

Nerfies: Deformable Neural Radiance Fields

Keunhong Park¹, Utkarsh Sinha², Jonathan T. Barron², Sofien Bouaziz², Dan B Goldman²,
Steven M. Seitz^{1,2}, Ricardo Martin-Brualla²

Abstract

We present the first method capable of photorealistically reconstructing deformable scenes using photos/videos captured casually from mobile phones. Our approach augments neural radiance fields (NeRF) by optimizing an additional continuous volumetric deformation field that warps each observed point into a canonical 5D NeRF. We observe that these NeRF-like deformation fields are prone to local minima, and propose a coarse-to-fine optimization method for coordinate-based models that allows for more robust optimization. By adapting principles from geometry processing and physical simulation to NeRF-like models, we propose an elastic regularization of the deformation field that further improves robustness. We show that our method can turn casually captured selfie photos/videos into deformable NeRF models that allow for photorealistic renderings of the subject from arbitrary viewpoints, which we dub “nerfies.” We evaluate our method by collecting time-synchronized data using a rig with two mobile phones, yielding train/validation images of the same pose at different viewpoints. We show that our method faithfully reconstructs non-rigidly deforming scenes and reproduces unseen views with high fidelity.

¹ University of Washington

² Google Research

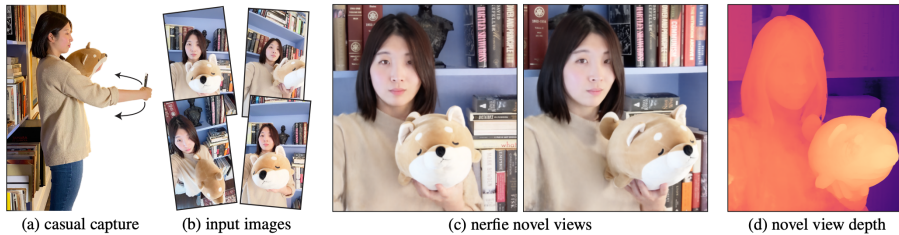


Figure 1: We reconstruct photo-realistic *nerfies* from a user casually waving a mobile phone. Our system uses selfie photos/videos to produce a free-viewpoint representation with accurate renders and geometry. Please see video.

1. Introduction

High quality 3D human scanning has come a long way — but the best results currently require a specialized lab with many synchronized lights and cameras, e.g., [1], [2], [3]. What if you could capture a photorealistic model of yourself (or someone else) just by waving your mobile phone camera? Such a capability would dramatically increase accessibility and applications of 3D modeling technology.

Modeling people with hand-held cameras is especially challenging due both to 1) nonrigidity — our inability to stay perfectly still, and 2) challenging materials like hair, glasses, and earrings that violate assumptions used in most reconstruction methods. In this paper we introduce an approach to address both of these challenges, by generalizing Neural Radiance Fields (NeRF) [4] to model shape deformations. Our technique recovers high fidelity 3D reconstructions from short videos, providing free-viewpoint visualizations while accurately capturing hair, glasses, and other complex, view-dependent materials, as shown in Figure 1. A special case of particular interest is capturing a 3D self-portrait — we call such casual 3D selfie reconstructions *nerfies*.

Rather than represent shape explicitly, NeRF [4] uses a neural network to encode color and density as a function of location and viewing angle, and generates novel views using volume rendering. Their approach produces 3D visualizations of unprecedented quality, faithfully representing thin structures, semi-transparent materials, and view-dependent effects. To model non-rigidly deforming scenes, we generalize NeRF by introducing an additional component: A canonical NeRF model serves as a template for all the observations, supplemented by a deformation field for each observation that warps 3D points in the frame of reference of an observation into the frame of reference of the canonical model. We represent this deformation field as a multi-layer perceptron (MLP), similar to the radiance field in NeRF. This deformation field is conditioned on a per-image learned latent code, allowing it to vary between observations.

Without constraints, the deformation fields are prone to distortions and overfitting. We employ a similar approach to the elastic energy formulations that have seen success for mesh fitting [5], [6], [7], [8]. However, our volumetric deformation field formulation greatly simplifies such regularization, because we can easily compute the Jacobian of the deformation field through automatic differentiation, and directly regularize its singular values.

To robustly optimize the deformation field, we propose a novel coarse-to-fine optimization scheme that modulates the components of the input positional encoding of the deformation field network by frequency. By zeroing out the high frequencies at the start of optimization, the network is limited to learn smooth deformations, which are later refined as higher frequencies are introduced into the optimization.

For evaluation, we capture image sequences from a rig of two synchronized, rigidly attached, calibrated cameras, and use the reconstruction from one camera

to predict views from the other. We plan to release the code and data.

In summary, our contributions are:

1. An extension to NeRF to handle non-rigidly deforming objects that optimizes a deformation field per observation.
2. Rigidity priors suitable for deformation fields defined by neural networks.
3. A coarse-to-fine regularization approach that modulates the capacity of the deformation field during optimization.
4. A system to reconstruct free-viewpoint selfies from casual mobile phone captures.

2. Related Work

Non-Rigid Reconstruction: Non-rigid reconstruction decomposes a scene into a geometric model and a deformation model. Earlier works focused on sparse representations such as keypoints projected onto 2D images [9], [10], making the problem highly ambiguous. Multi-view captures [1], [2] simplify the problem to one of registering and fusing 3D scans [11]. DynamicFusion [12] uses a single RGBD camera moving in space, solving jointly for a canonical model, deformation, and camera pose. More recently, learning-based methods have been used to find correspondences useful for non-rigid reconstruction [13], [14]. Unlike prior work, our method does not require depth nor multi-view capture and works on monocular RGB inputs. Most similar to our work, Neural Volumes [15] learns a 3D representation of a deformable scene using a voxel grid and warp field regressed from a 3D CNN. However, their method requires dozens of synchronized cameras and our evaluation shows that it does not extend to sequences captured from a single camera. Yoon et al. [16] reconstruct dynamic scenes from moving camera trajectories, but their method relies on strong semantic priors, in the form of monocular depth estimation, which are combined with multi-view cues. OFlow [17] solves for temporal flow-fields using ODEs, and thus requires temporal information. ShapeFlow [18] learns 3D shapes a divergence-free deformations of a learned template. Instead, we propose an elastic energy regularization.

Domain-Specific Modeling: Many reconstruction methods use domain-specific knowledge to model the shape and appearance of categories with limited topological variation, such as faces [19], [20], [21], human bodies [22], [23], and animals [24], [25]. Although some methods show impressive results in monocular face reconstruction from color and RGBD cameras [26], such models often lack detail (e.g., hair), or do not model certain aspects of a category (e.g., eyewear or garments). Recently, image translation networks have been applied to improve the realism of composited facial edits [27], [28]. In contrast, our work does not rely on domain-specific knowledge, enabling us to model the whole scene, including eyeglasses and hair for human subjects.

Coordinate-based Models: Our method builds on the recent success of coordinate-based models, which encode a spatial field in the weights of a mul-

tilayer perceptron (MLP) and require significantly less memory compared to discrete representations. These methods have been used to represent shapes [29], [30], [31] and scenes [4], [32]. Of particular interest are NeRFs [4], that use periodic positional encoding layers [33], [34] to increase resolution, and whose formulation has been extended to handle different lighting conditions [35], [36], transient objects [36], large scenes [37], [38] and to model object categories [39]. Our work extends NeRFs to handle non-rigid scenes.

Concurrent Work: Two concurrent works [40], [41] propose to represent deformable scenes using a translation field in conjunction with a template. This is similar to our framework with the following differences:

1. we condition the deformation with a per-example latent [42] instead of time [40];
2. propose an as-rigid-as-possible regularization of the deformation field while NR-NeRF [41] penalizes the divergence of the translation field;
3. propose a coarse-to-fine regularization to prevent getting stuck in local minima; and
4. propose an improved SE(3) parameterization of the deformation field. Other concurrent works [43], [44] reconstruct space-time videos by recovering time-varying NeRFs while leveraging external supervision such as monocular depth estimation and flow-estimation to resolve ambiguities.

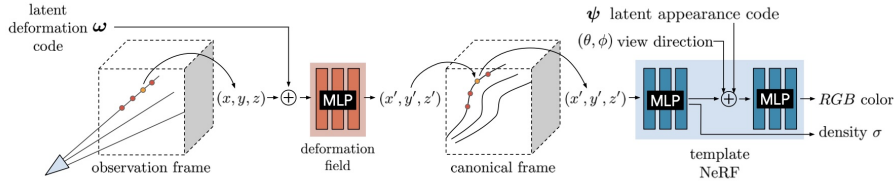


Figure 2: We associate a latent deformation code ω and an appearance code ψ to each image. Camera rays are traced in the observation frame and transformed to the canonical frame using a deformation field MLP conditioned on ω .

3. Deformable Neural Radiance Fields

Here we describe our method for modeling non-rigidly deforming scenes given a set of casually captured images of the scene. We decompose a non-rigidly deforming scene into a template volume represented as a neural radiance field (NeRF) [4] (§3.1) and a per-observation deformation field (§3.2) that associates a point in observation coordinates to a point on the template (overview in Figure 2). The deformation field is our key extension to NeRF and allows us to represent moving subjects. Jointly optimizing a NeRF together with a deformation field leads to an under-constrained optimization problem. We therefore introduce an elastic regularization on the deformation (§3.3), a background regularization

(§3.4), and a continuous, coarse-to-fine annealing technique that avoids bad local minima (§3.5).

3.1 Neural Radiance Fields

A neural radiance field (NeRF) is a continuous, volumetric representation. It is a function $F : (\mathbf{x}, \mathbf{d}, \psi_i) \rightarrow (\mathbf{c}, \sigma)$ which maps a 3D position $\mathbf{x} = (x, y, z)$ and viewing direction $\mathbf{d} = (\varphi, \theta)$ to a color $\mathbf{c} = (r, g, b)$ and density σ . In practice, NeRF maps the inputs $\mathbf{x}$ and $\mathbf{d}$ using a sinusoidal positional encoding $\gamma : \mathbb{R}^3 \rightarrow \mathbb{R}^{3+6m}$ defined as $\gamma(\mathbf{x}) = (\mathbf{x}, \dots, \sin(2^k \pi \mathbf{x}), \cos(2^k \pi \mathbf{x}), \dots)$, where m is a hyper-parameter that controls the total number of frequency bands and $k \in \{0, \dots, m-1\}$. This function projects a coordinate vector $x \in \{\mathbb{R}^3\}$ to a high dimensional space using a set of sine and cosine functions of increasing frequencies. This allows the MLP to model high-frequency signals in low-frequency domains as shown in [34]. Coupled with volume rendering techniques, NeRFs can represent scenes with photo-realistic quality. We build upon NeRF to tackle the problem of capturing deformable scenes.

Similar to NeRF-W [36], we also provide an appearance latent code ψ for each observed frame $i \in \{1, \dots, n\}$ that modulates the color output to handle appearance variations between input frames, e.g., exposure and white balance.

