Nerfies: Deformable Neural Radiance Fields (Supplementary Materials)

Keunhong Park[1*] Utkarsh Sinha[2] Jonathan T. Barron[2] Sofien Bouaziz[2]
Dan B Goldman[2] Steven M. Seitz[1,2] Ricardo Martin-Brualla[2]
[1]University of Washington [2]Google Research
nerfies.github.io

A. Details of SE(3) Field Formulation

As mentioned in the main text, we encode a rigid transform as a screw axis [14] $\mathcal{S} = (\mathbf{r}; \mathbf{v}) \in \mathbb{R}^6$ where

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

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

$$[\mathbf{x}]_\times = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix} . \quad (2)$$

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

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

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

$$e^{\mathcal{S}} = \begin{pmatrix} e^{\mathbf{r}} & \mathbf{p} \\ 0 & 1 \end{pmatrix} . \quad (4)$$

The deformed point is then given by $\mathbf{x}' = e^{\mathcal{S}}\mathbf{x}$.

Why does an SE(3) field work better? Consider the example in Fig. 1 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., $\mathbf{t_2}$) need translations of small magnitude while points towards the outside (e.g., $\mathbf{t_1}$) 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 $\theta = \theta_1 = \theta_2$.

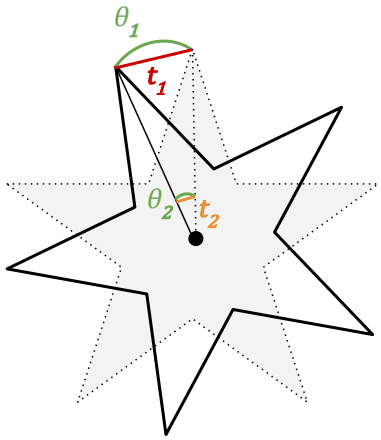


Figure 1: Here we illustrate why a rigid transformation field works better than a translation field with a simple toy example where a star is rotated counter-clockwise around its center. A translation field requires different parameters for every point to encode the rotation (e.g., $\|\mathbf{t_1}\| \gg \|\mathbf{t_2}\|$) whereas a rotation field only needs a single parameter to encode the rotation (e.g., $\theta_1 = \theta_2$). More details in §A.

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 Fig. 3.

NTK: We also show a visualization of the neural tangent kernel (NTK) induced by our annealed positional encoding in Fig. 2. This figure shows the normalized NTK for an 8

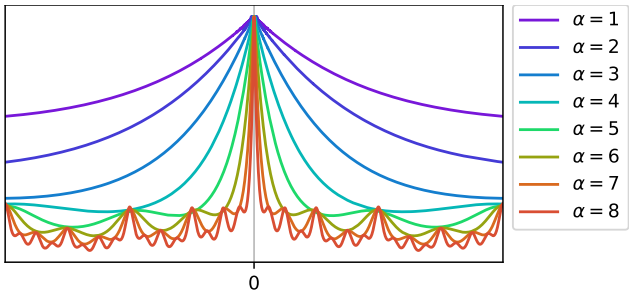


Figure 2: Visualizations of the neural-tangent kernel (NTK) [9] of our annealed positional encoding for different values of α. Our coarse-to-fine optimization scheme works by easing in the influence of each positional encoding frequency through a parameter α. This has the effect of shrinking the bandwidth of the NTK corresponding to the deformation MLP as α is increased, thereby allowing higher frequency deformations.

