeRF (DS)	19.3	.591	.296	20.0	.821	.111	26.9	.948	.0787	21.6	.800	.148	23.3	.896	.133	22.2	.811	.153
Hyper-NeRF (DS, w/ elastic)	19.5	.605	.277	20.2	.823	.109	27.5	.954	.0756	21.9	.806	.144	22.7	.882	.133	22.3	.814	.148
Hyper-NeRF (AP)	19.6	.596	.319	20.0	.814	.131	27.2	.950	.0941	22.2	.817	.149	22.4	.874	.142	22.2	.810	.167
Hyper-NeRF (w/o deform)	20.6	.714	.613	21.4	.846	.212	27.6	.950	.108	22.0	.793	.196	24.3	.914	.170	23.2	.843	.260

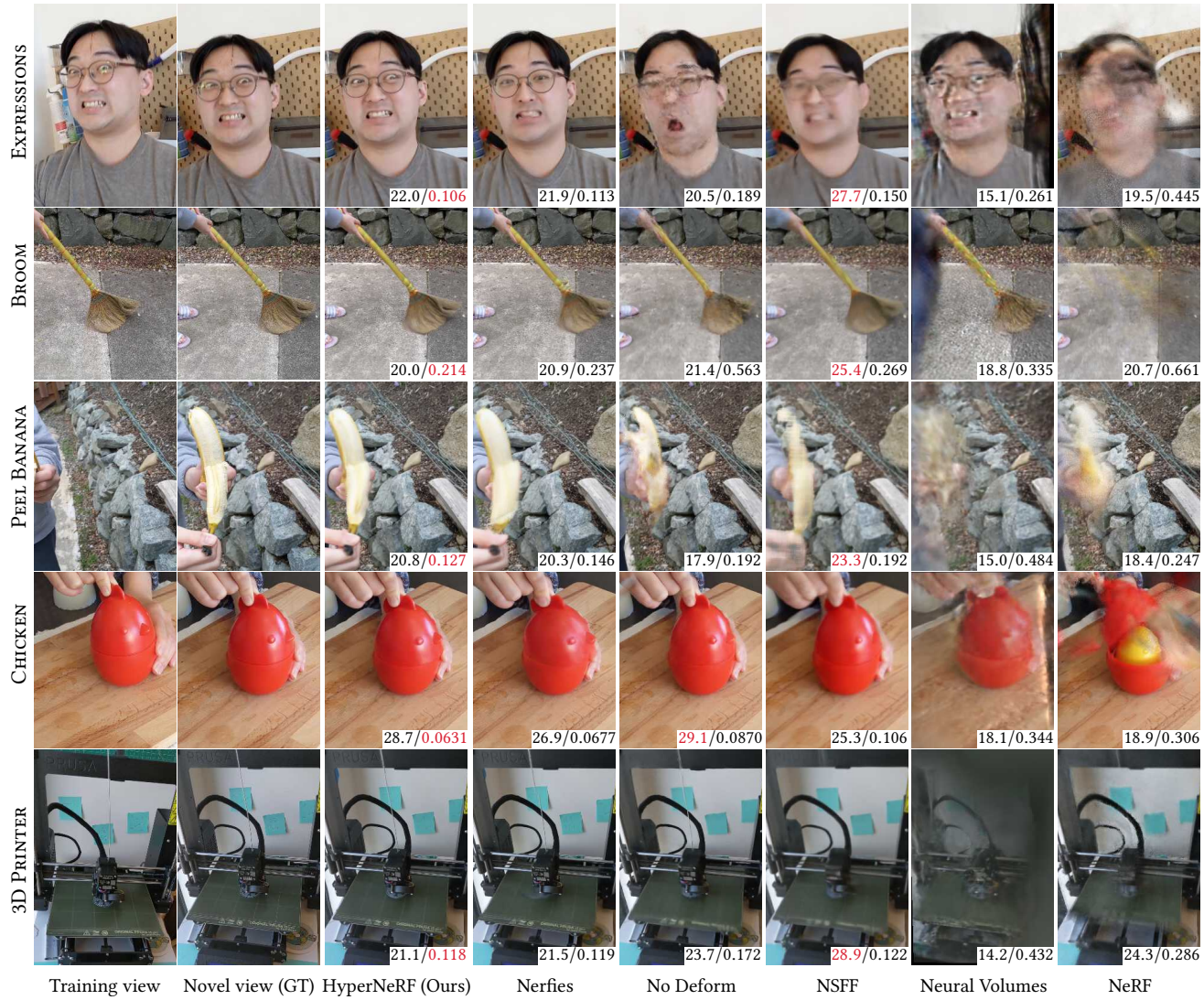


Fig. 10. Comparisons of baselines and our method on validation rig scenes. PSNR / LPIPS metrics on bottom right with best colored red. Baselines shown are: Nerfies [Park et al. 2020], HyperNeRF without deformations, Neural Volumes [Lombardi et al. 2019], NSFF [Li et al. 2020], and NeRF [Mildenhall et al. 2020]. Note how PSNR often prefers blurry results.

