Intrinsic Neural Fields: Learning Functions on Manifolds – Supplementary Material

Lukas Koestler*,1, Daniel Grittner*,1, Michael Moeller[2], Daniel Cremers[1], and Zorah Lähner[2]

[1] Technische Universität München
[2] Universität Siegen

In this supplementary, we elaborate on the implementation details for our intrinsic neural fields (**??**), discuss the details of our experiments (**??**), show that our method is not overly initialization-dependent (**??**), and, finally, provide further theoretical results (**??**).

The high-resolution intrinsic neural texture field on the human model that we showcase in Fig. 1 is available as a textured mesh as an `.obj` in the supplementary and on sketchfab[3]. We would like to note that the small texture seams are not due to our method, but due to the conversion from our network to a uv texture map. Intrinsic neural fields do not possess the discontinuities which are present in the uv map. The texture is created by an inverse uv lookup of each texel and an evaluation of the intrinsic neural field at the corresponding point on the manifold. We provide the textured mesh as a convenient possibility for qualitative inspection with current tools but it is never used to evaluate the proposed method qualitatively or quantitatively in the paper. We also provide a qualitative comparison of a reconstruction of a specular object with Meshroom[4], a free photogrammetry software. In this case we used the 3D geometry from Meshroom but replaced the texture with view-dependent results from our method. The video can be found here: https://www.youtube.com/watch?v=L8Q6gp2d7nU.

A Implementation Details

For our method, we calculate the eigenfunctions of the Laplace-Beltrami operator of a given triangle mesh once by solving the generalized eigenvalue problem for the first d eigenvalues using the robust Laplacian by Sharp and Crane [**?**]. If the given geometry is a pointcloud, we create a local triangulation around each point which lets us perform a normal ray-mesh intersection. Additionally, the robust Laplacian [**?**] supports calculating eigenfunctions on pointclouds.

Depending on whether the viewing direction is taken into account, we use the respective network architectures shown in **??** for our experiments. Both networks take as input a point $\mathbf{p}$ from the surface of the discrete 2-manifold embedded into its d eigenfunctions. Since $\mathbf{p}$ might not be a vertex, we linearly interpolate the eigenfunctions of the vertices v_i, v_j, and v_k, which span the triangle face

[3] https://skfb.ly/otvFK
[4] https://alicevision.org/

where $\mathbf{p}$ is located, using the barycentric coordinates. For embedding the unit viewing direction $\mathbf{d} \in \mathbb{R}^3$, we use the sine/cosine positional encoding [**?**].

During the training, we randomly sample preprocessed rays from the whole training split. We use a batch size of 4096 across all our experiments. As optimizer, we use Adam with the default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$). Our loss function is the mean L1 loss over a batch of randomly sampled rays $\mathcal{B}$

$$\mathcal{L}_1 = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{p} \in \mathcal{B}} \| F_\theta(\mathbf{p}) - c_{\text{gt}}(\mathbf{p}) \|_1 \tag{11}$$

where F_θ is an intrinsic neural field and $c_{\text{gt}}(\mathbf{p})$ is the ground-truth RGB color.

B Experimental Details

B.1 Modification of NeuTex

In order to make the comparison between our method and NeuTex [**?**] fair, we adapt the latter one to the setting of a given geometry. Since the geometry is known in our experimental setup, we remove the latent vector used for learning a representation of the geometry. Additionally, we remove the volume density from NeuTex because our experiments are focused on learning a function on the surface of a given 2-manifold. Furthermore, we do not incorporate view dependence in the experiments of Sec. 5.1. Hence, we use the the provided, non-view dependent MLP architecture for F_{tex} from the official Github repository[5]. We, additionally, add a sigmoid non-linearity to the last linear layer to ensure that the RGB color values are in $[0, 1]$. The overall architecture is shown in **??**.