layer MLP of width 256. Note how the bandwidth of the interpolation kernel gets narrower as the value of α 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 $\mathbf{M}=\operatorname{diag}\left(1,\ldots,1,\det(\mathbf{U}\mathbf{V}^T)\right)$. It follows that:

$$\begin{aligned}
\|\mathbf{J}-\mathbf{R}\|_F &= \left\|\mathbf{U}\mathbf{D}\mathbf{V}^T-\mathbf{U}\mathbf{M}\mathbf{V}^T\right\|_F && (5)\\
&= \left\|\mathbf{U}(\mathbf{D}-\mathbf{M})\mathbf{V}^T\right\|_F && (6)\\
&= \sqrt{\operatorname{tr}\left(\mathbf{U}(\mathbf{D}-\mathbf{M})\mathbf{V}^T\mathbf{V}(\mathbf{D}-\mathbf{M})\mathbf{U}^T\right)} && (7)\\
&= \sqrt{\operatorname{tr}\left(\mathbf{U}(\mathbf{D}-\mathbf{M})^2\mathbf{U}^T\right)} && (8)\\
&= \sqrt{\operatorname{tr}\left((\mathbf{D}-\mathbf{M})^2\right)} && (9)\\
&= \sqrt{\sum_j (\sigma_j - m_j)^2}\,, && (10)
\end{aligned}$$

where m_j is the jth diagonal of $\mathbf{M}$ and σ_j is the jth 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 Fig. 4 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.

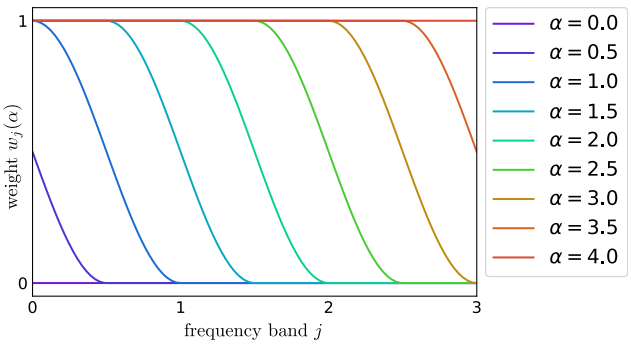


Figure 3: A visualization of the window function $w_j(\alpha)$ for the annealed positional encoding. We show an example with a maximum number of frequency bands of $m=4$ where $j\in\{0,\ldots,m-1\}$. $\alpha=0$ sets the weight of all frequency bands to zero leaving only the identity mapping, while an $\alpha=4$ 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.

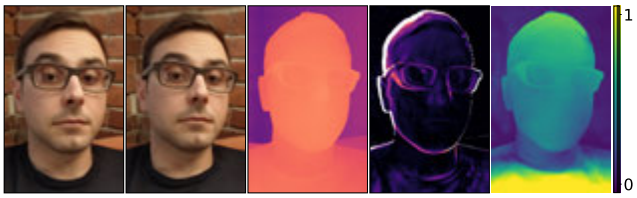


Figure 4: Users move even when trying not to. Here we visualize the depth difference and deformation magnitude between the template and an observation.

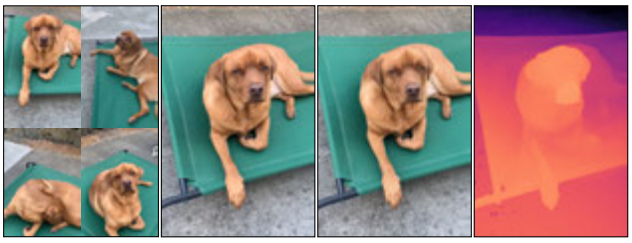


Figure 5: Not relying on domain specific priors enables our method to reconstruct any deformable object. In this case, the dog fails to stay still, yet we recover an accurate model.

Domain Agnostic: In Fig. 5, Fig. 14, and Fig. 15. we show that our method works agnostic of the type of subject.

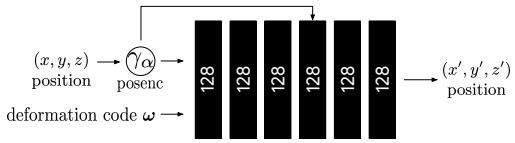


Figure 6: A diagram of our deformation network. The deformation network takes a position encoded position $\gamma_\alpha(\mathbf{x})$ using our coarse-to-fine annealing parameterized by α, along with a deformation code $\boldsymbol{\omega}$ and outputs a deformed position $\mathbf{x}'$. The architecture is identical for all of our experiments.