The NeRF training procedure relies on the fact that given a 3D scene, two intersecting rays from two different cameras should yield the same color. Disregarding specular reflection and transmission, this assumption is true for all static scenes. Unfortunately, many scenes are not completely static; e.g., it is hard for people to stay completely still when posing for a photo, or worse, when waving a phone when capturing themselves in a selfie video.

3.2 Neural Deformation Fields

With the understanding of this limitation, we extend NeRF to allow the reconstruction of non-rigidly deforming scenes. Instead of directly casting rays through a NeRF, we use it as a canonical template of the scene. This template contains the relative structure and appearance of the scene while a rendering will use a non-rigidly deformed version of the template (see Figure 3 for an example). DynamicFusion [12] and Neural Volumes [15] also model a template and a per-frame deformation, but the deformation is defined on mesh points and on a voxel grid respectively, whereas we model it as a continuous function using an MLP.

We employ an observation-to-canonical deformation for every frame $i \in \{1, \dots, n\}$, where n is the number of observed frames. This defines a mapping $T_i : \mathbf{x} \rightarrow \mathbf{x}'$ that maps all observation-space coordinates x to a canonical-space coordinate $\mathbf{x}'$. We model the deformation fields for all time steps using a mapping $T_i : \mathbf{x}, \omega_i \rightarrow \mathbf{x}'$, which is conditioned on a per-frame learned latent deformation code ω_i . Each latent code encodes the state of the scene in

frame i . Given a canonical-space radiance field F and a observation-to-canonical mapping T , the observation-space radiance field can be evaluated as:

$$G(\mathbf{x}, \mathbf{d}, \psi_i, \omega_i) = F(T(\mathbf{x}, \omega_i), \mathbf{d}, \psi_i)$$

When rendering, we simply cast rays and sample points in the observation frame and then use the deformation field to map the sampled points to the template, see Figure 2.

A simple model of deformation is a displacement field $\mathbf{V} : (\mathbf{x}, \omega_i) \rightarrow \mathbf{t}$, defining the transformation as $\mathbf{T}(\mathbf{x}, \omega_i) = \mathbf{x} + \mathbf{V}(\mathbf{x}, \omega_i)$. This formulation is sufficient to represent all continuous deformations; however, rotating a group of points with a translation field requires a different translation for each point, making it difficult to rotate regions of the scene

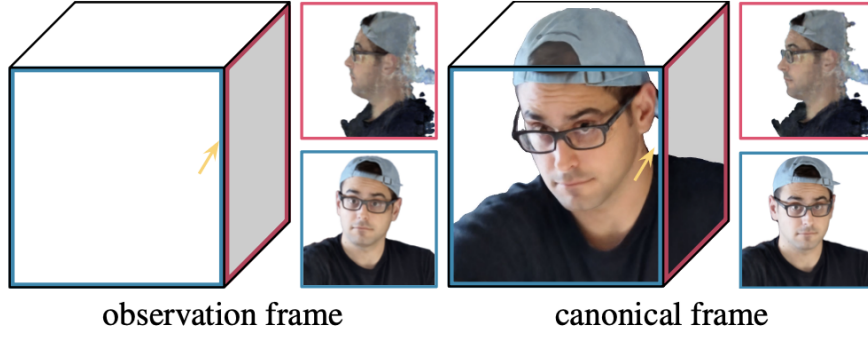


Figure 3: Visualizations of the recovered 3D model in the observation and canonical frames of reference, with insets showing orthographic views in the forward and left directions. Note the right-to-left and front-to-back displacements between the observation and canonical model, which are modeled by the deformation field for this observation.

simultaneously. We therefore formulate the deformation using a dense $SE(3)$ field $W : (\mathbf{x}, \omega_i) \rightarrow SE(3)$. An $SE(3)$ transform encodes rigid motion, allowing us to rotate a set of distant points with the same parameters.

We encode a rigid transform as a screw axis [45] $\mathcal{S} = (\mathbf{r}; \mathbf{v}) \in \mathbb{R}^6$. Note that $\mathbf{r} \in \mathfrak{so}(3)$ encodes a rotation where $\hat{\mathbf{r}} = \mathbf{r}/\|\mathbf{r}\|$ is the axis of rotation and $\theta = \|\mathbf{r}\|$ is the angle of rotation. The exponential of $\mathbf{r}$ (also known as Rodrigues' formula [46]) yields a rotation matrix $e^{\mathbf{r}} \in SO(3)$:

$$e^{\mathbf{r}} \equiv [\mathbf{r}]_{\times} = \mathbf{I} + \frac{\sin \theta}{\theta} [\mathbf{r}]_{\times} + \frac{1 - \cos \theta}{\theta^2} [\mathbf{r}]_{\times}^2$$

where $[\mathbf{x}]_{\times}$ denotes the cross-product matrix of a vector x .

Similarly, the translation encoded by the screw motion S can be recovered as $\mathbf{p} = \mathbf{G}\mathbf{v}$ where

$$\mathbf{G} = \mathbf{I} + \frac{1 - \cos \theta}{\theta^2} [\mathbf{r}]_{\times} + \frac{\theta - \sin \theta}{\theta^3} [\mathbf{r}]_{\times}^2$$

Combining these formulas and using the exponential map, we get the transformed point as $\mathbf{x}' = \mathbf{e}^{\mathbf{r}} \mathbf{x} + \mathbf{p}$.

As mentioned before, we encode the transformation field in an MLP $\mathbf{W} : (\mathbf{x}, \omega_i) \rightarrow (\mathbf{r}, \mathbf{v})$ using a NeRF-like architecture, and represent the transformation of every frame i by conditioning on a latent code ω_i . We optimize the latent code through an embedding layer [42]. Like with the template, we map the input x using positional encoding γ_{α} (see §3.5). An important property of the $\text{se}(3)$ representation is that e^S is the identity when $S = \mathbf{O}$. We therefore initialize the weights of the last layer of the MLP from $\mathbf{U}(-10^{-5}, 10^{-5})$ to initialize the deformation near the identity.

3.3 Elastic Regularization

The deformation field adds ambiguities that make optimization more challenging. For example, an object moving backwards is visually equivalent to it shrinking in size, with many solutions in between. These ambiguities lead to underconstrained optimization problems which yield implausible results and artifacts (see Figure 6). It is therefore crucial to introduce priors that lead to a more plausible solution.

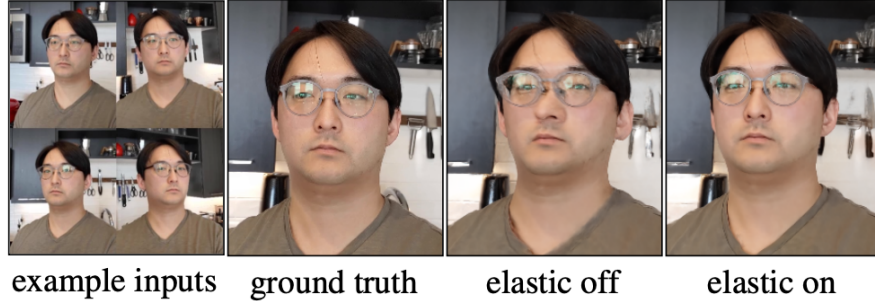


Figure 4: Our elastic regularization helps when the scene is under-constrained. With only 20 input images biased towards one side of the face, elastic regularization resolves ambiguity and reduces distortion.

It is common in geometry processing and physics simulation to model non-rigid deformations using elastic energies measuring the deviation of local deformations from a rigid motion [5], [6], [7], [8]. In the vision community, these energies have been extensively used for the reconstruction and tracking of non-rigid scenes and objects [2], [12], [47] making them good candidates for our approach. While

they have been most commonly used for discretized surfaces, e.g., meshes, we can apply a similar concept in the context of our continuous deformation field.

Elastic Energy: For a fixed latent code ω_i , our deformation field T is a non-linear mapping from observation-coordinates in $\mathbb{R}^3$ to canonical coordinates in $\mathbb{R}^3$. The Jacobian $J_T(\mathbf{x})$ of this mapping at a point $\mathbf{x} \in \mathbb{R}^3$ describes the best linear approximation of the transformation at that point. We can therefore control the local behavior of the deformation through J_T [48]. Note that unlike other approaches using discretized surfaces, our continuous formulation allows us to directly compute J_T through automatic differentiation of the MLP. There are several ways to penalize the deviation of the Jacobian J_T from a rigid transformation. Considering the singular-value decomposition of the Jacobian $J_T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, multiple approaches [5], [6] penalize the deviation from the closest rotation as $\|J_T - \mathbf{R}\|_F^2$, where $\mathbf{R} = \mathbf{V}\mathbf{U}^T$ and $\|\cdot\|_F$ is the Frobenius norm. We opt to directly work with the singular values of J_T and measure its deviation from the identity. We opt to directly work with the singular values of $\mathbf{J}\mathbf{T}$ and measure its deviation from the identity. The log of the singular values gives equal weight to a contraction and expansion of the same factor, and we found it to perform better. We therefore penalize the deviation of the log singular values from zero:

$$\mathcal{L}_{\text{elastic}}(\mathbf{x}) = \|\log \mathbf{\Sigma}\|_F^2$$

where $\log$ here is the matrix logarithm.

Robustness: Although humans are mostly rigid, there are some movements which can break our assumption of local rigidity, e.g., facial expressions which locally stretch and compress our skin. We therefore remap the elastic energy

defined above using a robust loss:

$$L_{\text{elastic-r}}(\mathbf{x}) = \rho(\|\log \mathbf{\Sigma}\|_F, c)$$

$$\rho(x, c) = \frac{2(x/c)^2}{(x/c)^2 + 4}$$

where $\rho(\hat{\mathbf{u}})$ is the Geman-McClure robust error function [49] parameterized with hyperparameter $c = 0.03$ as per Barron [50]. This robust error function causes the gradients of the loss to fall off to zero for large values of the argument, thereby reducing the influence of outliers during training.

Weighting: We allow the deformation field to behave freely in empty space, since the subject moving relative to the background requires a non-rigid deformation somewhere in space. We therefore weight the elastic penalty at each sample along the ray by its contribution to the rendered view, i.e. $\mathbf{w}_i$ in Eqn. 5 of NeRF [4].

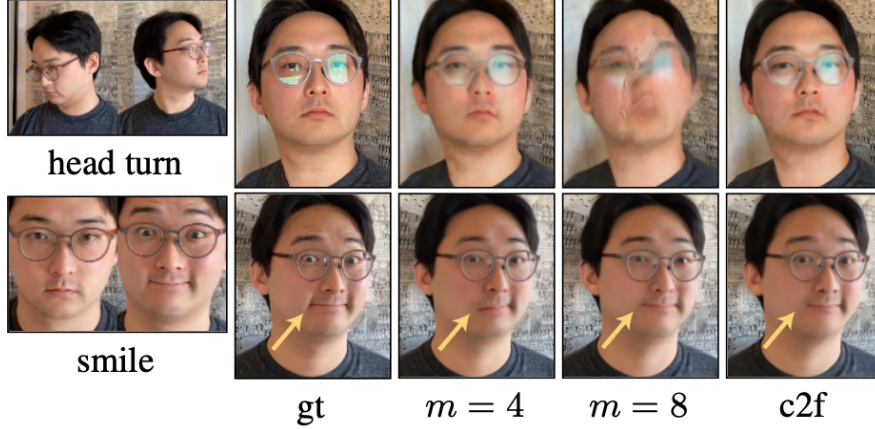


Figure 5: With $\mathbf{m} = 4$ positional encoding frequencies, the model does not capture the smile. With $\mathbf{m} = 8$, it fails to rotate the head. With coarse-to-fine (c2f) regularization, the model captures both.

