

Intrinsic Neural Fields: Learning Functions on Manifolds

Lukas Koestler^{1*}, Daniel Grittner^{1*}, Michael Moeller², Daniel Cremers¹, and Zorah Lähler²

¹ Technical University of Munich
{lukas.koestler,daniel.grittner,cremers}@tum.de

² University of Siegen
{michael.moeller,zorah.laehner}@uni-siegen.de

* Equal contribution

Abstract. Neural fields have gained significant attention in the computer vision community due to their excellent performance in novel view synthesis, geometry reconstruction, and generative modeling. Some of their advantages are a sound theoretic foundation and an easy implementation in current deep learning frameworks. While neural fields have been applied to signals on manifolds, *e.g.*, for texture reconstruction, their representation has been limited to extrinsically embedding the shape into Euclidean space. The extrinsic embedding ignores known intrinsic manifold properties and is inflexible wrt. transfer of the learned function. To overcome these limitations, this work introduces intrinsic neural fields, a novel and versatile representation for neural fields on manifolds. Intrinsic neural fields combine the advantages of neural fields with the spectral properties of the Laplace-Beltrami operator. We show theoretically that intrinsic neural fields inherit many desirable properties of the extrinsic neural field framework but exhibit additional intrinsic qualities, like isometry invariance. In experiments, we show intrinsic neural fields can reconstruct high-fidelity textures from images with state-of-the-art quality and are robust to the discretization of the underlying manifold. We demonstrate the versatility of intrinsic neural fields by tackling various applications: texture transfer between deformed shapes & different shapes, texture reconstruction from real-world images with view dependence, and discretization-agnostic learning on meshes and point clouds.

1 Introduction

Neural fields have grown incredibly popular for novel view synthesis since the breakthrough work by Mildenhall et al. [29]. They showed that neural radiance fields together with differentiable volume rendering can be used to reconstruct scenes and often yield photorealistic renderings from novel viewpoints. This inspired work in related fields, *e.g.*, human shape modeling [34], shape and texture generation from text [28], and texture representation on shapes [32,2], where neural fields are able to generate a wide variety of functions with high fidelity.

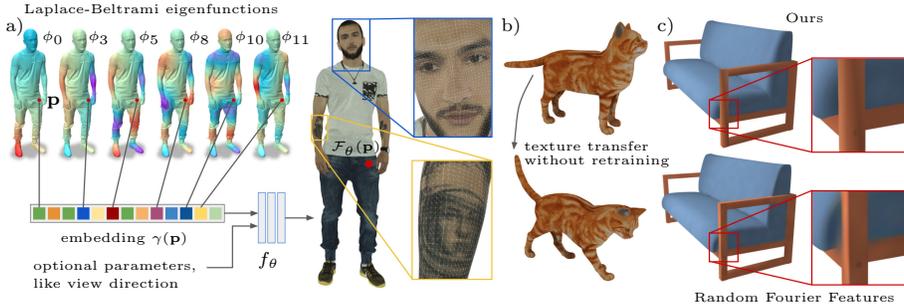


Fig. 1: (a) Overview of our method. We use the eigenfunctions ϕ_i of the Laplace-Beltrami operator (LBO) at each point as a point embedding $\gamma(\mathbf{p})$. This overcomes the spectral bias of the multilayer perceptron (MLP) f_θ , and hence the combined *intrinsic neural field* \mathcal{F}_θ can represent a high-frequency function on the surface. Notice that \mathbf{p} can be inside a triangle, and the function is clearly more detailed than the discretization (*insets*). (b) An intrinsic neural texture field trained on one shape (*top*) can be transferred to a new shape (*bottom*) without retraining. (c) Due to our intrinsic approach (LBO eigenfunctions) local geometry is maintained in close but separate parts, whereas an extrinsic approach (Random Fourier Features [50]) shows bleeding artifacts when trained with sparse supervision.

These methods use neural fields as functions from a point in Euclidean space to the quantity of interest. While this is valid for many applications, for others, the output actually lives on a general manifold. For example, texture mappings define a high-frequency color function on the surface of a 3D object. TextureFields [32] and Text2Mesh [28] solve this discrepancy by defining a mapping of each surface point to its Euclidean embedding and then learning the neural field there. Both show that they can achieve detail preservation above the discretization level, but the detour to Euclidean space has drawbacks. The Euclidean and geodesic distance between points can differ significantly. This is important on intricate shapes with fine geometric details that overlap because the local geometry prior is lost. Further, extrinsic representations cannot be used in the presence of surface deformations without retraining or applying heuristics.