Due to the higher complexity of additionally learning a mapping between the surface of the manifold and the uv-space, we pretrain the mapping networks F_{uv} and F_{uv}^{-1}. In each training iteration, we randomly sample $N = 25,000$ points from the sphere and map them into the 3D world coordinate space of the geometry using F_{uv}^{-1}. Then, we map the predicted 3D world points back onto the sphere using F_{uv}. We train for $200,000$ iterations using the Adam optimizer with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$) and a learning rate of 0.0001. The loss function is a combination of the mean Chamfer distance $\mathcal{L}_{\text{chamfer}}$ between the predicted 3D world points and the vertices of the mesh and the mean 2D-3D-2D cycle loss between the sampled and predicted uv-points u_i

$$\mathcal{L}_{\text{cycle}} = \frac{1}{N} \sum_{i=0}^{N-1} \| F_{\text{uv}}(F_{\text{uv}}^{-1}(u_i)) - u_i \|_2^2. \tag{12}$$

After the pretraining, we employ a combination of the rendering loss and the 3D-2D-3D cycle loss as the loss function for learning a surface function on a

[5] https://github.com/fbxiang/NeuTex

Table 4: Hyperparameter table: Texture reconstruction.

Hyperparameter	Value
`optimizer`	Adam with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$)
`learning rate`	0.0001
`batch size`	4096
`image size`	512×512
`random seed`	0
`eigenfunctions`	1 to 256, 1794 to 2304, 3841 to 4096 where 0 is the constant eigenfunction
`epochs`	1000

given geometry:

$$\mathcal{L}_{\text{neutex}} = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{p} \in \mathcal{B}} \|F_{\text{tex}}(F_{\text{uv}}(\mathbf{p})) - c_{gt}\|_2^2 + \|F_{\text{uv}}^{-1}(F_{\text{uv}}(\mathbf{p})) - \mathbf{p}\|_2^2. \tag{13}$$

For the experiment in Sec. 5.1, we, additionally, increase the embedding size of the sphere coordinates to 1023 for NeuTex, so that it is similar to the other methods. For the sine/cosine positional encoding, we select the frequency bands from $[0, 6]$ linearly spaced because it covers a range that is similar to RFF with $\sigma = 8$.

B.2 Texture Reconstruction

In this experiment, we use the same cat[6] and human[7] mesh as in [**?**]. The 2D views of the meshes are rendered using Blender and blenderproc [**?**]. For the experiments in Sec. 5.1 and Sec. 5.2, we randomly render 5 training, 100 validation, and 200 test 512×512 views. The training views are visualized in **??**. The hyperparameters are given in **??**. Further qualitative results for Sec. 5.1 can be found in **??**. The human shown in Fig. 1 was trained using a 4096×4096 high-resolution dataset. It consists of 20 training, 20 validation, and 20 test views that were randomly generated. All training views are visualized in **??**. In **??** the hyperparameters for the high-resolution human are shown.

[6] https://free3d.com/3d-model/cat-v1–522281.html
[7] https://www.turbosquid.com/3d-models/water-park-slides-3d-max/1093267

Table 5: Hyperparameter table: High-resolution texture reconstruction.

Hyperparameter	Value
`optimizer`	Adam with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$)
`learning rate`	0.0001
`batch size`	4096
`image size`	4096×4096
`random seed`	0
`eigenfunctions`	1 to 4096 where 0 is the constant eigenfunction
`epochs`	500

B.3 Discretization-agnostic Intrinsic Neural Fields

We use the same datasets for the cat and human as in Sec. 5.1 but generate different discretizations of the meshes with the scripts from the Github project[8] by Sharp et al. [**?**]. The hyperparameters can be found in **??**. The scripts and the meshes will be released together with the code.

B.4 Intrinsic Neural Field Transfer

For the neural texture transfer, we train an intrinsic neural field on the cat from Sec. 5.1 with the hyperparameters shown in **??**. Since the input to our method is only the first d LBO eigenfunctions, we can reuse the network on the cat on other shapes without retraining as long as we know how to transfer the eigenfunctions. This is exactly what functional maps [**?**] do. We use the method of [**?**], which works with both isometric and non-isometric pairs, to calculate a correspondence P between the cat $\mathcal{C}$ and a target $\mathcal{T}$, and obtain the functional map by projecting $C = \Phi_{\mathcal{T}}^{\top} P \Phi_{\mathcal{C}}$. Instead of using the eigenfunctions $\Phi_{\mathcal{T}}$ of the target shape directly, we use $\Phi_{\mathcal{T}} C$ as input to the network.