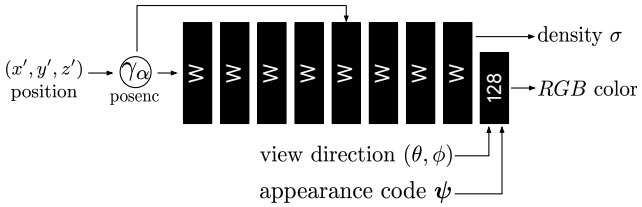


Figure 7: A diagram of the canonical NeRF network. Our network is identical to the original NeRF MLP, except we provide an appearance latent code $\boldsymbol{\psi}$ along with the view direction to allow modulating the appearance as in the NeRF-A model of [15]. The width W of the network is defined according to Tab. 2.

E. Additional Implementation Details

Architecture Details: We provide architecture details of the deformation field network and canonical NeRF networks in Figures 6 and 7 respectively.

Training: We train our network using the Adam optimizer [10] with a learning rate exponentially decayed by a factor 0.1 until the maximum number of iterations is reached. The exact hyper-parameters for each configuration are provided in Tab. 2.

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 $\varepsilon \sim \mathcal{N}(0, 0.001)$ and use a robust Geman-McClure loss function [7] with $\alpha = -2$ and $c = 0.001$ implemented as per Barron [1].

Implementation: We extend the JAX [3] implementation of NeRF [6] 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 [17]. 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 [5]. We then use COLMAP [19] 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 selfie 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 *nerfies*, like figure-eight camera paths in front of the user. We compute the 2D facial landmarks using MediaPipe's face mesh [13], 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 [12] 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 Fig. 14 and Fig. 15.

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 images)		Beanie (74 images)		Curls (57 images)		Kitchen (40 images)		Lamp (55 images)		Toby Sit (308 images)		Mean		Drinking (193 images)		Tail (238 images)		Badminton (356 images)		Broom (197 images)		Mean	
	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓
NeRF [16]	.619	.474	.580	.583	.504	.616	.695	.434	.656	.444	.793	.463	.641	.502	.619	.397	.676	.571	.771	.392	.643	.667	.677	.506
NeRF + latent	.695	.463	.687	.535	.619	.539	.746	.403	.735	.386	.798	.385	.713	.452	.855	.233	.800	.404	.850	.308	.688	.576	.798	.380
Neural Volumes [12]	.503	.616	.562	.595	.538	.588	.609	.569	.563	.533	.583	.473	.560	.562	.771	.198	.503	.559	.219	.516	.515	.544	.502	.454
NSFF†	.599	.490	.606	.483	.458	.569	.762	.394	.673	.343	.825	.391	.654	.445	.918	.175	.672	.363	.649	.368	.577	.357	.704	.316
$\gamma(t)$ + Trans† [11]	.781	.354	.737	.471	.732	.426	.823	.344	.836	.283	.870	.420	.796	.383	.910	.151	.882	.391	.927	.221	.750	.627	.867	.347
Ours ($\lambda = 0.01$)	.826	.305	.786	.391	.842	.319	.878	.280	.888	.232	.806	.159	.838	.281	.894	.0872	.754	.161	.926	.130	.674	.245	.812	.156
Ours ($\lambda = 0.001$)	.840	.307	.805	.391	.846	.312	.863	.279	.886	.230	.805	.174	.841	.282	.881	.0962	.731	.175	.922	.132	.605	.270	.785	.168
No elastic	.809	.317	.824	.382	.830	.322	.851	.290	.889	.230	.821	.257	.837	.300	.890	.0863	.735	.174	.919	.132	.593	.287	.784	.170
No coarse-to-fine	.828	.312	.771	.408	.841	.321	.877	.277	.867	.242	.807	.244	.832	.301	.892	.0960	.763	.257	.912	.151	.695	.406	.815	.228
No SE3	.823	.314	.782	.401	.839	.317	.870	.282	.872	.235	.810	.206	.833	.293	.895	.0867	.715	.191	.899	.156	.599	.276	.777	.177
Ours (base)	.828	.319	.737	.456	.818	.345	.829	.323	.851	.254	.792	.184	.809	.314	.894	.127	.768	.298	.894	.173	.695	.503	.813	.275
No BG Loss	.779	.317	.758	.395	.696	.371	.844	.290	.806	.260	.775	.145	.776	.296	.893	.0856	.719	.210	.875	.161	.593	.330	.770	.196