Similar challenges have been solved in geometry processing by using purely intrinsic representations, most famously properties derived from the Laplace-Beltrami operator (LBO). Some of the main advantages of the LBO are its invariance under rigid and isometric deformations and reparametrization. We follow this direction by defining intrinsic neural fields on manifolds independent of the extrinsic Euclidean embedding and thus inherit the favorable properties of intrinsic representations. This is enabled by the fact that random Fourier features [50], an embedding technique that enabled the recent success of Euclidean neural fields, have an intrinsic analog based on the LBO. The result is a fully differentiable method that can learn high-frequency information on any 3D geometry representation that admits the computation of the LBO. A schematic overview of our method can be found in Figure 1. Our main theoretical and experimental **contributions** are:

- We introduce **intrinsic neural fields**, a novel and versatile representation for neural fields on manifolds. Intrinsic neural fields combine the advantages of neural fields with the spectral properties of the Laplace-Beltrami operator.
- We extend the **neural tangent kernel analysis** of [50] to the manifold setting. This yields a proof characterizing the stationarity of the kernel induced by intrinsic neural fields and insight into their spectral properties.
- We show that intrinsic neural fields can **reconstruct high-fidelity textures** from images with state-of-the-art quality.
- We demonstrate the versatility of intrinsic neural fields by tackling **various applications**: texture transfer between isometric and non-isometric shapes, texture reconstruction from real-world images with view dependence, and discretization-agnostic learning on meshes and point clouds.

We will release the full source code for all experiments and the associated data along with the final publication.

This work studies how a *neural field* can be defined on a manifold. Current approaches use the *extrinsic Euclidean embedding* and define the neural field on the manifold as a Euclidean neural field in the extrinsic embedding space – we describe this approach in [Sec. 3.1](#). In contrast, our approach uses the well-known Laplace-Beltrami Operator (LBO), which we briefly introduce in [Sec. 3.2](#). The final definition of *intrinsic neural fields* is given in [Sec. 4](#). The experimental results are presented in [Sec. 5](#).

2 Related Work

This work investigates neural fields for learning on manifolds, and we will only consider directly related work in this section. We point interested readers to the following overview articles: neural fields in visual computing [58], advances in neural rendering [51], and an introduction into spectral shape processing [25].

Neural Fields. While representing 3D objects and scenes with coordinate-based neural networks, or neural fields, has already been studied more than two decades ago [16,37,36], the topic has gained increased interest following the breakthrough work by Mildenhall et al. [29]. They show that a Neural Radiance Field (NeRF) often yields photorealistic renderings from novel viewpoints. One key technique underlying this success is positional encoding, which transforms the three-dimensional input coordinates into a higher dimensional space using sines and cosines with varying frequencies. This encoding overcomes the low-frequency bias of neural networks [38,3] and thus enables high-fidelity reconstructions. The aforementioned phenomenon is analyzed using the neural tangent kernel [20] by Tancik et al. [50], and our analysis extends theirs from Euclidean space to manifolds. Simultaneously to Tancik et al., Sitzmann et al. [47] use periodic activation functions for neural scene representation, which is similar to the above-mentioned positional encoding [4]. Additionally, many other works [60,41,40,27,22,19,63,54,26] offer insights into neural fields and their embedding functions. However, none of these works considers neural fields on manifolds.

Neural Fields on Manifolds. Prior works [32,11,2,57,34,30,59,28,18] use neural fields to represent a wide variety of quantities on manifolds. Oechsle et al. [32] use the extrinsic embedding of the manifold to learn textures as multilayer perceptrons. Their Texture Fields serve as an important baseline for this work. NeuTex by Xiang et al. [57] combines neural, volumetric scene representations with a 2D texture network to facilitate interpretable and editable texture learning. To enable this disentanglement, their method uses mapping networks from the 3D space of the object to the 2D space of the texture and back. We compare with an adapted version of their method that utilizes the known geometry of the object. Baatz et al. [2] introduce NeRF-*Tex*, a combination of neural radiance fields (NeRFs) and classical texture maps. Their method uses multiple small-scale NeRFs to cover the surface of a shape and represent mesoscale structures, such as fur, fabric, and grass. Because their method focuses on mesoscale structures and artistic editing, we believe that extending the current work to their setting is an interesting direction for future research.