B.5 Real-world Data and View Dependence

For experiments with real-world data, we select the BigBIRD dataset [**?**]. **??** shows quantitative results on ten instances and the main paper contains qualitative results on the objects 'detergent' and 'cinnamon toast crunch'. The dataset provides images from different directions, foreground-background segmentation masks, camera calibration, and a reconstructed mesh of the geometry of each

[8] https://github.com/nmwsharp/discretization-robust-correspondence-benchmark

Table 6: Hyperparameter table: Texture transfer.

Hyperparameter	Value
`optimizer`	Adam with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$)
`learning rate`	0.0001
`batch size`	4096
`image size`	512×512
`random seed`	0
`eigenfunctions`	1 to 512 where 0 is the constant eigenfunction
`epochs`	500

object. The provided meshes and the object masks do not align well, which can cause color bleeding from the background into the reconstruction. Hence, we improve the masks using intensity thresholding and morphological operations. Specifically, potential background pixels are identified based on their intensity due to the mostly white background. They are removed if they are close to the boundary of the initial mask. Finally, a small margin of the mask is eroded to limit the number of false positive mask pixels. This process could be replaced by a more advanced method for a large-scale experiment on real-world data. For both objects, we train our method on 60 evenly-spaced views from a 360 degree perspective. The view used for qualitative comparison is centered between two training views. The model for this experiment implements the view-dependent network architecture visualized in **??** and uses the hyperparameters shown in **??**. We will release the preprocessing code and the training data used in this experiment along with the final publication.

C Initialization Dependence of Intrinsic Neural Fields

In order to test the dependence of our method on the initialization, we repeat the experiment from Sec. 5.1 with different seeds. The results can be found in **??**. Our method has a relative standard deviation of roughly 1% for DSSIM and LPIPS and only about 0.2% for PSNR, which shows that intrinsic neural fields are not overly initialization-dependent.

D Theory

In this section, we provide further details regarding the theory of intrinsic neural fields. In **??** we demonstrate that the composed neural tangent kernel (NTK) can

Table 7: Hyperparameter table: Real-world data and view dependence.

Hyperparameter	Value
`optimizer`	Adam with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$)
`learning rate`	0.0002
`learning rate schedule`	plateau with factor=0.1, patience=10, threshold=0.0001
`batch size`	16384
`image size`	2848×4272
`random seed`	0
`eigenfunctions`	1 to 4096 where 0 is the constant eigenfunction
`epochs`	500

Table 8: Quantitative evaluation (PSNR) on the first ten objects of the BigBIRD dataset: 3m high tack spray adhesive, advil liqui gels, band aid clear strips, band aid sheer strips, blue clover baby toy, bumblebee albacore, campbells chicken noodle soup, campbells soup at hand creamy tomato, canon ack e10 box, cheez it white cheddar. We skip "aunt jemima original syrup" as the mesh in the dataset is broken.

	1	2	3	4	5	6	7	8	9	10	avg
Ours	**31.8**	**29.9**	**32.7**	**31.9**	36.2	**30.2**	**30.4**	**31.9**	**34.1**	**31.8**	**32.09**
RFF	30.9	28.5	31.0	29.9	**36.6**	28.0	28.8	30.0	33.7	30.7	30.81

be non-stationary for a general 2-manifold. In **??** the proof for Theorem 1 is given specifically for 1-manifolds. Finally, in **??** we provide further theoretical results regarding the NTK, specifically **??**, which was used in the proof of Theorem 1.

D.1 Theorem 1 on 1-Manifolds

The proof for Thm. 1 only considered n-spheres. Here, we will give a short proof why it extends to general closed compact 1-manifolds. The proof depends only on properties of the spherical harmonics which are equivalent to the LBO eigenfunctions of closed 1-manifolds.