3.4 Background Regularization

The deformation field is unconstrained and therefore everything is free to move around. We optionally add a regularization term which prevents the background from moving. Given a set of 3D points in the scene which we know should be static, we can penalize any deformations at these points. For example, camera registration using structure from motion produces a set of 3D feature points that behave rigidly across at least some set of observations. Given these static 3D points $\mathbf{x}_1 \dots, \mathbf{x}_K$, we penalize movement as:

$$\mathcal{L}_{\text{bg}}(\mathbf{x}) = \|\log \Sigma\|_F^2$$

In addition to keeping the background points from moving, this regularization also has the benefit of aligning the observation coordinate frame to the canonical coordinate frame.

3.5 Coarse-to-Fine Deformation Regularization

A common trade-off that arises during registration and flow estimation is the choice between modeling minute versus large motions, that can lead to overly smooth results or

incorrect registration (local minima). Coarse-to-fine strategies circumvent the issue by first solving the problem in low-resolution, where motion is small, and iteratively up-scaling the solution and refining it [51]. We observe that our deformation model suffers from similar issues, and propose a coarse-to-fine regularization to mitigate them.

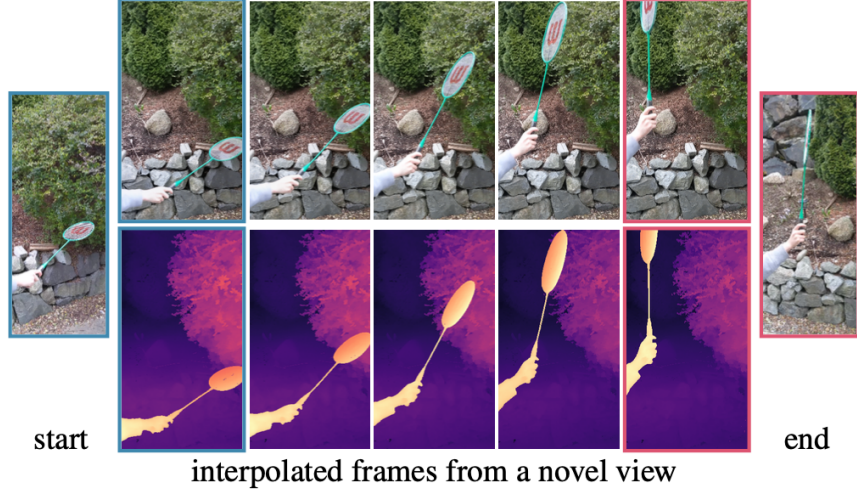


Figure 6: Novel views synthesized by linearly interpolating the deformation latent codes of two frames (left and right) show a smooth motion trajectory.

Recall the positional encoding parameter m introduced in §3.1 that controls the number of frequency bands used in the encoding. Tancik et al. [34] show that controls it the smoothness of the network: a low value of m results in a low-frequency bias (low resolution) while a higher value of m results in a higher-frequency bias (high resolution).

Consider a motion like in Figure 5, where subject rotates their head and smiles. With a small m for the deformation field, the model cannot capture the minute motion of the smile; conversely, with a larger m , the model fails to correctly rotate the head because the template overfits to an underoptimized deformation field. To overcome this trade-off, we propose a coarse-to-fine approach that starts with a low-frequency bias and ends with a high-frequency bias.

Tancik et al. [34] show that positional encoding can be interpreted in terms of the Neural Tangent Kernel (NTK) [52] of NeRF’s MLP: a stationary interpolating kernel where m controls a tunable “bandwidth” of that kernel. A small number of frequencies induces a wide kernel which causes under-fitting of the data, while a large number of frequencies induces a narrow kernel causing over-fitting. With this in mind, we propose a method to smoothly anneal the bandwidth of the NTK by introducing a parameter α that windows the frequency bands of the positional encoding, akin to how coarse-to-fine optimization schemes solve for coarse solutions that are subsequently refined at higher resolutions. We define the weight for each frequency band j as:

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

where linearly annealing the parameter $\alpha \in [0, \mathbf{m}]$ can be interpreted as sliding a truncated Hann window (where the left side is clamped to 1 and

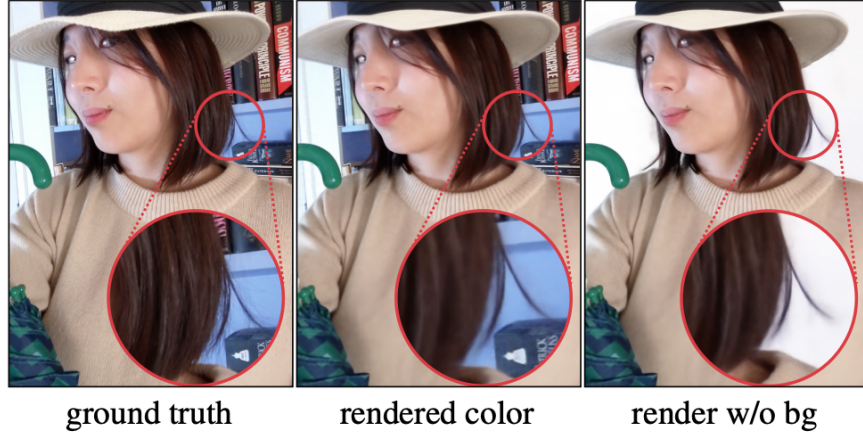


Figure 7: With $\mathbf{m} = 4$ positional encoding frequencies, the model does not capture the smile. With $\mathbf{m} = 8$, it fails to rotate the head. With coarse-to-fine (c2f) regularization, the model captures both.

the right side is clamped to 0) across the frequency bands. The positional encoding is then defined as $\gamma_\alpha(\mathbf{x}) = (x, \dots, w_k(\alpha) \sin(2^k \pi x), w_k(\alpha) \cos(2^k \pi x), \dots)$. During training, we set $\alpha(t) = \frac{mt}{N}$ where t is the current training iteration, and N is a hyper-parameter for when α should reach the maximum number of frequencies m . We provide further analysis in the supplementary materials.

4. Nerfies: Casual Free-Viewpoint Selfies

So far we have presented a generic method of reconstructing non-rigidly deforming scenes. We now present a key application of our system – reconstructing high quality models of human subjects from casually captured selfies, which we dub “nerfies”. Our system takes as input a sequence of selfie photos or a selfie video in which the user is standing mostly still. Users are instructed to wave the camera around their face, covering viewpoints within a 45° cone. We observe that 20 second captures are sufficient. In our method, we assume that the subject stands against a static background to enable a consistent geometric registration of the cameras. We filter blurry frames using the variance of the Laplacian [53], keeping about 600 frames per capture.

Camera Registration: We seek a registration of the cameras with respect to the static background. We use COLMAP [54] to compute pose for each image and camera intrinsics. This step assumes that enough features are present in the background to register the sequence.

Foreground Segmentation: In some cases, SfM will match features on the moving subject, causing significant misalignment in the background. This is problematic in video captures with correlated frames. In those cases, we found it helpful to discard image features on the subject, which can be detected using a foreground segmentation network.

5. Experiments

5.1. Implementation Details

Our NeRF template implementation closely follows the original [4], except we use a Softplus activation $\ln(1 + e^x)$ for the density. We use a deformation network with

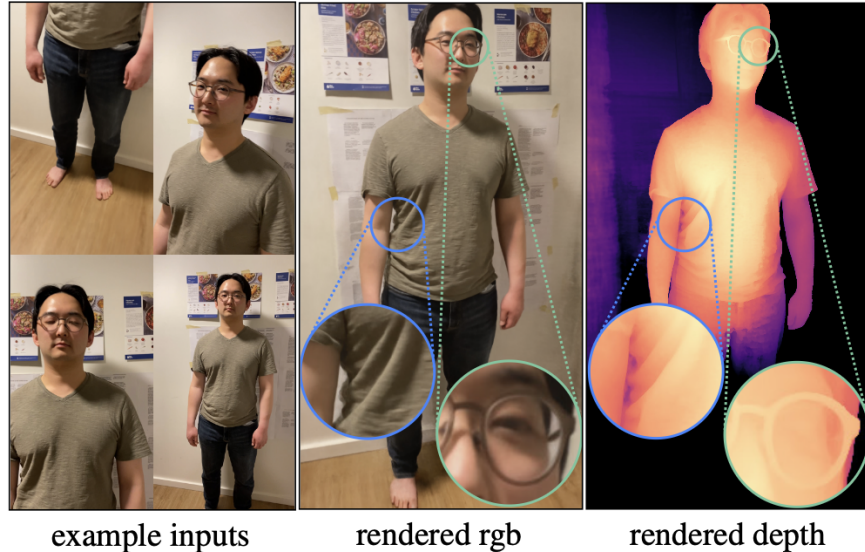


Figure 8: With $m = 4$ positional encoding frequencies, the model does not capture the smile. With $m = 8$, it fails to rotate the head. With coarse-to-fine (c2f) regularization, the model captures both.

depth 6, hidden size 128, and a skip connection at the 4th layer. We use 256 coarse and fine ray samples for full HD (1920×1080) models and half that for the half resolution models. We use 8 dimensions for the latent deformation and appearance codes. For coarse-to-fine optimization we use 6 frequency bands and linearly anneal α from 0 to 6 over 80K iterations. We use the same MSE photometric loss as in NeRF [4] and weight the losses as $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rgb}} + \lambda \mathcal{L}_{\text{elastic-r}} + \mu \mathcal{L}_{\text{bg}}$ where we use $\lambda = \mu = 10^{-3}$ for all experiments except when mentioned. We train on 8 V100 GPUs for a week for full HD models, and



Figure 9: With $m = 4$ positional encoding frequencies, the model does not capture the smile. With $m = 8$, it fails to rotate the head. With coarse-to-fine (c2f) regularization, the model captures both.

for 16 hours for the half resolution models used for the comparisons in Table 1, Figure 11, and Figure 10. We provide more details in the Section A of the appendix.

5.2. Evaluation Dataset

In order to evaluate the quality of our reconstruction, we must be able to measure how faithfully we can recreate the scene from a viewpoint unseen during training. Since we are reconstructing non-rigidly deforming scenes, we cannot simply hold out views from an input capture, as the structure of the scene will be slightly different in every image. We therefore build a simple multi-view data capture rig for the sole purpose of evaluation. We found the multi-view dataset of Yoon et al. [16] not representative of many capture scenarios, as it contains too few viewpoints (12) and exaggerated frame-to-frame motions due to temporal subsampling.