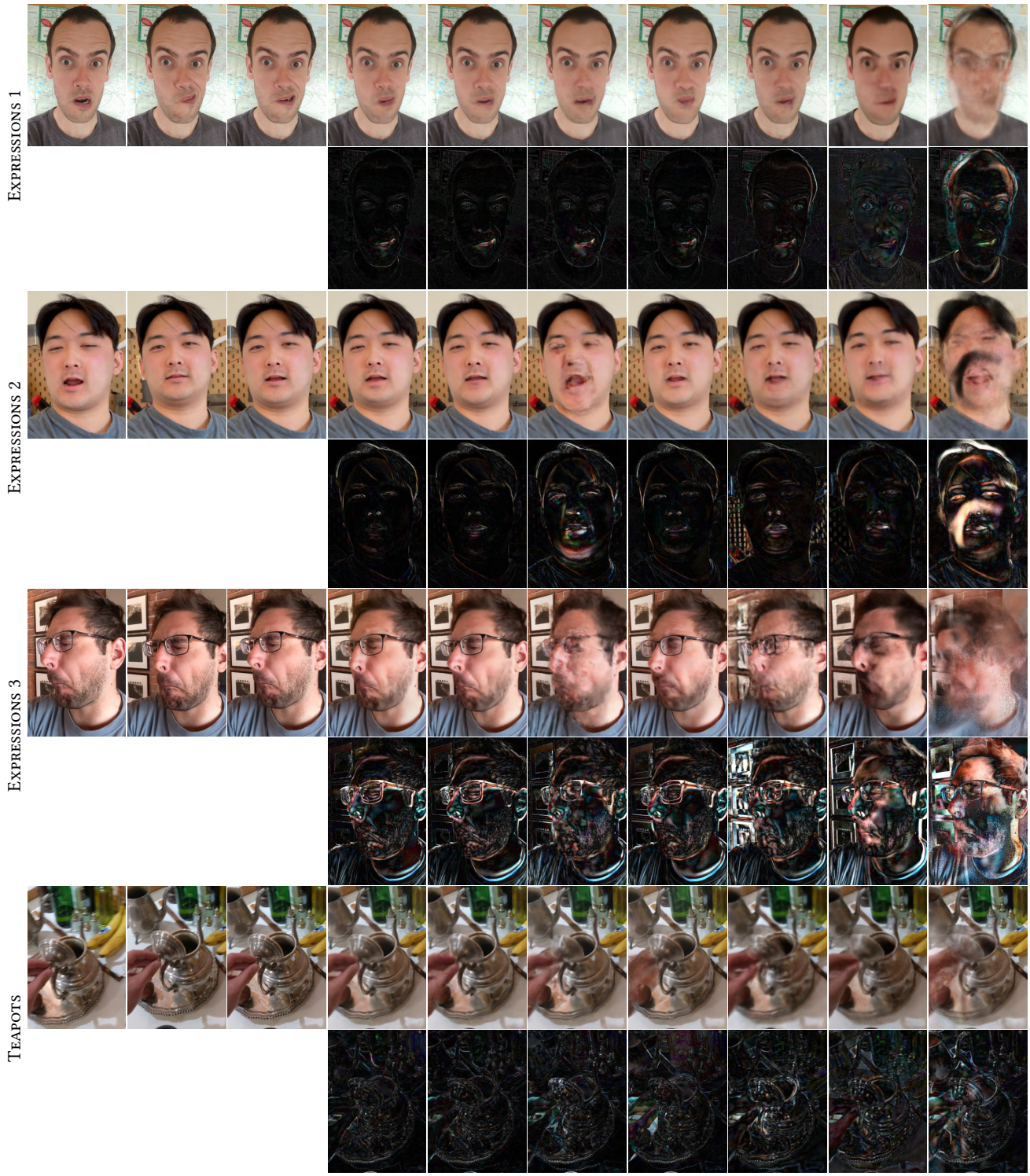


Fig. 11. Qualitative comparisons of our method, ablations, and baselines on the interpolation dataset (See Sec. 5.2.1). The interpolated frame is rendered by linearly interpolating between the latent codes of the two reference frames. The bottom row of each sequence shows the absolute error. Baselines shown are: Nerfies [Park et al. 2020], Neural Volumes [Lombardi et al. 2019], NSFF [Li et al. 2020], and NeRF [Mildenhall et al. 2020].