Since the proof in the main paper is for n-spheres, it also holds for circles of arbitrary radius which are 1-spheres. Notice that all closed compact 1-manifolds $\mathcal{N}$ are isometric to the circle with radius equal to the circumference of $\mathcal{N}$. This is quite obvious if one considers that the geodesic distance between two points

Table 9: Initialization dependence for intrinsic neural fields. We retrain our method with different seeds on the texture reconstruction experiment from Sec. 5.1. The results show that intrinsic neural fields are not overly dependent on the initialization. Additionally, the results presented in Sec. 5.1 with seed 0 are not cherry picked.

	Seed	0	1	2	3	4	5	6	7	8	9	Avg.	Std.
cat	PSNR ↑	34.82	34.73	34.80	34.81	34.85	34.95	34.87	35.00	34.77	**35.01**	34.86	0.262%
	DSSIM ↓	0.095	0.096	0.096	0.096	0.093	0.094	0.093	0.092	0.096	**0.091**	0.094	1.846%
	LPIPS ↓	0.153	0.155	0.156	0.158	0.153	0.158	0.154	**0.151**	0.156	0.156	0.155	1.305%
human	PSNR ↑	32.48	32.43	32.35	32.47	32.47	**32.56**	32.50	32.50	32.42	32.42	32.46	0.171%
	DSSIM ↓	0.121	0.122	0.124	0.121	0.121	**0.120**	0.121	0.121	0.121	0.122	0.121	0.843%
	LPIPS ↓	0.306	0.309	0.305	0.309	0.305	0.306	0.306	**0.301**	0.303	0.305	0.306	0.763%

on a curve is simply the arc length between them, the geodesic distance on a closed 1-manifold is the minimum of both possible arc lengths. Therefore, the geodesic distances of any 1-manifolds with fixed circumference r is invariant to its extrinsic embedding, and all 1-manifold with circumference r are isometric to each other.

The spherical harmonics are the eigenfunctions of the Laplace-Beltrami operator and those are invariant under isometries. Therefore, the spherical harmonics and all their properties transfer to general closed compact 1-manifolds and the proof still holds.

D.2 Neural Tangent Kernel

In this section, we prove **??** that was used in the proof of Thm. 1. Additionally, we briefly discuss the positive definiteness of the NTK. As in the main paper, we consider the same setting as Jacot et al. [**?**].

Lemma 1. *Let* $\mathrm{k}_{NTK} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ *be the neural tangent kernel for a multilayer perceptron (MLP) with non-linearity* σ*. If the inputs are restricted to a* ***scaled*** *hypersphere* $\|x\| = r$ *then there holds*

$$\mathrm{k}_{NTK}(x, x') = \mathrm{h}_{NTK}(\langle x, x' \rangle)\,, \tag{14}$$

for a scalar function $\mathrm{h}_{NTK} : \mathbb{R} \to \mathbb{R}$*.*

Proof. For this proof we adopt the notation of Jacot et al. [**?**] to simplify following both papers simultaneously. We give a detailed proof, as this also gives good insight into the NTK. Let all requirements be equivalent to the ones used for [**?**, Prop. 1]. Jacot et al. show that the neural network Gaussian process (NNGP)

has covariance $\Sigma^{(L)}$ defined recursively

$$\Sigma^{(1)}(x, x') = \frac{1}{n_0} x^\top x' + \beta^2 \tag{15}$$

$$\Sigma^{(L+1)}(x, x') = \mathbb{E}_{(u,v)\sim N(0,\Lambda^{(L)}(x,x'))}[\sigma(u)\sigma(v)] + \beta^2 \tag{16}$$