Our rig (Figure 9) is a pole with two Pixel 3’s rigidly attached. We have two methods for data capture: (a) for selfies we use the front-facing camera and capture time-synchronized photos using the method of Ansari et al. [55], which achieves sub millisecond synchronization; or (b) we use the back-facing camera and record two videos which we manually synchronize based on the audio; we then subsample to 5 fps. We register the images using COLMAP [54] with rigid relative camera pose constraints. Sequences captured with (a) contain fewer frames (40~78) but the focus, exposure, and time are precisely synchronized. Sequences captured with (b) have denser samples in time (between 193 and 356 frames) but the synchronization is less precise and exposure and focus may vary between the cameras. We split each capture into a training set and a validation set. We alternate assigning the left view to the training set, and right to the validation, and vice versa. This avoids having regions of the scene that one

camera has not seen.

Quasi-static scenes: We capture 5 human subjects using method (a), that attempt to stay as still as possible during capture, and a mostly still dog using method (b).

Dynamic scenes: We capture 4 dynamic scenes containing deliberate motions of a human subject, a dog wagging its tail, and two moving objects using method (b).

5.3. Evaluation

Here we provide quantitative and qualitative evaluations of our model. However, to best appreciate the quality of the reconstructed nerfies, we encourage the reader to watch the supplementary video that contains many example results.

Quantitative Evaluation: We compare against NeRF and a NeRF + latent baseline, where NeRF is conditioned on a per-image learned latent code [42] to modulate density and color. We also compare with a variant of our system similar to the concurrent work of D-NeRF [40], which conditions a translational deformation field with a position encoded time variable $\gamma(\mathbf{t})$ instead of a latent code ($\gamma(\mathbf{t}) + trans$ in Table 1). We also compare with the high quality model of Neural Volumes (NV) [15] using a single view as input to the encoder, and Neural Scene Flow Fields (NSFF) [43]. We do not evaluate the method of Yoon et al. [16] due to the lack of available code (note that NSFF outperforms it). NSFF and the ($\gamma(\mathbf{t}) + trans$ baseline use temporal information while other baselines and our method do not. NSFF also uses auxilliary supervision such as estimated flow and relative depth maps; we do not. Note that the default hyper-parameters for NSFF [?] provided with the official code performs poorly on our datasets — we therefore contacted the authors to help us tune the hyper-parameters. Photometric differences between the two rig cameras may exist due to different exposure/white balance settings and camera response curves. We therefore swap the per-frame appearance code ψ_i for a per-camera $\psi_{\mathbf{L}}, \psi_{\mathbf{R}} \in \mathbb{R}^2$ instead for validation rig captures.

Table 1 reports LPIPS [56] and PSNR metrics for the unseen validation views. PSNR favors blurry images and is therefore not an ideal metric for dynamic scene reconstruction; we find that LPIPS is more representative of visual quality. See Figure 11 and Figure 10 for side-by-side images with associated PSNR/LPIPS metrics. Our method struggles with PSNR due to slight misalignments resulting from factors such as gauge ambiguity [57] while we outperform all baselines in terms of LPIPS for all sequences.

Ablation Study: We evaluate each of our contributions: SE(3) deformations, elastic regularization, background regularization, and coarse-to-fine optimization. We ablate them one at a time, and all at once (Ours (bare) in Table 1). As expected, a stronger elastic regularization ($\lambda = 0.01$) improves results for dynamic scenes compared to the baseline ($\lambda = 0.001$) while minimally impacting quasi-

static scenes. Removing the elastic loss hurts performance for quasi-static scenes while having minimal effect on the dynamic scene; this may be due to the larger influence of other losses in the presence of larger motion. Elastic regularization fixes distortion artifacts when the scene is under-constrained (e.g., Figure 4). Disabling coarse-to-fine regularization mildly drops performance for quasi-static scenes while causing a significant drop for dynamic scenes. This is expected since large motions are a main source of local minima (e.g., Figure 5). Our SE(3) deformations also quantitatively outperform translational deformations. Background regularization helps PSNR by reducing shifts in static regions and removing it performs worse. Finally, removing all of our contributions performs the worst in terms of LPIPS.

Qualitative Results: We show results for the captures used in the quantitative evaluation in Figure 11 and Figure 10. Our method can reconstruct fine details such as strands of hair (e.g., in CURLS of Table 1 and Figure 7), shirt wrinkles, and glasses (Figure 8). Our method works on general scenes beyond human subjects as shown in Figure 11 and Figure 10. In addition, we can create smooth animations by interpolating the deformation latent codes of any input state as shown in Figure 6.

Elastic Regularization: Figure 4 shows an example where the user only captured 20 images mostly from one side of their face, while their head tracked the camera. This results in ambiguous geometry. Elastic regularization helps in such under-constrained cases, reducing distortion significantly.

Depth Visualizations: We visualize the quality of our reconstruction using depth renders of the density field. Unlike NeRF[4] that visualizes the expected ray termination distance, we use the median depth termination distance, which we found to be less biased by residual density in free space (see Figure 7). We define it as the depth of the first sample with accumulated transmittance $T_i \geq 0.5$ (Eqn. 3 of NeRF [4]).

Limitations: Our method struggles with topological changes e.g., opening/closing of the mouth (see Figure 12) and may fail for certain frames in the presence of rapid motion (see supplementary). As mentioned in §3.4, our deformations are unconstrained so static regions may shift; this contributes to the disjunction between PSNR and LPIPS in Table 1. Future work may address this by modeling static regions separately as in [36], [43]. Finally, the quality of our method depends on camera registration, and when SfM fails so do we.

	Glasses (78)	Beanie (74)	Curls (57)	Kitchen (40)	Lamp (55)	Toby Sit (308)	Mean
NeRF [39]	18.1	16.8	14.4	19.1	17.4	22.8	18.1
NeRF + latent	19.5	19.5	17.3	20.1	18.9	19.4	19.1
Neural Volumes [31]	15.4	15.7	15.2	16.2	13.8	13.7	15.0
NSFF [†]	19.6	21.5	18.0	21.4	20.5	26.9	21.3
$\gamma(t) + \text{Trans}^\dagger$ [29]	22.2	20.8	20.7	22.5	21.9	25.3	22.2
Ours ($\lambda = 0.01$)	23.4	22.2	24.6	23.9	23.6	22.9	23.4
Ours ($\lambda = 0.001$)	24.2	23.2	24.9	23.5	23.7	22.8	23.7
No elastic	23.1	24.2	24.1	22.9	23.7	23.0	23.5
No coarse-to-fine	23.8	21.9	24.5	24.0	22.8	22.7	23.3
No SE3	23.5	21.9	24.5	23.7	22.7	22.9	23.2
Ours (base)	24.0	20.9	23.5	22.4	22.1	22.7	22.6
No BG Loss	22.3	21.5	20.1	22.5	20.3	22.3	21.5

(a) Quasi-Static, PSNR $\uparrow$

	Glasses (78)	Beanie (74)	Curls (57)	Kitchen (40)	Lamp (55)	Toby Sit (308)	Mean
NeRF [39]	.474	.583	.616	.434	.444	.463	.502
NeRF + latent	.463	.535	.539	.403	.386	.385	.452
Neural Volumes [31]	.616	.595	.588	.569	.533	.473	.562
NSFF [†]	.407	.402	.432	.317	.239	.208	.334
$\gamma(t) + \text{Trans}^\dagger$ [29]	.354	.471	.426	.344	.283	.420	.383
Ours ($\lambda = 0.01$)	.305	.391	.319	.280	.232	.159	.281
Ours ($\lambda = 0.001$)	.307	.391	.312	.279	.230	.174	.282
No elastic	.317	.382	.322	.290	.230	.257	.300
No coarse-to-fine	.312	.408	.321	.277	.242	.244	.301
No SE3	.314	.401	.317	.282	.235	.206	.293
Ours (base)	.319	.456	.345	.323	.254	.184	.314
No BG Loss	.317	.395	.371	.290	.260	.145	.296

(b) Quasi-Static, LPIPS $\downarrow$

	Drinking (193)	Tail (238)	Badminton (356)	Broom (197)	Mean
NeRF [39]	18.6	23.0	18.8	21.0	20.3
NeRF + latent	21.9	24.9	20.0	21.9	22.2
Neural Volumes [31]	16.2	18.5	13.1	16.1	16.0
NSFF [†]	27.7	30.6	21.7	28.2	27.1
$\gamma(t) + \text{Trans}^\dagger$ [29]	23.7	27.2	22.9	23.4	24.3
Ours ($\lambda = 0.01$)	22.4	23.9	22.4	21.5	22.5
Ours ($\lambda = 0.001$)	21.8	23.6	22.1	21.0	22.1
No elastic	22.2	23.7	22.0	20.9	22.2
No coarse-to-fine	22.3	24.3	21.8	21.9	22.6
No SE3	22.4	23.5	21.2	20.9	22.0
Ours (base)	22.6	24.3	21.1	22.1	22.5
No BG Loss	22.3	23.5	20.4	20.9	21.8

(c) Dynamic, PSNR $\uparrow$

	Drinking (193)	Tail (238)	Badminton (356)	Broom (197)	Mean
NeRF [39]	.397	.571	.392	.667	.506
NeRF + latent	.233	.404	.308	.576	.380
Neural Volumes [31]	.198	.559	.516	.544	.454
NSFF [†]	.0803	.245	.205	.202	.183
$\gamma(t) + \text{Trans}^\dagger$ [29]	.151	.391	.221	.627	.347
Ours ($\lambda = 0.01$)	.0872	.161	.130	.245	.156
Ours ($\lambda = 0.001$)	.0962	.175	.132	.270	.168
No elastic	.0863	.174	.132	.287	.170
No coarse-to-fine	.0960	.257	.151	.406	.228
No SE3	.0867	.191	.156	.276	.177
Ours (base)	.127	.246	.173	.503	.275
No BG Loss	.0856	.210	.161	.330	.196

(d) Dynamic, LPIPS $\downarrow$

Table 1: Quantitative evaluation on validation captures against baselines and ablations of our system, we color code each row as **best**, **second best**, and **third best**. Please see §5.3 for more details.



Figure 10: With $m = 4$ positional encoding frequencies, the model does not capture the smile. With $m = 8$, it fails to rotate the head. With coarse-to-fine (c2f) regularization, the model captures both.

6. Conclusion

Deformable Neural Radiance Fields extend NeRF by modeling non-rigidly deforming scenes. We show that our as-rigid-as-possible deformation prior, and coarse-to-fine deformation regularization are the key to obtaining high-quality results. We showcase the application of casual selfie captures (*nerfies*), and enable high-fidelity reconstructions of human subjects using a cellphone capture. Future work may tackle larger/faster motion, topological variations, and enhance the speed of training/inference.

Acknowledgments

We thank Peter Hedman and Daniel Duckworth for providing feedback in early drafts, and all our capture subjects for their patience, including Toby who was a good boy.