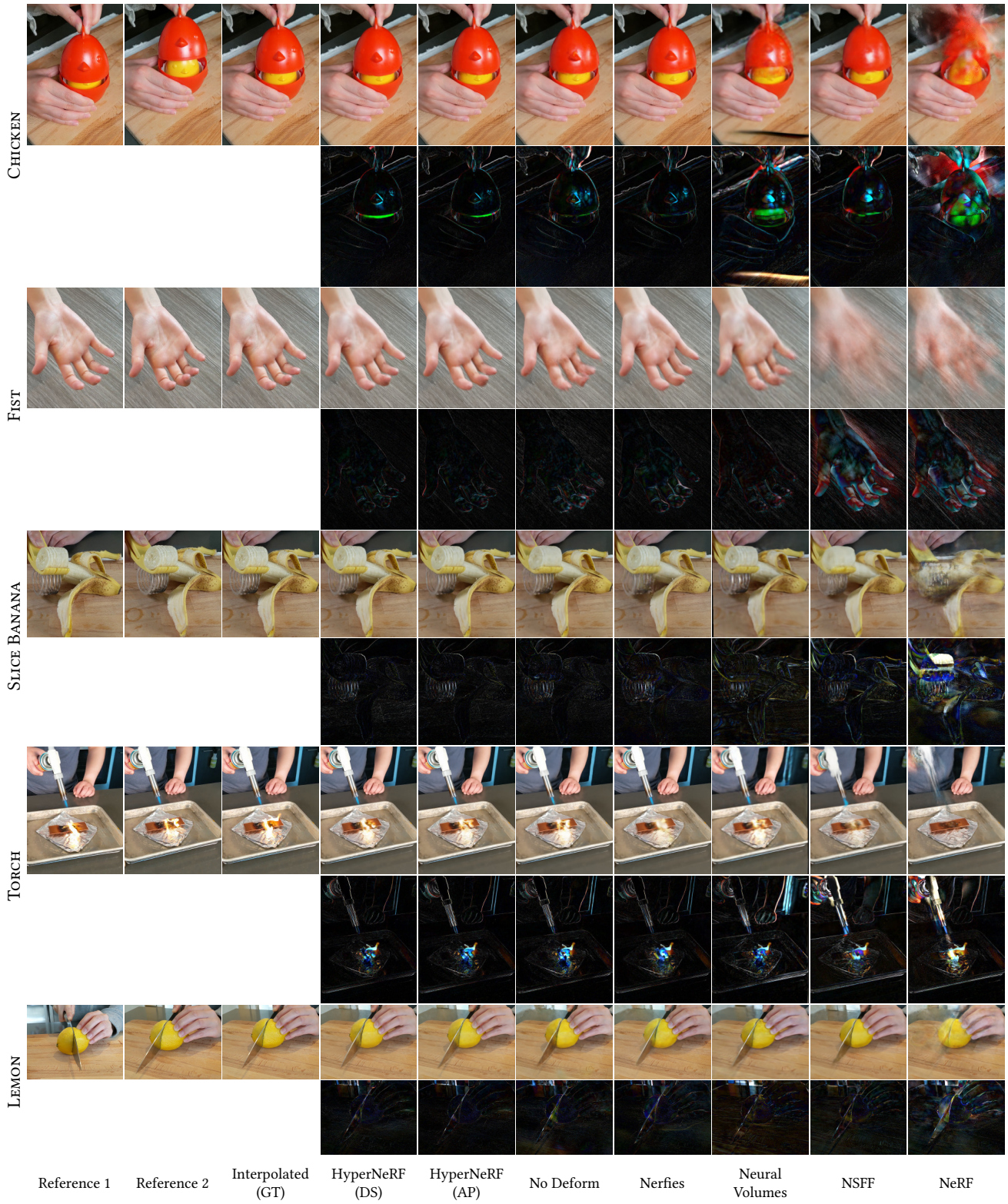


Fig. 12. (continued) Qualitative comparisons of our method, ablations, and baselines on the interpolation dataset (See Sec. 5.2.1). The interpolated frame is rendered by linearly interpolating between the latent codes of the two reference frames. The bottom row of each sequence shows the absolute error. Baselines shown are: Nerfies [Park et al. 2020], Neural Volumes [Lombardi et al. 2019], NSFF [Li et al. 2020], and NeRF [Mildenhall et al. 2020].

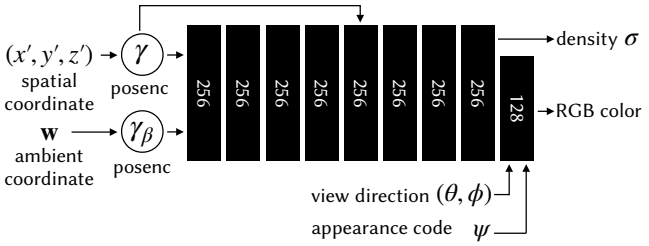


Fig. 13. A diagram of our hyper-space template network, which is identical to the original NeRF MLP, except it takes an additional ambient coordinate and an appearance latent code $\boldsymbol{\psi}$ as in Martin-Brualla et al. [2021]

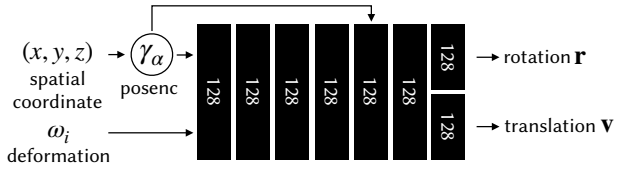


Fig. 14. A diagram of our deformation network. It is identical to the deformation MLP of Nerfies [Park et al. 2020].

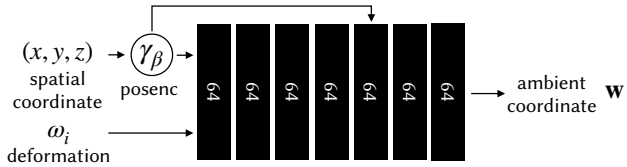


Fig. 15. A diagram of our ambient slicing surface network. It shares the input deformation code $\boldsymbol{\omega_i}$ with the deformation network. The ambient slicing surface network uses a different windowed positional encoding parameter β, and directly outputs an ambient coordinate $\mathbf{w}$.

A NETWORK ARCHITECTURE

We provide detailed architecture diagrams for the template MLP in Fig. 13, deformation MLP in Fig. 14, and the ambient slicing surface MLP in Fig. 15.

B EVALUATION

B.1 Additional Qualitative Results

We show qualitative results from the interpolation experiments in Fig. 11 and Fig. 12.