$$\Lambda^{(L)}(x, x') = \begin{pmatrix} \Sigma^{(L)}(x, x) & \Sigma^{(L)}(x, x') \\ \Sigma^{(L)}(x', x) & \Sigma^{(L)}(x', x') \end{pmatrix}, \tag{17}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is the non-linearity of the network and β is related to the bias of the network. We use $u = f(x)$ and $v = f(x')$ instead of the Gaussian process notation used in [**?**]. We prove by induction that $\Sigma^{(L)}(x, x')$ depends only on $x^\top x'$, which also implies that $\Sigma^{(L)}(x, x)$ does not depend on x because $x^\top x = r^2$. For $L = 1$ this follows directly from the definition. Assume now that for L we have that $\Sigma^{(L)}(x, x')$ depends only on $x^\top x'$. It directly follows that $\Lambda^{(L)}(x, x')$ and thus $\Sigma^{(L+1)}(x, x')$ also only depend on $x^\top x'$, which is the induction step.

Given the NNGP kernel, the neural tangent kernel (NTK) is given by Theorem 1 from Jacot et al:

$$\Theta^{(1)}_\infty(x, x') = \Sigma^{(1)}(x, x') \tag{18}$$

$$\Theta^{(L+1)}_\infty(x, x') = \Theta^{(L)}_\infty(x, x')\dot{\Sigma}^{(L+1)}(x, x') + \Sigma^{(L+1)}(x, x') \tag{19}$$

$$\dot{\Sigma}^{(L+1)}(x, x') = \mathbb{E}_{(u,v)\sim N(0,\Lambda^{(L)}(x,x'))}[\dot{\sigma}(u)\dot{\sigma}(v)] + \beta^2\,, \tag{20}$$

where $\dot{\sigma}$ is the derivative of the non-linearity. By a similar induction argument to above we obtain that $\Theta^{(L)}_\infty(x, x')$ only depends on $x^\top x'$ and hence that $\Theta^{(L)}_\infty(x, x)$ does not depend on x. In the notation of our **??** this means that $\mathrm{k}_{\mathrm{NTK}}$ only depends on $\langle x, x'\rangle$ and can thus be written as $\mathrm{h}_{\mathrm{NTK}}(\langle x, x'\rangle)$ for a scalar function $\mathrm{h}_{\mathrm{NTK}} : \mathbb{R} \to \mathbb{R}$. □

Positive-definiteness of the NTK. In the proof of Theorem 1, we used the fact that the NTK is positive definite as shown by [**?**, Prop. 2]. Their proposition is stated for $\|x\| = 1$ and the extension to $\|x\| = r$ requires only slight changes, which we will detail in the following. In the third step of Jacot et al.'s proof when doing the change of variables to arrive at their Eqn. 1 the following changes

$$\mathbb{E}_{(X,Y)\sim N(0,\tilde{\Sigma})}[\sigma(X)\sigma(Y)] + \beta^2 = \hat{\mu}\left(\frac{n_0\beta^2 + x^\top x'}{n_0\beta^2 + r^2}\right) + \beta^2\,, \tag{21}$$

where $\hat{\mu} : [-1, 1] \to \mathbb{R}$ is the dual in the sense of [**?**, Lem. 2] of the function $\mu : \mathbb{R} \to \mathbb{R}$ defined by $\mu(x) = \sigma\left(x\sqrt{r^2/n_0 + \beta^2}\right)$. Finally, in step 5 of their proof

$$\nu(x^\top x') = \nu(r^2\rho) = \beta^2 + \sum_{i=0}^{\infty} a_i \left(\frac{n_0\beta^2 + r^2\rho}{n_0\beta^2 + r^2}\right)^i. \tag{22}$$