Figure 11: With $m = 4$ positional encoding frequencies, the model does not capture the smile. With $m = 8$, it fails to rotate the head. With coarse-to-fine (c2f) regularization, the model captures both.

A. Details of SE(3) Field Formulation

As mentioned in the main text, we encode a rigid transform as a screw axis [45] $\mathcal{S} = (\mathbf{r}; \mathbf{v}) \in \mathbb{R}^6$ where

$$e^{\mathbf{r}} \equiv e^{[\mathbf{r}]_{\times}} = \mathbf{I} + \frac{\sin \theta}{\theta} [\mathbf{r}]_{\times} + \frac{1 - \cos \theta}{\theta^2} [\mathbf{r}]_{\times}^2$$

$[\mathbf{x}]_{\times}$ is a skew-symmetric matrix also known as the crossproduct matrix of a vector x since given two 3-vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{a}_{\times} \mathbf{b}$ gives the cross product $\mathbf{a} \times \mathbf{b}$.

$$[\mathbf{x}]_{\times} = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}$$

The translation encoded by the screw motion $\mathcal{S}$ can be recovered as $\mathbf{p} = \mathbf{G}\mathbf{v}$ where

$$\mathbf{G} = \mathbf{I} + \frac{1 - \cos \theta}{\theta^2} [\mathbf{r}]_{\times} + \frac{\theta - \sin \theta}{\theta^3} [\mathbf{r}]_{\times}^2$$



Figure 12: Our method fails in the presence of topological changes. Although the rendered color views look good from some viewpoints, the recovered geometry is incorrect.

The exponential of $\mathcal{S}$ can also be expressed in homogeneous matrix form $\mathbf{e}^{\mathcal{S}} \in SE(3)$:

$$e^{\mathcal{S}} = \begin{pmatrix} e^{\mathbf{r}} & \mathbf{p} \\ 0 & 1 \end{pmatrix}$$

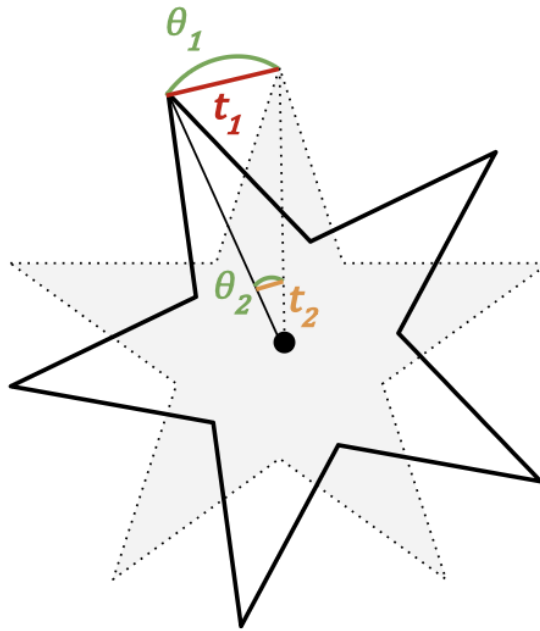


Figure 13: Here we illustrate why a rigid transformation field works better than a translation field with a simple toy example where a star is rotated counter-clockwise around its center. A translation field requires different parameters for every point to encode the rotation (e.g., $\| \mathbf{t}_1 \| \gg \| \mathbf{t}_2 \|$) whereas a rotation field only needs a single parameter to encode the rotation (e.g., $.1 = .2$). More details in §A.

The deformed point is then given by $x' = e^S x$.

Why does an SE(3) field work better? Consider the example in Figure 13 where a star has been rotated counter-clockwise along its center. Now consider what transformation would be required at every point on the star to encode this rotation. With a translation field, points towards the center (e.g., t2) need translations of small magnitude while points towards the outside (e.g., t1) need translations of larger magnitude. Every point on the star requires a different parameter to encode a simple rotation. On the otherhand, with a rotation, every point on the star can be parameterized by a single angle which is the angle of rotation $\theta = .1 = .2$. This makes optimization much easier since the deformation field MLP only needs to predict a single parameter across space. We illustrate this further in §I.

B. Details of Coarse-to-Fine Optimization

Window Function: Our coarse-to-fine deformation regularization is implemented by windowing the frequency bands of the positional encoding. Eqn. 8 of the main paper defines this windowing function as a weight applied to each frequency band. We visualize our windowing function for different values of α in Figure 15.

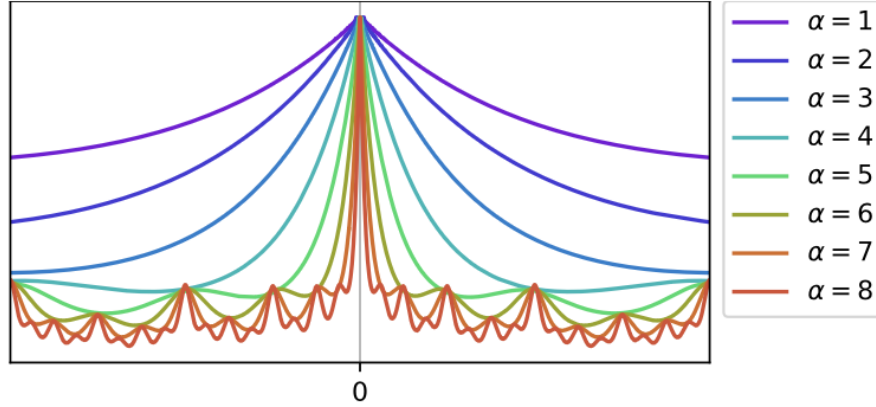


Figure 14: Visualizations of the neural-tangent kernel (NTK) [52] of our annealed positional encoding for different values of α . Our coarse-to-fine optimization scheme works by easing in the influence of each positional encoding frequency through a parameter α . This has the effect of shrinking the bandwidth of the NTK corresponding to the deformation MLP as α is increased, thereby allowing higher frequency deformations.

NTK: We also show a visualization of the neural tangent kernel (NTK) induced by our annealed positional encoding in Figure 14. This figure shows the normal-

ized NTK for an 8 layer MLP of width 256. Note how the bandwidth of the interpolation kernel gets narrower as the value of a increases.

C. Details of Elastic Regularization

Motivation for elastic energy formulation. Elastic energies are often implemented as the deviation of the Jacobian $\mathbf{J}$ from the closest rotation $\mathbf{R}$: $\|\mathbf{J} - \mathbf{R}\|_F$ [p45]. Let $\mathbf{J} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ be the SVD of $\mathbf{J}$, then $\mathbf{R} = \mathbf{U}\mathbf{M}\mathbf{V}^T$ where

$$\begin{aligned}\|\mathbf{J} - \mathbf{R}\|_F &= \|\mathbf{U}\mathbf{D}\mathbf{V}^T - \mathbf{U}\mathbf{M}\mathbf{V}^T\|_F \\ &= \|\mathbf{U}(\mathbf{D} - \mathbf{M})\mathbf{V}^T\|_F \\ &= \sqrt{\text{tr}(\mathbf{U}(\mathbf{D} - \mathbf{M})\mathbf{V}^T\mathbf{V}(\mathbf{D} - \mathbf{M})\mathbf{U}^T)} \\ &= \sqrt{\text{tr}(\mathbf{U}(\mathbf{D} - \mathbf{M})^2\mathbf{U}^T)} \\ &= \sqrt{\text{tr}((\mathbf{D} - \mathbf{M})^2)} \\ &= \sqrt{\sum_j (\sigma_j - m_j)^2}\end{aligned}$$

where m_j is the j th diagonal of $\mathbf{M}$ and σ_j is the j th singular value of $\mathbf{J}$. This is equivalent to penalizing the deviation of the singular values of $\mathbf{J}$ from 1. The $\mathbf{M}$ matrix factors in reflections as negative singular values rather a reflection in $\mathbf{U}$ or $\mathbf{V}$. Because this formulation penalizes expansions more than contractions of the same factor, we penalize the log of the singular values directly.

D. Additional Illustrations

Unintentional Movement: In Figure 16 we show an example of how a person can move even when trying to sit still. We visualize the degree of movement by showing the difference in predicted depth as well as by showing a direct plot of the magnitude of the deformation field at the predicted depth point.

Domain Agnostic: In Figure 17, Figure 11, and Figure 10. we show that our method works agnostic of the type of subject.

E. Additional Implementation Details

Architecture Details: We provide architecture details of the deformation field network and canonical NeRF networks in Figure 18 and Figure 19 respectively. **Training:** We train our network using the Adam optimizer [58] with a learning rate exponentially decayed by a factor 0.1 until the maximum number of iterations

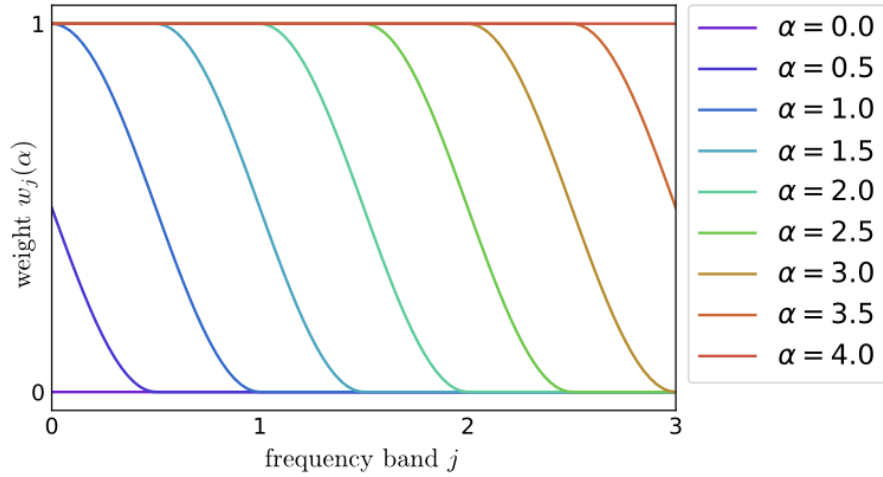


Figure 15: A visualization of the window function $w_j(\alpha)$ for the annealed positional encoding. We show an example with a maximum number of frequency bands of $\mathbf{m} = 4$ where $\mathbf{j} \in \{0, \dots, \mathbf{m} - 1\}$. $\alpha = 0$ sets the weight of all frequency bands to zero leaving only the identity mapping, while an $\alpha = 4$ sets the weight of all frequency bands to one. Increasing the value of α is equivalent to sliding the window to the right across the frequency bands.