B.2 Additional Quantitative Metrics

We provide per-sequence evaluations metrics for the interpolation task: Tab. 5 reports LPIPS [Zhang et al. 2018], Tab. 4 report MS-SSIM [Wang et al. 2003], and Tab. 3 report PSNR scores for each sequence and method.

B.3 Details of the NSFF Experiments

We use an updated version of the NSFF [Li et al. 2020] code, provided by the authors, which includes numerous bug fixes and improvements. The default hyper-parameters provided did not perform well on our sequences and we therefore tune them:

β_z	$\beta_{\text{optical flow}}$	$\beta_{\text{flow smoothness}}$	β_{cyc}	β_w
0.04	0.02	0.1	1.0	0.1

In addition, the original code decays the supervised losses (depth and flow) over 25,000 iterations regardless of the length of the dataset resulting in poor performance for longer sequences. Therefore we instead decay the losses over $1000N$ iterations, where N is the number of frames in the dataset.

B.4 Details of the Neural Volumes Experiments

We closely follow the experiment procedure described in Sec. E.2. of Nerfies [Park et al. 2020].

C 2D LEVEL SET EXPERIMENTS

Here we describe our method for learning a family of topologically varying 2D signed distance functions (SDF). This method was used to generate Fig. 3.

We used truncated signed distance functions, truncated to range between -0.05 and 0.05. We learn a template MLP

$$F : (x, y, w) \to s\,, \tag{8}$$

which takes as input a normalized 2D coordinate $(x, y) \in [-1, 1]^2$ an ambient coordinate $w \in \mathbb{R}$, and outputs a signed distance s. This function defines a 3D surface which is sliced by a slicing surface. This be an axis-aligned plane, defined by a value of w defining a plane that spans the x and y axes, or a deformable slicing surface defined by an ambient slicing surface MLP

$$H : (x, y, \omega_i) \to w\,, \tag{9}$$

where (x, y) are again the spatial coordinates, $\boldsymbol{\omega_i}$ is a per-shape latent code, and w is the output ambient coordinate. We show a detailed diagram of the 2D template in Fig. 16 and deformable slicing surface in Fig. 17.

Training. Training batches are generated by randomly sampling points from the continuous, truncated SDFs of each shape i, resulting in data of the form $\{((x, y, \boldsymbol{\omega}_i), s)\}$. We use a Pseudo-Huber loss [Charbonnier et al. 1997] function:

$$L_\delta(s - s^*) = \delta^2 \left(\sqrt{1 + \left(\frac{s - s^*}{\delta} \right)^2} - 1 \right), \tag{10}$$

where s is the predicted SDF value, s^* is the ground truth SDF value, and δ is a hyper-parameter that controls the steepness which we set to $\delta = 0.005$.

Implementation Details. We use a positional encoding with a minimum degree as well as a maximum degree:

$$\gamma(\mathbf{x}) = \left[\sin(2^{m_-}\mathbf{x}), \cos(2^{m_-}\mathbf{x}), \ldots, \sin(2^{m_+-1}\mathbf{x}), \cos(2^{m_+-1}\mathbf{x}) \right]^{\mathrm{T}}, \tag{11}$$

where m_- is the minimum degree and m_+ is the maximum degree of the positional encoding. For the template MLP we $m_- = -2$ and

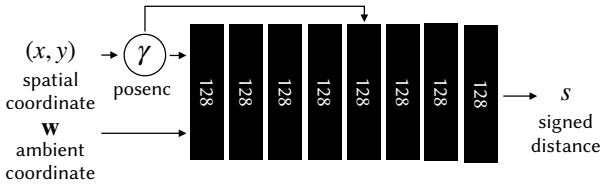


Fig. 16. A diagram of architecture for the 2D template MLP used in the 2D level set experiments.

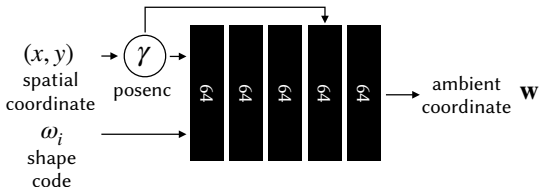


Fig. 17. A diagram of architecture for the ambient slicing surface MLP used in the 2D level set experiments.