Additionally, neural fields have been used to represent quantities other than texture on manifolds. Palafox et al. [34] define a neural deformation field that maps points on a canonical shape to their location after the shape is deformed. This model is applied to generate neural parametric models, which can be used similarly to traditional parametric models like SMPL [24]. Yifan et al. [59] decompose a neural signed distance function (SDF) into a coarse SDF and a high-frequency implicit displacement field. Morreale et al. [30] define neural surface maps, which can be used to define surface-to-surface correspondences among other applications. Text2Mesh [28] uses a coarse mesh and a textual description to generate a detailed mesh and associated texture as neural fields.

Intrinsic Geometry Processing. Intrinsic properties are a popular tool in geometry processing, especially in the analysis of deformable objects. Two of the most basic intrinsic features are Gauss curvature and intrinsic point descriptors based on the Laplace-Beltrami operator (LBO). They have been heavily used since the introduction of the global point signature [42] and refined since then [49,1]. Intrinsic properties are not derived from a manifold’s embedding into its embedding space but instead arise from the pairwise geodesic distance on the surface. These are directly related to natural kernel functions on manifolds, *e.g.*, shown by the efficient approximation of the geodesic distance from the heat kernel [12]. Kernel functions as a measure of similarity between points are very popular in geometry processing. They have been used in various applications, *e.g.*, in shape matching [53,9,23], parallel transport [45], and robustness wrt. discretization [52,43]. Manifold kernels naturally consider the local and global geometry [5], and our approach follows in this direction by showing a natural extension of neural fields on manifolds.

3 Background

Differential geometry offers two viewpoints onto manifolds: intrinsic and extrinsic. The extrinsic viewpoint studies the manifold \mathcal{M} through its *Euclidean*

embedding where each point $\mathbf{p} \in \mathcal{M}$ is associated with its corresponding point in Euclidean space. In contrast, the intrinsic viewpoint considers only properties of points independent of the extrinsic embedding, such as, the geodesic distance between a point pair. Both can have advantages depending on the method and application. An intrinsic viewpoint is by design invariant against certain deformations in the Euclidean embedding, like rigid transformations but also pose variations that are hard to characterize in the extrinsic view.

3.1 Neural Fields for Euclidean Space

A Euclidean neural field $\mathcal{F}_\theta^E : \mathbb{R}^m \rightarrow \mathbb{R}^o$ is a neural network that maps points in Euclidean space to vectors and is parametrized by weights $\theta \in \mathbb{R}^p$. The network is commonly chosen to be a multilayer perceptron (MLP). Let $\mathcal{M} \subset \mathbb{R}^m$ be a manifold with a Euclidean embedding into \mathbb{R}^m . Naturally, the restriction of \mathcal{F}_θ^E to \mathcal{M} leads to a neural field on a manifold: $\mathcal{F}_\theta : \mathcal{M} \rightarrow \mathbb{R}^o$, $\mathcal{F}_\theta(x) = \mathcal{F}_\theta^E(x)$.

Natural signals, such as images and scenes, are usually quite complex and contain high-frequency variations. Due to spectral bias, standard neural fields fail to learn high-frequency functions from low dimensional data [50,47] and generate blurry reconstructions. With the help of the neural tangent kernel, it was proven that the composition $\mathcal{F}_\theta^E \circ \gamma$ of a higher dimensional Euclidean neural field and a random Fourier feature (RFF) encoding γ helps to overcome the spectral bias and, consequently, enables the neural field to better represent high-frequency signals. The RFF encoding $\gamma : \mathbb{R}^m \rightarrow \mathbb{R}^d$ with $d \gg m$ is defined as

$$\gamma(\mathbf{x}) = [a_1 \cos(\mathbf{b}_1^\top \mathbf{x}), a_1 \sin(\mathbf{b}_1^\top \mathbf{x}), \dots, a_{d/2} \cos(\mathbf{b}_{d/2}^\top \mathbf{x}), a_{d/2} \sin(\mathbf{b}_{d/2}^\top \mathbf{x})], \quad (1)$$

where the coefficients $\mathbf{b}_i \in \mathbb{R}^m$ are randomly drawn from the multivariate normal distribution $\mathcal{N}(\mathbf{0}, (2\pi\sigma)^2 \mathbf{I})$. The factors a_i are often set to one for all i and $\sigma > 0$ is a hyperparameter that offers a trade-off between reconstruction fidelity and overfitting of the training data.