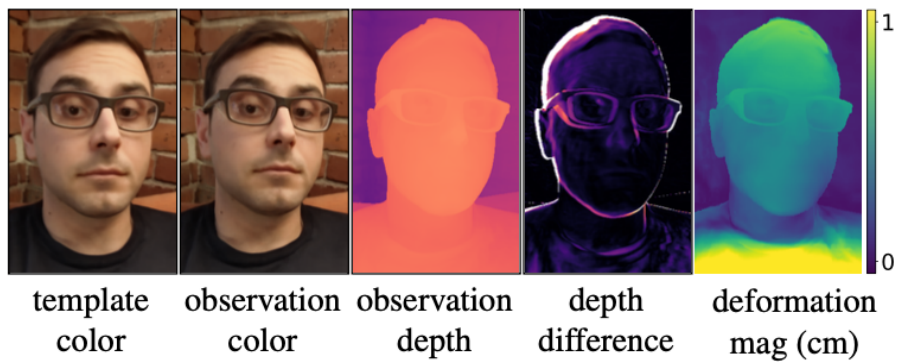


Figure 16: Users move even when trying not to. Here we visualize the depth difference and deformation magnitude between the template and an observation.

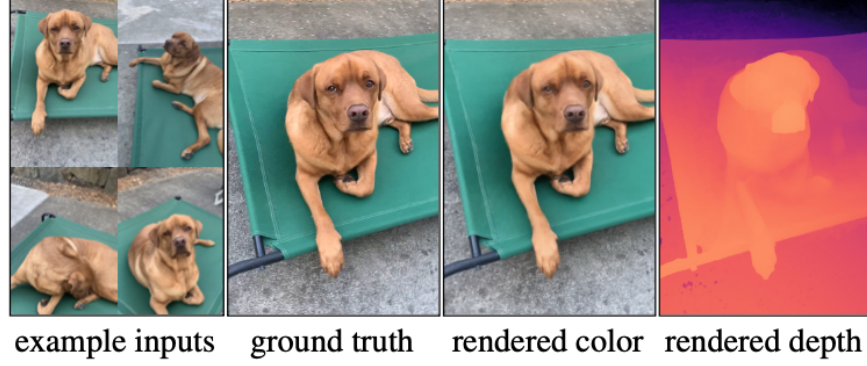


Figure 17: Not relying on domain specific priors enables our method to reconstruct any deformable object. In this case, the dog fails to stay still, yet we recover an accurate model.

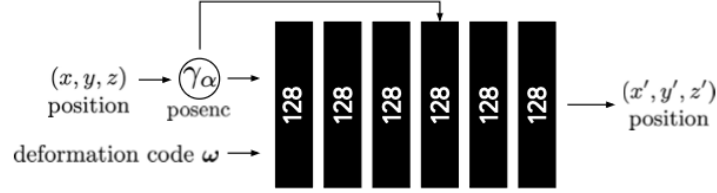


Figure 18: A diagram of our deformation network. The deformation network takes a position encoded position $a(x)$ using our coarse-to-fine annealing parameterized by a , along with a deformation code ω and outputs a deformed position $0 \times$. The architecture is identical for all of our experiments.

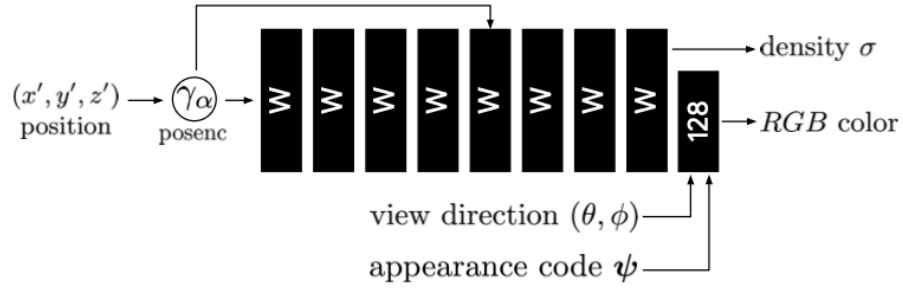


Figure 19: A diagram of the canonical NeRF network. Our network is identical to the original NeRF MLP, except we provide an appearance latent code ψ along with the view direction to allow modulating the appearance as in the NeRF-A model of [36]. The width W of the network is defined according to Table 3.

is reached. The exact hyper-parameters for each configuration are provided in Table 3.

Background Regularization: Since the total number of background points varies per scene, we sample 16384 points for each iteration when computing the background regularization loss in order to avoid memory issues. We additionally jitter each input point using Gaussian noise $e \sim N(0, 0.001)$ and use a robust Geman-McClure loss function [49] with $a = -2$ and $c = 0.001$ implemented as per Barron [50].

Implementation: We extend the JAX [59] implementation of NeRF [60] for our method.

F. Experiment Details

F.1. Dataset Processing

Blurry Frame Filtering: For video captures, we filter blurry frames using the variance of the Laplacian [53]. To compute the blur score for an image, we apply the Laplace operator with kernel size 3 and compute the variance of the resulting image. We then filter the images based on this score to leave around 600 frames for each capture.

Camera Registration: For camera registration, we first compute a foreground mask using a semantic segmentation network such as DeepLabV3 [61]. We then use COLMAP [54] to compute the camera registration while using the mask to ignore foreground pixels when computing features. We found that this step can improve the quality of the camera registration in the presence of a moving foreground. We skip this step for captures for which we cannot obtain a segmentation mask such as for BADMINTON and BROOM.

Facial Landmarks: Although not necessary for our method, we use facial landmarks for self and full body captures to estimate a canonical frame of reference. Using this canonical frame of reference, we automatically generate visually appealing novel view trajectories of our reconstructed nerfes, like figure-eight camera paths in front of the user. We compute the 2D facial landmarks using MediaPipe’s face mesh [62], and triangulate them in 3D using the Structure-from-Motion camera poses. We then set our canonicalized coordinate frame that is centered at the facemesh, with a standard orientation (+y up, +x right, -z into the face), and with approximately metric units, by setting the scale so that the distance between the eyes matches the average interpupillary distance of 6 cm. Note that the 3D triangulation of facial landmarks is only correct if the subject is static, which is not guaranteed in our method, but in practice we observed that the triangulation result is sufficiently good to define the coordinate frame even when the subject rotates the head side-to-side. For the animal captures, we manually generate virtual camera paths.

F.2. Baselines

Comparison to Neural Volumes: Neural Volumes [15] reconstructs a deformable model of a subject captured by dozens of time-synchronized cameras. To apply it to our setting, where only one camera sees the subject at each time instance, we modify the encoder to network to take a single input image instead of three, as in the original method. We disable the background estimation branch and learn instead the complete scene centered around the face and scaled to a unit cube. For each frame, we render the volume from the viewpoint of the second camera of the validation rig and compute image comparison metrics. We provide quantitative comparisons in Table 1 in the main paper, and qualitative comparisons in Figure 11 and Figure 10.

We use a 128^3 voxel grid, a 32^3 warp field and train the network for 100k iterations for each of the five sequences. We evaluate all results using the same camera parameters and spatial resolution. We show some renderings when interpolating the camera position between training and validation views in the supplementary video.

	Glasses (78)	Beanie (74)	Curls (57)	Kitchen (40)	Lamp (55)	Toby Sit (308)	Mean
NeRF [39]	.619	.580	.504	.695	.656	.793	.641
NeRF + latent	.695	.687	.619	.746	.735	.798	.713
Neural Volumes [31]	.503	.562	.538	.609	.563	.583	.560
NSFF [†]	.678	.760	.621	.780	.807	.913	.760
$\gamma(t) + \text{Trans}^\dagger$ [29]	.781	.737	.732	.823	.836	.870	.796
Ours ($\lambda = 0.01$)	.826	.786	.842	.878	.888	.806	.838
Ours ($\lambda = 0.001$)	.840	.805	.846	.863	.886	.805	.841
No elastic	.809	.824	.830	.851	.889	.821	.837
No coarse-to-fine	.828	.771	.841	.877	.867	.807	.832
No SE3	.823	.782	.839	.870	.872	.810	.833
Ours (base)	.828	.737	.818	.829	.851	.792	.809
No BG Loss	.779	.758	.696	.844	.806	.775	.776

(a) Quasi-Static, SSIM $\uparrow$

	Glasses (78)	Beanie (74)	Curls (57)	Kitchen (40)	Lamp (55)	Toby Sit (308)	Mean
NeRF [39]	.474	.583	.616	.434	.444	.463	.502
NeRF + latent	.463	.535	.539	.403	.386	.385	.452
Neural Volumes [31]	.616	.595	.588	.569	.533	.473	.562
NSFF [†]	.407	.402	.432	.317	.239	.208	.334
$\gamma(t) + \text{Trans}^\dagger$ [29]	.354	.471	.426	.344	.283	.420	.383
Ours ($\lambda = 0.01$)	.305	.391	.319	.280	.232	.159	.281
Ours ($\lambda = 0.001$)	.307	.391	.312	.279	.230	.174	.282
No elastic	.317	.382	.322	.290	.230	.257	.300
No coarse-to-fine	.312	.408	.321	.277	.242	.244	.301
No SE3	.314	.401	.317	.282	.235	.206	.293
Ours (base)	.319	.456	.345	.323	.254	.184	.314
No BG Loss	.317	.395	.371	.290	.260	.145	.296

(b) Quasi-Static, LPIPS $\downarrow$

	Drinking (193)	Tail (238)	Badminton (356)	Broom (197)	Mean
NeRF [39]	.619	.676	.771	.643	.677
NeRF + latent	.855	.800	.850	.688	.798
Neural Volumes [31]	.771	.503	.219	.515	.502
NSFF [†]	.964	.917	.840	.893	.904
$\gamma(t) + \text{Trans}^\dagger$ [29]	.910	.882	.927	.750	.867
Ours ($\lambda = 0.01$)	.894	.754	.926	.674	.812
Ours ($\lambda = 0.001$)	.881	.731	.922	.605	.785
No elastic	.890	.735	.919	.593	.784
No coarse-to-fine	.892	.763	.912	.695	.815
No SE3	.895	.715	.899	.599	.777
Ours (base)	.894	.768	.894	.695	.813
No BG Loss	.893	.719	.875	.593	.770

(c) Dynamic, SSIM $\uparrow$

	Drinking (193)	Tail (238)	Badminton (356)	Broom (197)	Mean
NeRF [39]	.397	.571	.392	.667	.506
NeRF + latent	.233	.404	.308	.576	.380
Neural Volumes [31]	.198	.559	.516	.544	.454
NSFF [†]	.0803	.245	.205	.202	.183
$\gamma(t) + \text{Trans}^\dagger$ [29]	.151	.391	.221	.627	.347
Ours ($\lambda = 0.01$)	.0872	.161	.130	.245	.156
Ours ($\lambda = 0.001$)	.0962	.175	.132	.270	.168
No elastic	.0863	.174	.132	.287	.170
No coarse-to-fine	.0960	.257	.151	.406	.228
No SE3	.0867	.191	.156	.276	.177
Ours (base)	.127	.297	.173	.503	.275
No BG Loss	.0856	.210	.161	.330	.196

(d) Dynamic, LPIPS $\downarrow$

Table 2: SSIM $\uparrow$ and LPIPS $\downarrow$ metrics on validation captures against baselines and ablations of our system, we color code each row as **best**, **second best**, and **third best**. Please see the main text for PSNR.