$m_+ = 3$, for the ambient slicing surface MLP we use $m_- = -2$ and $m_+ = 2$. We train using the Adam optimizer with a fixed learning rate of 10^{-3} for 2000 iterations with a batch size of 512 which takes around a minute on a Google Colab TPU.

Interpolating Shapes. We generate interpolated shaped by linearly interpolating the shape code ω_i between instances.

REFERENCES

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

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. arXiv:2103.13415 [cs.CV]

Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. 2018. Optimizing the Latent Space of Generative Networks. *ICML* (2018).

Aljaž Božič, Pablo Palafox, Michael Zollhöfer, Angela Dai, Justus Thies, and Matthias Nießner. 2020. Neural Non-Rigid Tracking. *arXiv preprint arXiv:2006.13240* (2020).

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs.* http://github.com/google/jax

Christoph Bregler, Aaron Hertzmann, and Henning Biermann. 2000. Recovering non-rigid 3D shape from image streams. *CVPR* (2000).

Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. 1997. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing* 6, 2 (1997), 298–311.

Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In *CVPR*. 5939–5948.

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 ToG* (2015).

Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. 2020. *JaxNeRF: an efficient JAX implementation of NeRF.* https://github.com/google-research/google-research/tree/master/jaxnerf

Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. 2016. Fusion4D: Real-time performance capture of challenging scenes. *ACM ToG* (2016).

Ohad Fried, Ayush Tewari, Michael Zollhöfer, Adam Finkelstein, Eli Shechtman, Dan B Goldman, Kyle Genova, Zeyu Jin, Christian Theobalt, and Maneesh Agrawala. 2019. Text-based editing of talking-head video. *ACM TOG* (2019).

Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2021. Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction. In *CVPR*. 8649–8658.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *NeurIPS* 27 (2014).

Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. 2021. Progressive Encoding for Neural Optimization. arXiv:2104.09125 [cs.LG]

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *CVPR*.

Arthur Jacot, Franck Gabriel, and Clément Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*.

Chiyu Jiang, Jingwei Huang, Andrea Tagliasacchi, Leonidas Guibas, et al. 2020. ShapeFlow: Learnable Deformations Among 3D Shapes. *arXiv preprint arXiv:2006.07982* (2020).

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

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

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. 2021. Neural 3D Video Synthesis. arXiv:2103.02597 [cs.CV]

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

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural volumes: learning dynamic renderable volumes from images. *ACM ToG* (2019).

Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, et al. 2018. LookinGood: Enhancing performance capture with real-time neural re-rendering. *SIGGRAPH Asia* (2018).

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. *CVPR* (2021).

Abhimitra Meka, Christian Haene, Rohit Pandey, Michael Zollhoefer, Sean Fanello, Graham Fyffe, Adarsh Kowdle, Xueming Yu, Jay Busch, Jason Dourgarian, Peter Denny, Sofien Bouaziz, Peter Lincoln, Matt Whalen, Geoff Harvey, Jonathan Taylor, Shahram Izadi, Andrea Tagliasacchi, Paul Debevec, Christian Theobalt, Julien Valentin, and Christoph Rhemann. 2019. Deep Reflectance Fields - High-Quality Facial Reflectance Field Inference From Color Gradient Illumination. *SIGGRAPH.*

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

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. *ECCV* (2020).

Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. *CVPR* (2015).

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. 2019. Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics. *ICCV* (2019).

Stanley Osher and James A Sethian. 1988. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics* 79, 1 (1988), 12–49.

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

Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. 2020. Nerfies: Deformable Neural Radiance Fields. *arXiv preprint arXiv:2011.12948.*

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2020. D-NeRF: Neural Radiance Fields for Dynamic Scenes. *arXiv preprint arXiv:2011.13961* (2020).

Tanner Schmidt, Richard Newcombe, and Dieter Fox. 2015. DART: dense articulated real-time tracking with consumer depth cameras. *Autonomous Robots* (2015).