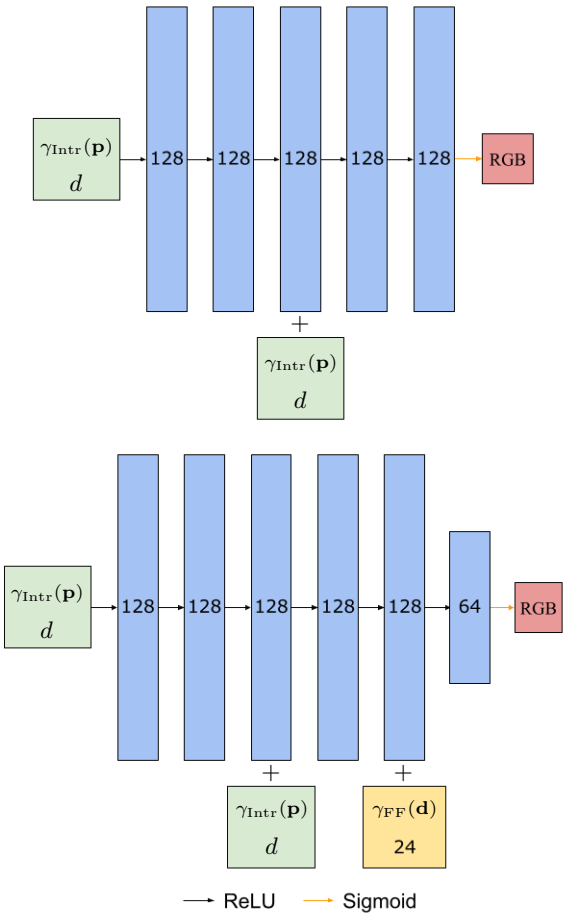


Fig. 8: Network architecture. We use a similar network architecture as used for NeRF [**?**]. A point on the 2-manifold is described by $\mathbf{p}$. The unit ray direction is represented by $\mathbf{d}$. The notations γ_{Intr} and γ_{FF} represent the proposed eigenfunction embedding and the sine/cosine positional encoding [**?**] respectively. The + sign denotes concatenation. The second architecture is used in the experiments of Sec. 5.4 while we use the first architecture in all other experiments.

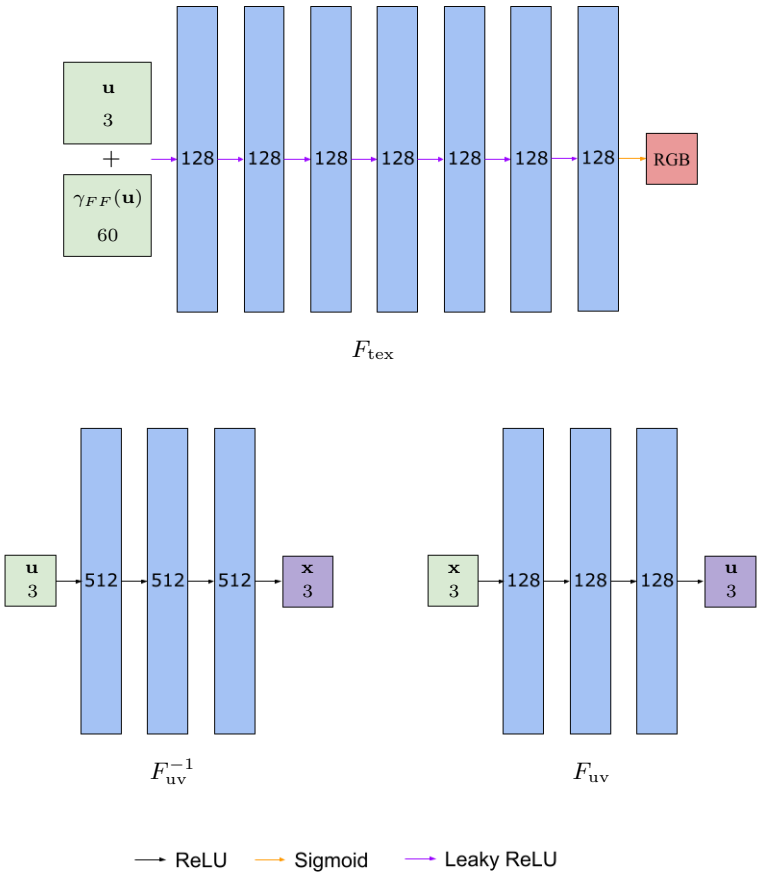


Fig. 9: Modified network architecture for NeuTex [**?**]. We remove the volume density as well as the view dependence enabling NeuTex to learn a simpler setting because the geometry is known and the textures are diffuse in the experiment of Sec. 5.1. A 3D coordinate on the sphere representing the uv-space is described by $\mathbf{u}$. The symbol $\mathbf{x}$ is a 3D world coordinate on the surface of the given 2-manifold. The notation for the sine/cosine positional encoding [**?**] is γ_{FF}. The + sign denotes concatenation.