Table 1: SSIM and LPIPS 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) [11] proposes to model dynamic scenes by directly conditioning the NeRF with a position-encoded time variable $\gamma(t)$, 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 $\gamma(t)$ while our method uses a deformation field; (b) NSFF uses a position-encoded time variable ($\gamma(t)$) to condition each observation whereas our method uses a per-example latent code [2]; (c) NSFF uses depth from MIDAS [18] and optical flow from RAFT [20] as supervision whereas our method only uses a photometric loss.

We quantitatively compare with NSFF in Tab. 1 of the paper and in Tab. 1, and show corresponding qualitative results in Fig. 14 and Fig. 15. 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 Tab. 1.

G. Additional Results

We show qualitative results from each of the sequences presented in our quantitative evaluation (Tab. 1 of paper, Tab. 1) for quasi-static scenes (Fig. 14) and dynamic scenes (Fig. 15).

H. Limitations

Topological Variation: Our method struggles when the scene has motion which varies the topology of the scene.

Config	Resolution	Steps	Learning Rate	Batch Size	# Samples Fine	# Samples Coarse	Width W
Full	1080p	1M	7.5e-4	3072	256	256	256
Half	540p	100K	1e-3	8096	128	128	128

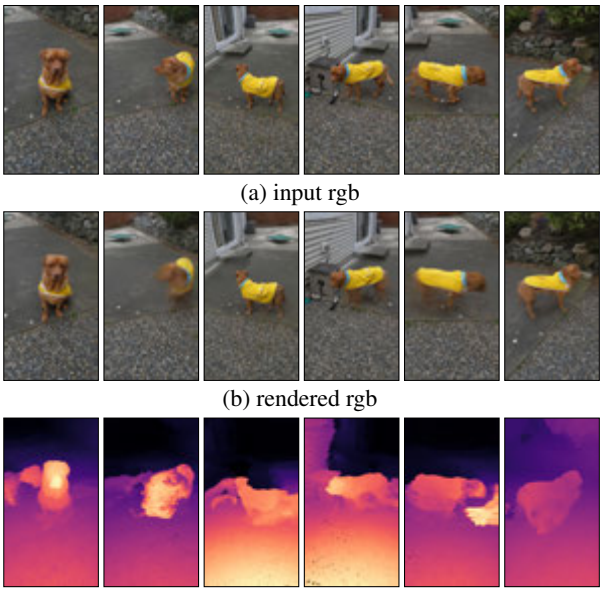
Table 2: Here we provide the hyper-parameters used for each configuration. Full is the full resolution configuration used in our qualitative results. Half is half the resolution of Full and is used for our quantitative evaluation and ablation studies.

Figure 8: This concave mask of the Swedish tennis player Bjorn Borg appears to be convex due to the hollow-face illusion. By Skagedal, 2004 (Public Domain).

example inputs novel views for same deformation code

Figure 9: If the user's gaze consistently follows the camera, the reconstructed *nerfie* represents the user's gaze as geometry, akin to the Hollow-Face illusion [8]. 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.