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. *CVPR* (2016).

Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. 2019a. DeepVoxels: Learning Persistent 3D Feature Embeddings. In *CVPR.*

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. 2019b. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In *NeurIPS.*

Tiancheng Sun, Jonathan T Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul E Debevec, and Ravi Ramamoorthi. 2019.

Table 3. Per-sequence PSNR metrics on the interpolation task. We color code each row as **best**, **second best**, and **third best**.

	Expressions 1 (391 images) PSNR↑	Expressions 2 (126 images) PSNR↑	Expressions 3 (110 images) PSNR↑	Teapots (243 images) PSNR↑	Chicken (113 images) PSNR↑	Fist (433 images) PSNR↑	Slice Banana (82 images) PSNR↑	Torch (173 images) PSNR↑	Lemon (415 images) PSNR↑	Mean PSNR↑
NeRF [Mildenhall et al. 2020]	21.6	21.1	15.6	23.6	18.8	23.8	20.8	22.5	24.1	21.3
NV [Lombardi et al. 2019]	26.7	25.6	18.6	26.2	22.6	29.3	24.8	24.6	28.8	25.3
NSFF [Li et al. 2020]	26.6	29.8	18.3	25.8	27.7	24.9	26.1	22.3	28.0	25.5
Nerfies [Park et al. 2020]	27.5	31.6	19.7	25.7	28.7	29.9	27.9	27.8	30.8	27.7
Nerfies (w/o elastic)	27.7	32.5	20.0	25.8	28.8	30.5	28.1	27.9	30.8	28.0
Hyper-NeRF (DS)	27.9	32.9	20.1	26.4	28.7	30.7	28.4	28.0	31.8	28.3
Hyper-NeRF (DS, w/ elastic)	27.8	32.4	19.9	26.4	28.8	30.2	28.3	27.9	31.4	28.1
Hyper-NeRF (AP)	27.9	33.1	20.0	26.3	28.9	30.5	28.4	28.0	31.3	28.3
Hyper-NeRF (w/o deform)	26.9	31.1	19.4	26.0	27.7	29.3	28.1	27.3	30.5	27.4

Table 4. Per-sequence MS-SSIM metrics on the interpolation task. We color code each row as **best**, **second best**, and **third best**.

	Expressions 1 (391 images) MS-SSIM↑	Expressions 2 (126 images) MS-SSIM↑	Expressions 3 (110 images) MS-SSIM↑	Teapots (243 images) MS-SSIM↑	Chicken (113 images) MS-SSIM↑	Fist (433 images) MS-SSIM↑	Slice Banana (82 images) MS-SSIM↑	Torch (173 images) MS-SSIM↑	Lemon (415 images) MS-SSIM↑	Mean MS-SSIM↑
NeRF [Mildenhall et al. 2020]	.708	.722	.449	.875	.761	.773	.721	.866	.826	.745
NV [Lombardi et al. 2019]	.900	.910	.647	.917	.861	.912	.907	.917	.951	.880
NSFF [Li et al. 2020]	.885	.933	.654	.915	.939	.797	.858	.883	.904	.863
Nerfies [Park et al. 2020]	.877	.963	.698	.922	.948	.940	.916	.959	.946	.908
Nerfies (w/o elastic)	.883	.967	.698	.920	.947	.946	.915	.961	.946	.909
Hyper-NeRF (DS)	.887	.970	.703	.931	.948	.950	.920	.962	.956	.914
Hyper-NeRF (DS, w/ elastic)	.885	.966	.700	.932	.949	.943	.918	.960	.952	.912
Hyper-NeRF (AP)	.885	.970	.703	.927	.949	.947	.917	.960	.947	.912
Hyper-NeRF (w/o deform)	.857	.947	.669	.922	.936	.924	.915	.951	.937	.895

Table 5. Per-sequence LPIPS metrics on the interpolation task. We color code each row as **best**, **second best**, and **third best**.