Comparison to NSFF: Concurrently to our work, Neural-Scene Flow Fields (NSFF) [43] proposes to model dynamic scenes by directly conditioning the NeRF with a position-encoded time variable $.(t)$, modulating color, density, and a scene-flow prediction. Differences from our method are: (a) NSFF directly modulates the density of the NeRF by conditioning it with $.(t)$ while our method uses a deformation field; (b) NSFF uses a position-encoded time variable $.(t)$ to condition each observation whereas our method uses a per-example latent code [42]; (c) NSFF uses depth from MIDAS [63] and optical flow from RAFT [64] as supervision whereas our method only uses a photometric loss.

We quantitatively compare with NSFF in Table 1 of the paper and in Table 2, and show corresponding qualitative results in Figure 11 and Figure 10. We use the official code released by the NSFF authors. The authors provided us with hyper-parameters tuned for our datasets.

Additional Metrics: MS-SSIM metrics are in Table 2.

G. Additional Results

We show qualitative results from each of the sequences presented in our quantitative evaluation (Table 1 of paper, Table 2) for quasi-static scenes (Figure 11) and dynamic scenes (Figure 10).

Config	ResolutionSteps		Learning Rate	Batch Size	# Sam- ples Fine	# Sam- ples Fine	Width W
FULL	1080p	1M	7.5e-4	3072	256	256	256
HALF	540p	100K	1e-3	8096	128	128	128

Table 3: Here we provide the hyper-parameters used for each configuration. FULL is the full resolution configuration used in our qualitative results. HALF is half the resolution of FULL and is used for our quantitative evaluation and ablation studies.

H. Limitations

Topological Variation: Our method struggles when the scene has motion which varies the topology of the scene. For example, when a person opens their mouth (as in Figure 11 of the main text), the effective topology of their head

changes. This is problematic for our method since we use a continuous deformation field parameterized by an MLP. In order to understand this, consider the mouth example and suppose that the template contains the person with their

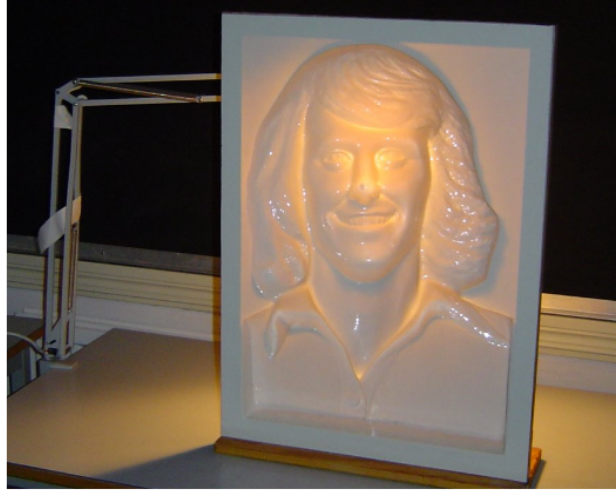


Figure 20: With $\mathbf{m} = 4$ positional encoding frequencies, the model does not capture the smile. With $\mathbf{m} = 8$, it fails to rotate the head. With coarse-to-fine (c2f) regularization, the model captures both.



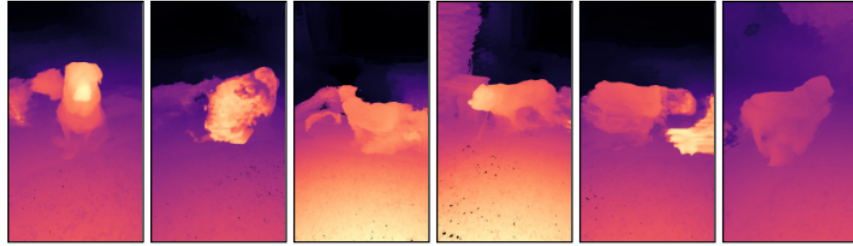
Figure 21: If the user's gaze consistently follows the camera, the reconstructed nerf represents the user's gaze as geometry, akin to the Hollow-Face illusion [65]. This is apparent in the depth map and makes the reconstructed model appear as if they are looking at the camera even when the geometry is fixed.



(a) input rgb



(b) rendered rgb



(c) rendered depth

Figure 22: This example shows Toby the dog moving around freely, showcasing two limitations of our method. (1) Rapid motion: Because Toby moves quite fast, the camera only sees him in certain poses for a short amount of time, resulting in a sparse set of observations for certain poses. This can make those poses under-constrained. (2) Orientation flips: Toby wanders back and forth, showing different sides of his body. Depending on which orientation Toby is modeled as in the template, it is difficult for the deformation field to predict a flipped orientation.

mouth open. Suppose that x_U and x_L are two adjacent points near the seam of the lips, and the x_U is on the upper lip and x_L is on the lower lip. It is then evident that a sharp discontinuity in the deformation is required to map both points to their appropriate positions on the template. Such a discontinuity is difficult for our continuous MLP to predict. We find that instead the optimization will often yield an incorrect but valid solution e.g., it will explain a closing mouth by protruding the lip and pulling it down as in Figure 11 of the paper.

Rapid Motion: NeRF relies on seeing multiple observations to constrain where density lies in the volumes. In the presence of rapid motion, such as in Figure 22, certain states of the scene may only be visible for a short period of time making it harder to reconstruct.

Orientation flips: Optimizations solving for any parameterization of rotations are known to be non-convex due to

both Gauge ambiguity and the inherent twistedness of the space of $SO(3)$ [66]. As a simple example to illustrate this, imagine trying to align two coins in 3D. If the coin is initialized in a flipped orientation where heads faces the tail side of the other coin, then the effort of the two coins must get worse before getting better when rotating towards the global minimum.

We encounter the same issue when optimizing for our deformation fields. If the template of a scene is in a certain orientation, but the deformation field for an observation is initialized in the wrong orientation the method will get stuck in a local minima and result in sub-optimal alignment. We show an example of this in Figure 22 where frames with Toby’s left side visible are reconstructed better than when Toby’s right side is visible.

Hollow Face Illusion: The hollow-face illusion is an optical illusion where a concave (pushed in) imprint of an object appears to be convex (pushed out) instead. A feature of this illusion is that the convex illusion appears to follow the viewer’s eye. This illusion has been purposefully used in the Disneyland haunted mansion to create face busts which appear to follow you and in the popular T-Rex illusion [67]. We show an example of this illusion in Figure 20.

We observe that the ambiguity which causes this illusion can also be a failure mode for our method. In Figure 21, we show an example where a user fixes their gaze in the direction of the camera while capturing themselves. Instead of modeling the eye motion as a deformation, our method models the eyes concavities as can be seen in the geometry.

I. 2D Deformation Experiment

Here we analyze the behavior of a deformation field in a 2D toy setting. In this 2D setting, a “scene” is comprised of a single image which is randomly translated, rotated, and nonlinearly distorted near the center. We show the full dataset in Figure 23. Akin to our deformable NeRF setting, the task is to



Figure 23: Our 2D toy dataset comprised of an image with a random translation, a random rotation, and a random non- linear distortion near the center. Astronaut photo by Robert Gibson (1984, Public Domain).

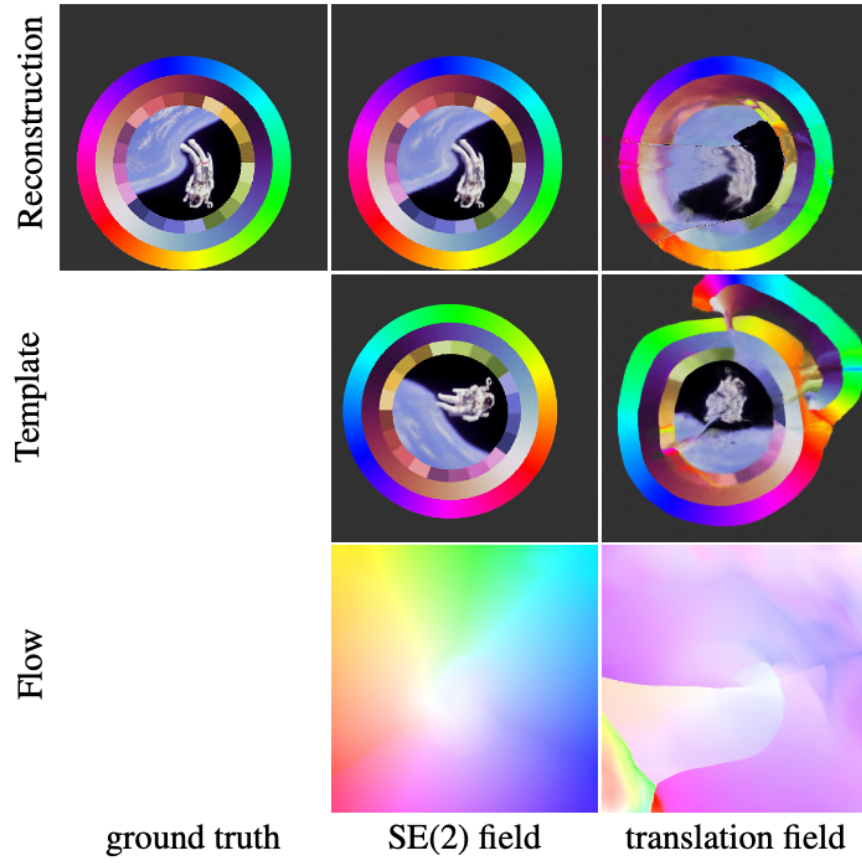


Figure 24: A comparison of our SE(2) feld and a translation feld. The translation parameterization has difficulty rotating groups of distant pixels whereas the rigid transformation successfully finds the correct orientation.

reconstruct each image by using a 2D deformation field which references a single template. The template is an MLP $F : (x, y) \rightarrow (r, g, b)$ which maps normalized image coordinates $x, y \in [-1, 1]$ to color values. The deformation field (i.e., 2D flow) is represented as an MLP $T : (x, y) \rightarrow (t_x, t_y)$ for a translation field or $T : (x, y) \rightarrow (\theta, p_x, p_y, t_x, t_y)$ for a rigid SE(2) transformation field. These are 2D analogs to the 3D translation field and SE(3) field described in §3.2.

Deformation Formulation: Figure 24 shows how an SE(2) rigid transformation field outperforms a translation field. An SE(2) field is able to faithfully reconstruct each image with a reasonable template and smooth deformation field. On the other hand, a translation field is not able to recover a reasonable template, and as a result the reconstruction has many artifacts and the flow field is messy.

Positional Encoding Frequencies: We show how changing the number of frequency bands changes the convergence behavior in Figure 25. With a small number of frequencies ($\mathbf{m} = 1$) we are able to converge to the correct orientation in the template, but cannot fully model the non-linear ‘swirl’ towards the center of the image. If we increase the number of frequencies ($\mathbf{m} = 2 \dots 6$) then while we can reconstruct the high swirl better, we start introducing artifacts due to early overfitting of the template and deformation field. With our coarse-to-fine approach we are able to both get the correct orientation without artifacts and also model the swirl.