For example, when a person opens their mouth (as in Fig. 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 mouth open. Suppose that $\mathbf{x}_U$ and $\mathbf{x}_L$ are two adjacent points near the seam of the lips, and the $\mathbf{x}_U$ is on the upper



(a) input rgb

(b) rendered rgb

(c) rendered depth

Figure 10: 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.

lip and $\mathbf{x}_L$ 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 Fig. 10, 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) [21]. 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 'fit' of the two coins must get worse before getting better when rotating towards the global minimum.

Figure 11: 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).

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 Fig. 10 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 [4]. We show an example of this illusion in Fig. 8.

We observe that the ambiguity which causes this illusion can also be a failure more for our method. In Fig. 9, 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.

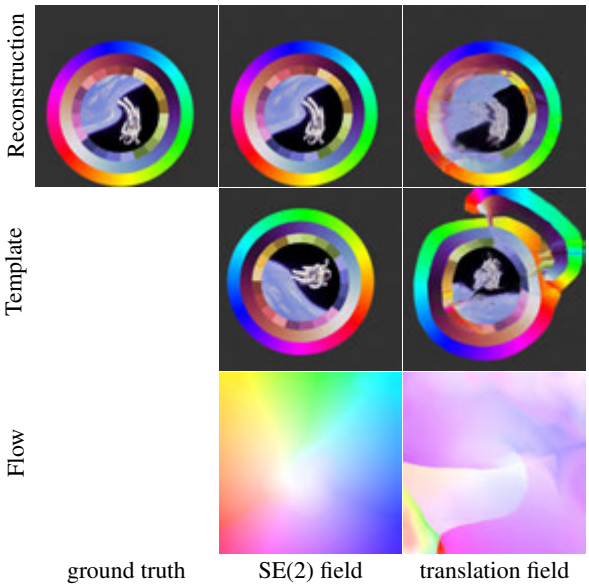


Figure 12: A comparison of our SE(2) field and a translation field. The translation parameterization has difficulty rotating groups of distant pixels whereas the rigid transformation successfully finds the correct orientation.

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 non-linearly distorted near the center. We show the full dataset in Fig. 11. Akin to our deformable NeRF setting, the task is to 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 Sec. 3.2.

Deformation Formulation: Fig. 12 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 Fig. 13. With a small number of frequencies ($m = 1$) 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 ($m = 2 \ldots 6$) 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.