	Expressions 1 (391 images) LPIPS↓	Expressions 2 (126 images) LPIPS↓	Expressions 3 (110 images) LPIPS↓	Teapots (243 images) LPIPS↓	Chicken (113 images) LPIPS↓	Fist (433 images) LPIPS↓	Slice Banana (82 images) LPIPS↓	Torch (173 images) LPIPS↓	Lemon (415 images) LPIPS↓	Mean LPIPS↓
NeRF [Mildenhall et al. 2020]	.582	.492	.747	.339	.453	.469	.513	.373	.437	.490
NV [Lombardi et al. 2019]	.215	.130	.318	.216	.243	.213	.209	.189	.190	.214
NSFF [Li et al. 2020]	.283	.134	.317	.210	.173	.329	.243	.253	.238	.242
Nerfies [Park et al. 2020]	.224	.0952	.275	.225	.141	.171	.209	.169	.223	.193
Nerfies (w/o elastic)	.219	.0857	.279	.229	.160	.158	.200	.168	.239	.193
Hyper-NeRF (DS)	.218	.0825	.274	.212	.156	.150	.191	.172	.210	.185
Hyper-NeRF (DS, w/ elastic)	.220	.0956	.269	.212	.145	.162	.249	.173	.230	.195
Hyper-NeRF (AP)	.240	.0894	.290	.234	.162	.158	.241	.183	.276	.208
Hyper-NeRF (w/o deform)	.312	.131	.352	.238	.203	.212	.299	.224	.303	.253

Table 6. Per-sequence "average" metrics [Barron et al. 2021] on the interpolation task. We color code each row as **best**, **second best**, and **third best**.

	Expressions 1 (391 images) Avg↓	Expressions 2 (126 images) Avg↓	Expressions 3 (110 images) Avg↓	Teapots (243 images) Avg↓	Chicken (113 images) Avg↓	Fist (433 images) Avg↓	Slice Banana (82 images) Avg↓	Torch (173 images) Avg↓	Lemon (415 images) Avg↓	Mean Avg↓
NeRF [Mildenhall et al. 2020]	.130	.127	.248	.0809	.143	.0973	.131	.0916	.0889	.126
NV [Lombardi et al. 2019]	.0524	.0475	.138	.0532	.0791	.0419	.0596	.0574	.0381	.0630
NSFF [Li et al. 2020]	.0596	.0331	.140	.0546	.0416	.0780	.0608	.0798	.0490	.0663
Nerfies [Park et al. 2020]	.0519	.0234	.118	.0552	.0351	.0350	.0462	.0385	.0351	.0487
Nerfies (w/o elastic)	.0501	.0206	.115	.0554	.0366	.0321	.0447	.0377	.0359	.0476
Hyper-NeRF (DS)	.0494	.0194	.114	.0504	.0363	.0305	.0426	.0377	.0308	.0456
Hyper-NeRF (DS, w/ elastic)	.0499	.0217	.115	.0501	.0352	.0332	.0471	.0383	.0332	.0471
Hyper-NeRF (AP)	.0509	.0196	.116	.0527	.0361	.0319	.0464	.0386	.0361	.0476
Hyper-NeRF (w/o deform)	.0623	.0287	.132	.0552	.0442	.0410	.0513	.0453	.0406	.0557

Single image portrait relighting. *SIGGRAPH* (2019).
Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *NeurIPS* (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* (2020).

Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred neural rendering: Image synthesis using neural textures. *ACM TOG* (2019).

Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2016. Face2Face: Real-time face capture and reenactment of rgb videos. *CVPR* (2016).

Lorenzo Torresani, Aaron Hertzmann, and Chris Bregler. 2008. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *TPAMI* (2008).

Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. 2021. Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video. arXiv:2012.12247 [cs.CV]

Zhou Wang, Eero P Simoncelli, and Alan C Bovik. 2003. Multiscale structural similarity for image quality assessment. *Asilomar Conference on Signals, Systems & Computers* (2003).

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

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. 2020. Multiview neural surface reconstruction by disentangling geometry and appearance. *arXiv preprint arXiv:2003.09852* (2020).

Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. 2020. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *CVPR*.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*.