References

- [1] A. Collet *et al.*, “High-quality streamable free-viewpoint video,” *ACM ToG*, 2015.
- [2] M. Dou *et al.*, “Fusion4D: Real-time performance capture of challenging scenes,” *ACM ToG*, 2016.
- [3] K. Guo *et al.*, “The relightables: Volumetric performance capture of humans with realistic relighting,” *ACM ToG*, 2019.
- [4] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: Representing scenes as neural radiance fields for view synthesis,” in *ECCV*, 2020.
- [5] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly, “Projective dynamics: Fusing constraint projections for fast simulation,” *ACM TOG*, 2014.
- [6] I. Chao, U. Pinkall, P. Sanan, and P. Schröder, “A simple geometric model for elastic deformations,” *ACM Trans. Graph.*, 2010.
- [7] O. Sorkine and M. Alexa, “As-rigid-as-possible surface modeling,” in *EUROGRAPHICS*, 2007.
- [8] R. W. Sumner, J. Schmid, and M. Pauly, “Embedded deformation for shape manipulation,” *ACM TOG*, 2007.
- [9] C. Bregler, A. Hertzmann, and H. Biermann, “Recovering non-rigid 3d shape from image streams,” in *CVPR*, 2000.

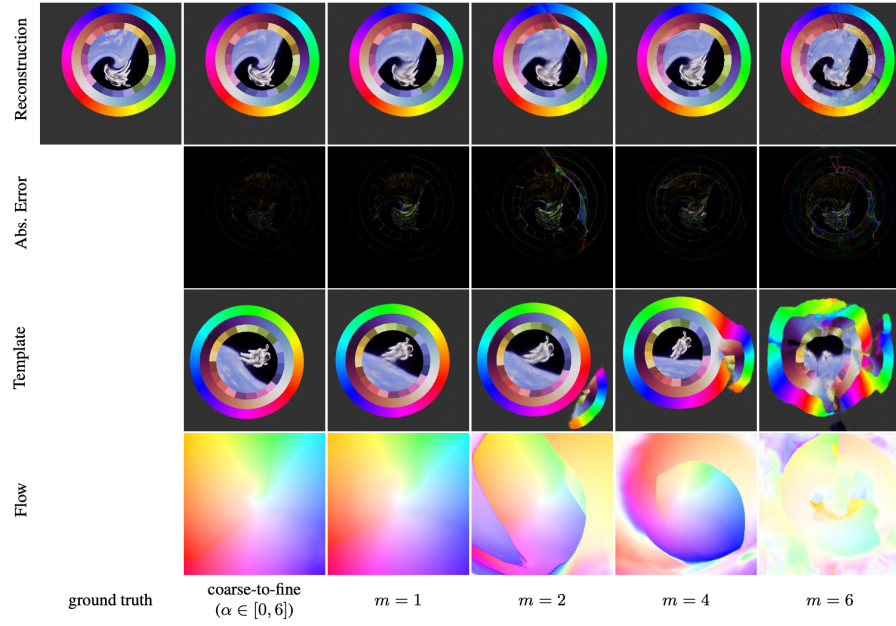


Figure 25: We show how the optimization changes depending on the number of frequency bands in the positional encoding of the deformation field. With 1 frequency, the model find the correct orientation of all images but is unable to model the high frequency distortion near the center. If we increase the frequencies then the templates overfits early and gets stuck in bad local minima. With our coarse-to-fine technique we are less prone to local minima while also modeling high frequency details.

- [10] L. Torresani, A. Hertzmann, and C. Bregler, “Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors,” *TPAMI*, 2008.
- [11] H. Li *et al.*, “Temporally coherent completion of dynamic shapes,” *ACM TOG*, 2012.
- [12] R. A. Newcombe, D. Fox, and S. M. Seitz, “DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time,” in *CVPR*, 2015.
- [13] A. Božič, P. Palafox, M. Zollhöfer, A. Dai, J. Thies, and M. Nießner, “Neural non-rigid tracking,” *arXiv preprint arXiv:2006.13240*, 2020.
- [14] T. Schmidt, R. Newcombe, and D. Fox, “DART: Dense articulated real-time tracking with consumer depth cameras,” *Autonomous Robots*, 2015.
- [15] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, “Neural volumes: Learning dynamic renderable volumes from images,” *ACM ToG*, 2019.
- [16] J. S. Yoon, K. Kim, O. Gallo, H. S. Park, and J. Kautz, “Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera,” in *CVPR*, 2020.
- [17] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, “Occupancy flow: 4d reconstruction by learning particle dynamics,” in *ICCV*, 2019.
- [18] C. Jiang, J. Huang, A. Tagliasacchi, L. Guibas, *et al.*, “Shapeflow: Learnable deformations among 3d shapes,” *arXiv preprint arXiv:2006.07982*, 2020.
- [19] V. Blanz and T. Vetter, “A morphable model for the synthesis of 3D faces,” in *SIGGRAPH*, 1999.
- [20] J. Booth, A. Roussos, A. Ponniah, D. Dunaway, and S. Zafeiriou, “Large scale 3D morphable models,” *IJCV*, 2018.
- [21] S. Bouaziz, Y. Wang, and M. Pauly, “Online modeling for realtime facial animation,” *ACM TOG*, 2013.
- [22] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, “SMPL: A skinned multi-person linear model,” *ACM Trans. Graph.*, 2015.
- [23] H. Xu, E. G. Bazavan, A. Zafir, W. T. Freeman, R. Sukthankar, and C. Sminchisescu, “GHUM & GHUML: Generative 3D human shape and articulated pose models,” in *CVPR*, 2020.
- [24] T. J. Cashman and A. W. Fitzgibbon, “What shape are dolphins? Building 3D morphable models from 2D images,” *TPAMI*, 2012.
- [25] S. Zuffi, A. Kanazawa, D. W. Jacobs, and M. J. Black, “3D menagerie: Modeling the 3D shape and pose of animals,” in *CVPR*, 2017.
- [26] M. Zollhöfer *et al.*, “State of the art on monocular 3D face reconstruction, tracking, and applications,” *Computer Graphics Forum*, 2018.
- [27] O. Fried *et al.*, “Text-based editing of talking-head video,” *ACM TOG*, 2019.
- [28] H. Kim *et al.*, “Deep video portraits,” *ACM ToG*, 2018.

- [29] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, “Occupancy networks: Learning 3D reconstruction in function space,” in *CVPR*, 2019.
- [30] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “DeepSDF: Learning continuous signed distance functions for shape representation,” in *CVPR*, 2019.
- [31] Z. Chen and H. Zhang, “Learning implicit fields for generative shape modeling,” in *CVPR*, 2019.
- [32] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, “Scene representation networks: Continuous 3D-structure-aware neural scene representations,” in *NeurIPS*, 2019.
- [33] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, “Implicit neural representations with periodic activation functions,” in *NeurIPS*, 2020.
- [34] M. Tancik *et al.*, “Fourier features let networks learn high frequency functions in low dimensional domains,” in *NeurIPS*, 2020.
- [35] S. Bi *et al.*, “Neural reflectance fields for appearance acquisition,” *arXiv preprint arXiv:2008.03824*, 2020.
- [36] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, “NeRF in the wild: Neural radiance fields for unconstrained photo collections,” in *CVPR*, 2021.
- [37] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt, “Neural sparse voxel fields,” in *NeurIPS*, 2020.
- [38] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, “NeRF++: Analyzing and improving neural radiance fields,” *arXiv preprint arXiv:2010.07492*, 2020.
- [39] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, “GRAF: Generative radiance fields for 3D-aware image synthesis,” in *NeurIPS*, 2020.
- [40] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, “D-NeRF: Neural radiance fields for dynamic scenes,” *arXiv preprint arXiv:2011.13961*, 2020.
- [41] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt, “Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video,” 2021.
- [42] P. Bojanowski, A. Joulin, D. Lopez-Pas, and A. Szlam, “Optimizing the latent space of generative networks,” in *ICML*, 2018.
- [43] Z. Li, S. Niklaus, N. Snavely, and O. Wang, “Neural scene flow fields for space-time view synthesis of dynamic scenes,” *arXiv preprint arXiv:2011.13084*, 2020.
- [44] W. Xian, J.-B. Huang, J. Kopf, and C. Kim, “Space-time neural irradiance fields for free-viewpoint video,” *arXiv preprint arXiv:2011.12950*, 2020.
- [45] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

- [46] O. Rodrigues, “De l’attraction des sphéroïdes,” *Correspondence Sur l’École Impériale Polytechnique*, pp. 361–385, 1816.
- [47] M. Zollhöfer *et al.*, “Real-time non-rigid reconstruction using an RGB-D camera,” *ACM Trans. Graph.*, 2014.
- [48] E. Sifakis and J. Barbic, “FEM simulation of 3D deformable solids: A practitioner’s guide to theory, discretization and model reduction,” in *ACM SIGGRAPH 2012 courses*, 2012.
- [49] S. Geman and D. E. McClure, “Bayesian image analysis: An application to single photon emission tomography,” *Proceedings of the American Statistical Association*, 1985.
- [50] J. T. Barron, “A general and adaptive robust loss function,” in *CVPR*, 2019.
- [51] B. LUCAS, “An iterative image registration technique with an application to stereo vision,” in *Proceedings of imaging understanding workshop, 1981*, 1981.
- [52] A. Jacot, F. Gabriel, and C. Hongler, “Neural tangent kernel: Convergence and generalization in neural networks,” in *NeurIPS*, 2018.
- [53] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martinez, and J. Fernández-Valdivia, “Diatom autofocusing in brightfield microscopy: A comparative study,” in *ICPR*, IEEE, 2000, pp. 314–317.
- [54] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *CVPR*, 2016.
- [55] S. Ansari, N. Wadhwa, R. Garg, and J. Chen, “Wireless software synchronization of multiple distributed cameras,” in *ICCP*, 2019.
- [56] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *CVPR*, 2018.
- [57] P. F. McLauchlan, “Gauge invariance in projective 3d reconstruction,” in *Proceedings IEEE workshop on multi-view modeling and analysis of visual scenes (MVIEW’99)*, IEEE, 1999, pp. 37–44.
- [58] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [59] J. Bradbury *et al.*, *JAX: Composable transformations of Python+NumPy programs*. (2018).
- [60] B. Deng, J. T. Barron, and P. P. Srinivasan, “JaxNeRF: An efficient JAX implementation of NeRF.” 2020.
- [61] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” 2017.
- [62] C. Lugaresi *et al.*, “MediaPipe: A framework for building perception pipelines,” *arXiv preprint arXiv:1906.08172*, 2019.
- [63] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, “Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

- [64] Z. Teed and J. Deng, “RAFT: Recurrent all-pairs field transforms for optical flow,” in *European conference on computer vision*, Springer, 2020, pp. 402–419.
- [65] R. L. Gregory, *The intelligent eye*. 1970.
- [66] K. Wilson, D. Bindel, and N. Snavely, “When is rotations averaging hard?” in *European conference on computer vision*, Springer, 2016, pp. 255–270.
- [67] brusspup, “Amazing t-rex illusion!” Dec. 2013.