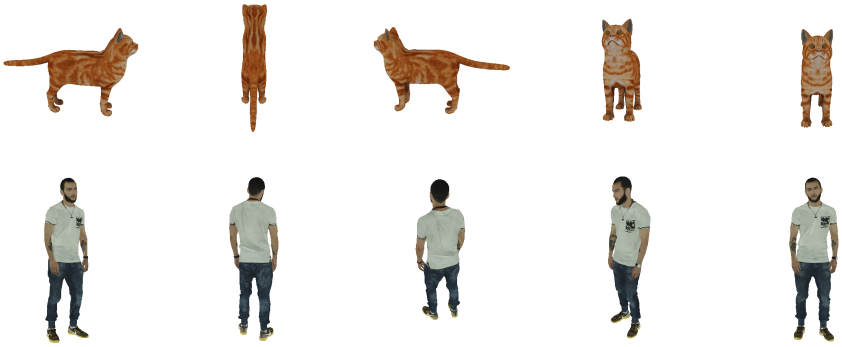
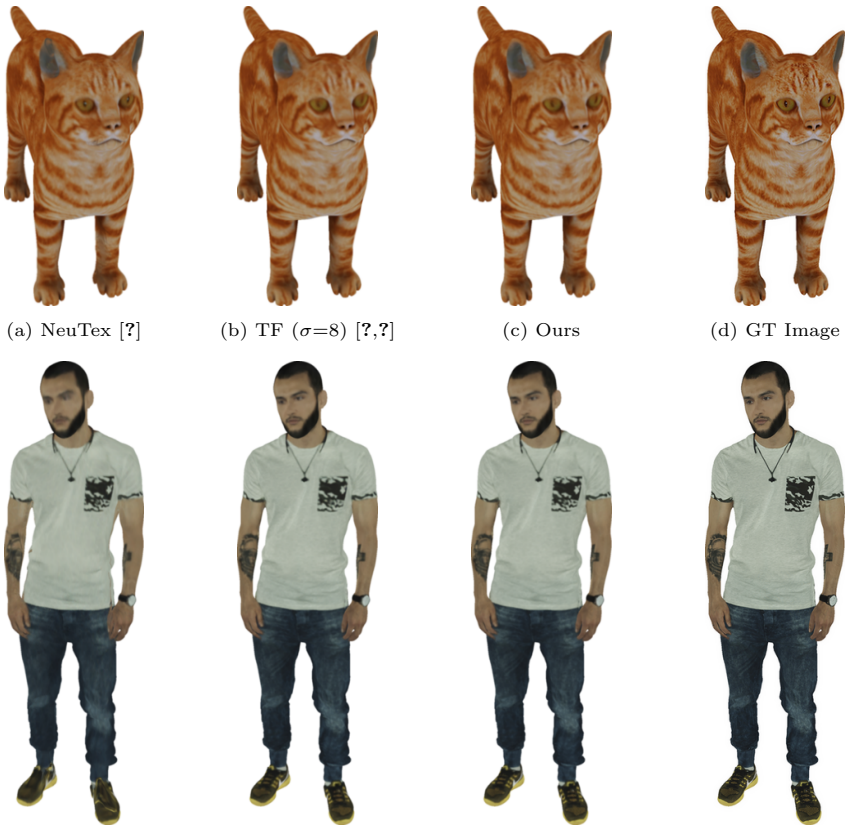


Fig. 10: Training views for the cat and human datasets with 5 views used in Sec. 5.1, Sec. 5.2, and Sec. 5.3.