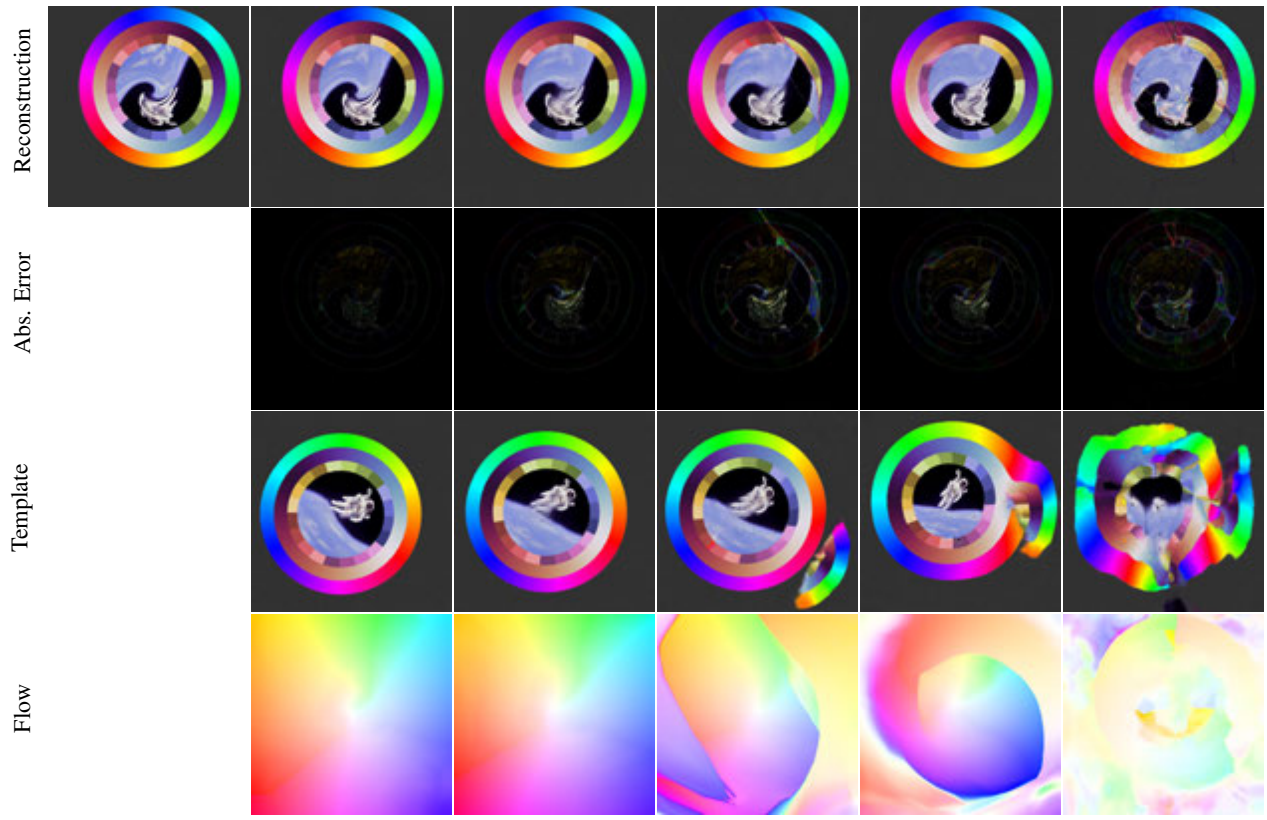


Figure 13: We show how the optimization changes depending on the number of frequency bands in the positional encoding of the deformation field. With 1 frequency, the model find the correct orientation of all images but is unable to model the high frequency distortion near the center. If we increase the 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.

Figure 14: Comparisons of baselines and our method on quasi-static scenes. PSNR / LPIPS metrics on bottom right with best colored red.