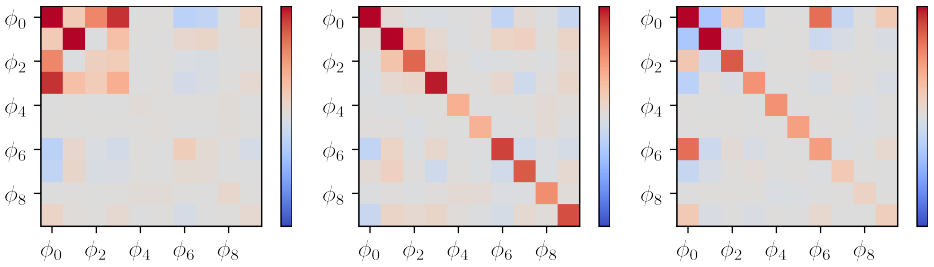
Fig. 11: Training views for the human high resolution dataset. Due to the large size of the 4096×4096 png images, we converted them to jpg and scaled them down to 1024×1024 for this figure.



(a) NeuTex [**?**] (b) TF (σ=8) [**?**,**?**] (c) Ours (d) GT Image

(e) NeuTex [**?**] (f) TF (σ=8) [**?**,**?**] (g) Ours (h) GT Image

Fig. 12: Further qualitative comparisons of unseen views from the texture reconstruction experiment (Sec. 5.1). All the methods from this figure use an embedding size of 1023. We want to point out that these renderings are not high-quality due to the low resolution of the training views.



(a) XY (b) RFF [?] (c) Ours

Fig. 13: Non-stationarity of NTKs on a general 2-manifold. We investigate the non-stationarity of the neural tangent kernels (NTKs) on a general 2-manifold, namely the cat shown in Fig. 3. Each small square in the image shows the coefficient c_{ij} when projecting the kernel onto the LBO basis: $c_{ij} = \int_{\mathcal{M}} \int_{\mathcal{M}} \phi_i(\mathbf{p}) k(\mathbf{p}, \mathbf{q}) \phi_j(\mathbf{q}) \, \mathrm{d}\mathbf{p} \, \mathrm{d}\mathbf{q}$. The integral is approximated numerically as the area-weighted sum over the vertices. A stationary kernel as defined in Eqn. 4 would have entries only along the main diagonal $c_{ij} = c_i \delta_{ij}$. The composed NTK is non-stationary for all features. We do not consider this a shortcoming of the proposed intrinsic neural fields because the NTK adapts to the intrinsic geometry of the underlying manifold as we show in Fig. 3.

References

1. Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., Katam, H.: Blenderproc. arXiv preprint arXiv:1911.01911 (2019)
2. Eisenberger, M., Lähner, Z., Cremers, D.: Smooth shells: Multi-scale shape registration with functional maps. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
3. Jacot, A., Hongler, C., Gabriel, F.: Neural tangent kernel: Convergence and generalization in neural networks. In: Conference on Neural Information Processing Systems (NeurIPS) (2018)
4. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision (ECCV) (2020)
5. Oechsle, M., Mescheder, L.M., Niemeyer, M., Strauss, T., Geiger, A.: Texture fields: Learning texture representations in function space. In: IEEE International Conference on Computer Vision (ICCV) (2019)
6. Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., Guibas, L.: Functional maps: A flexible representation of maps between shapes. ACM Transactions on Graphics (TOG) **31** (2012)
7. Sharp, N., Attaiki, S., Crane, K., Ovsjanikov, M.: Diffusionnet: Discretization agnostic learning on surfaces. ACM Transactions on Graphics (TOG) **XX** (20XX)
8. Sharp, N., Crane, K.: A laplacian for nonmanifold triangle meshes. Computer Graphics Forum **39** (2020)
9. Singh, A., Sha, J., Narayan, K.S., Achim, T., Abbeel, P.: Bigbird: A large-scale 3d database of object instances. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
10. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. In: Conference on Neural Information Processing Systems (NeurIPS) (2020)
11. Xiang, F., Xu, Z., Hasan, M., Hold-Geoffroy, Y., Sunkavalli, K., Su, H.: Neutex: Neural texture mapping for volumetric neural rendering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021)