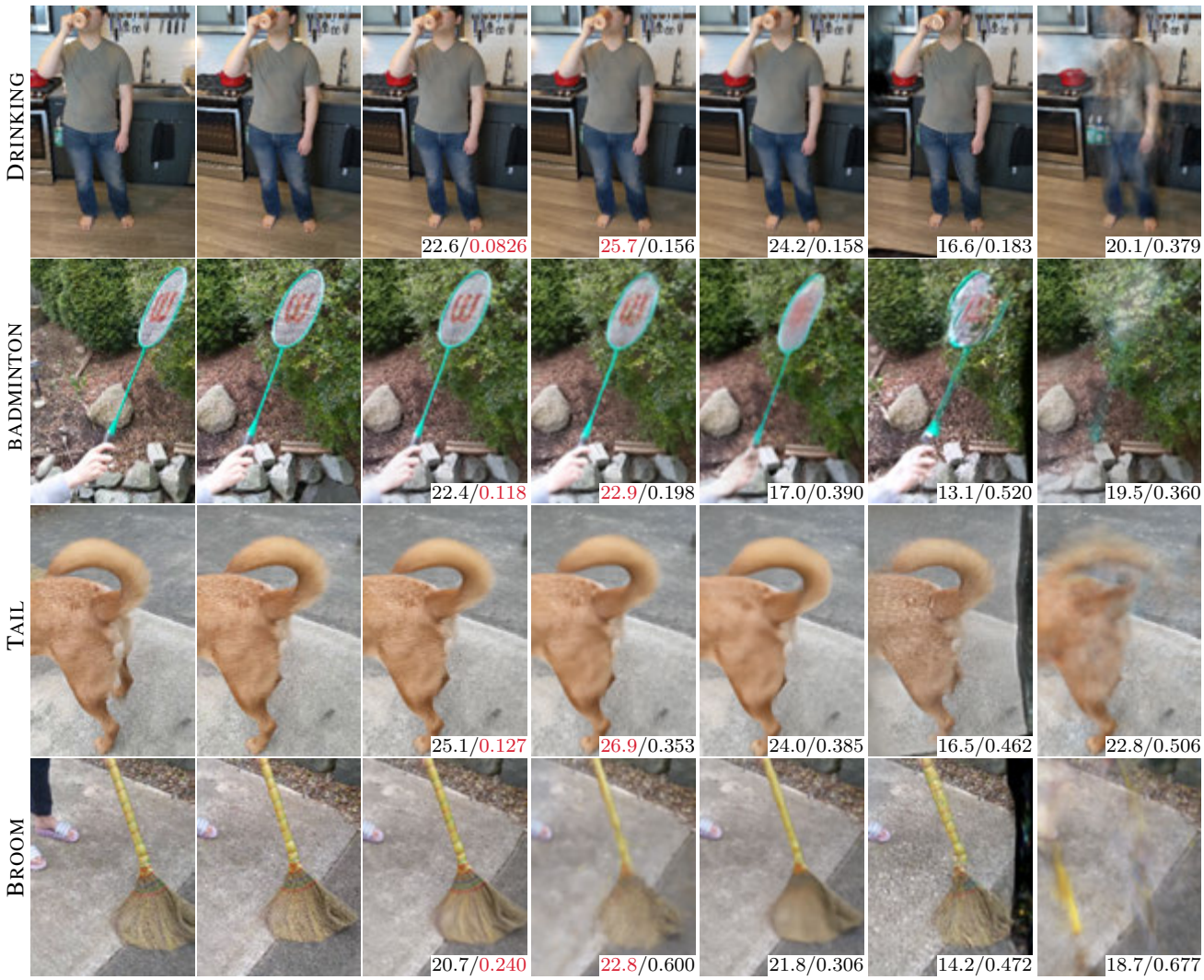


Figure 15: Comparisons of baselines and our method on dynamic scenes. PSNR / LPIPS metrics on bottom right with best colored red.

References

[1] Jonathan T. Barron. A general and adaptive robust loss function. *CVPR*, 2019. 3

[2] Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the latent space of generative networks. *ICML*, 2018. 4

[3] 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. JAX: composable transformations of Python+NumPy programs, 2018. 3

[4] brusspup. Amazing t-rex illusion!, Dec 2013. 5

[5] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017. 3

[6] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020. 3

[7] Stuart Geman and Donald E. McClure. Bayesian image analysis: An application to single photon emission tomography. *Proceedings of the American Statistical Association*, 1985. 3

[8] Richard Langton Gregory. The intelligent eye. 1970. 4

[9] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018. 2

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[11] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2020. 4, 8, 9

[12] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: learning dynamic renderable volumes from images. *ACM ToG*, 2019. 3, 4, 8, 9

[13] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. MediaPipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019. 3

[14] Kevin M Lynch and Frank C Park. *Modern Robotics*. Cambridge University Press, 2017. 1

[15] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. *CVPR*, 2021. 3

[16] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *ECCV*, 2020. 4, 8, 9

[17] José Luis Pech-Pacheco, Gabriel Cristóbal, Jesús Chamorro-Martinez, and Joaquín Fernández-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In *ICPR*, volume 3, pages 314–317. IEEE, 2000. 3

[18] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 4

[19] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. *CVPR*, 2016. 3

[20] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, pages 402–419. Springer, 2020. 4

[21] Kyle Wilson, David Bindel, and Noah Snavely. When is rotations averaging hard? In *European Conference on Computer Vision*, pages 255–270. Springer, 2016. 5