3.2 The Laplace-Beltrami Operator

In the following, we briefly introduce the Laplace-Beltrami operator (LBO) and refer the interested reader to [42] for more details. The LBO $\Delta_{\mathcal{M}}$ is the generalization of the Euclidean Laplace operator on general closed compact manifolds. Its eigenfunctions $\phi_i : \mathcal{M} \rightarrow \mathbb{R}$ and eigenvalues $\lambda_i \in \mathbb{R}$ are the non-trivial solutions of the equation $\Delta_{\mathcal{M}} \phi_i = \lambda_i \phi_i$. The eigenvalues are non-negative and induce a natural ordering which we will use for the rest of the paper. The eigenfunctions are orthonormal to each other, build an optimal basis for the space of square-integrable functions [35], and are frequency ordered allowing a low-pass filtering by projecting onto the first k eigenfunctions. Hence, a function $f : \mathcal{M} \rightarrow \mathbb{R} \in L^2(\mathcal{M})$ can be expanded in this basis:

$$f = \sum_{i=0}^{\infty} c_i \phi_i = \sum_{i=0}^{\infty} \langle f, \phi_i \rangle \phi_i \approx \sum_{i=0}^k \langle f, \phi_i \rangle \phi_i, \quad (2)$$

where the quality of the last \approx depends on the amount of high-frequency information in f . The projection onto the LBO basis is similar to the Fourier transform, allowing the same operations, and thus we use the LBO basis as the replacement for Fourier features. In fact, if $[0, 1]^2$ is considered as a manifold, its LBO eigenfunctions with different boundary conditions are exactly combinations of sines and cosines. Furthermore, the eigenfunctions of the LBO are identical up to sign ambiguity for isometric shapes since the LBO is entirely intrinsic.

4 Intrinsic Neural Fields

We introduce *Intrinsic Neural Fields* based on the eigenfunctions of the Laplace-Beltrami operator (LBO) which can represent detailed surface information, like texture, directly on the manifold. In the presence of prior geometric surface information, it is more efficient than using the extrinsic Euclidean embedding space which is often mainly empty. Additionally, this representation is naturally translation and rotation invariant, surface representation invariant, as well as invariant to isometric deformations.

Definition 1 (Intrinsic Neural Field). *Let $\mathcal{M} \subset \mathbb{R}^m$ be a closed, compact manifold and ϕ_1, \dots, ϕ_d be the first d Laplace-Beltrami eigenfunctions of \mathcal{M} . We define an **intrinsic neural field** $\mathcal{F}_\theta : \mathcal{M} \rightarrow \mathbb{R}^o$ as*

$$\mathcal{F}_\theta(\mathbf{p}) = (f_\theta \circ \gamma)(\mathbf{p}) = f_\theta(a_1\phi_1(\mathbf{p}), \dots, a_d\phi_d(\mathbf{p})). \quad (3)$$

where $\gamma : \mathcal{M} \rightarrow \mathbb{R}^d, \gamma(\mathbf{p}) = (a_1\phi_1(\mathbf{p}), \dots, a_d\phi_d(\mathbf{p}))$, with $a_i \geq 0$ and $\lambda_i = \lambda_j \Rightarrow a_i = a_j$, is our embedding function and $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^o$ represents a neural network with weights $\theta \in \mathbb{R}^p$.

Within this work, we will use $a_i = 1$, which has proven sufficient in praxis, and multilayer perceptrons (MLPs) for f_θ , as this architectural choice is common for Euclidean neural fields [58]. A detailed description of the architecture can be found in Fig. 8. It is possible to choose different embedding functions γ but we choose the LBO eigenfunctions as they have nice theoretical properties (see Section 4.1) and are directly related to Fourier features.

In Fig. 2, we apply intrinsic neural fields to the task of signal reconstruction on a 1D manifold to give an intuition about how it works and what its advantages are. The results show that the neural tangent kernel (NTK) for intrinsic neural fields exhibits favorable properties, which we prove in Sec. 4.1. We show that we can represent high-frequency signals on manifold surfaces that go far beyond the discretization level. In Sec. 5, we apply the proposed intrinsic neural fields to a variety of tasks including texture reconstruction from images, texture transfer between shapes without retraining, and view-dependent appearance modeling.

4.1 Theory

In this section, we will prove that the embedding function γ proposed in Definition 1 generalizes the stationarity result of [50] to certain manifolds. Stationarity is a desirable property if the kernel is used for interpolation, for example,

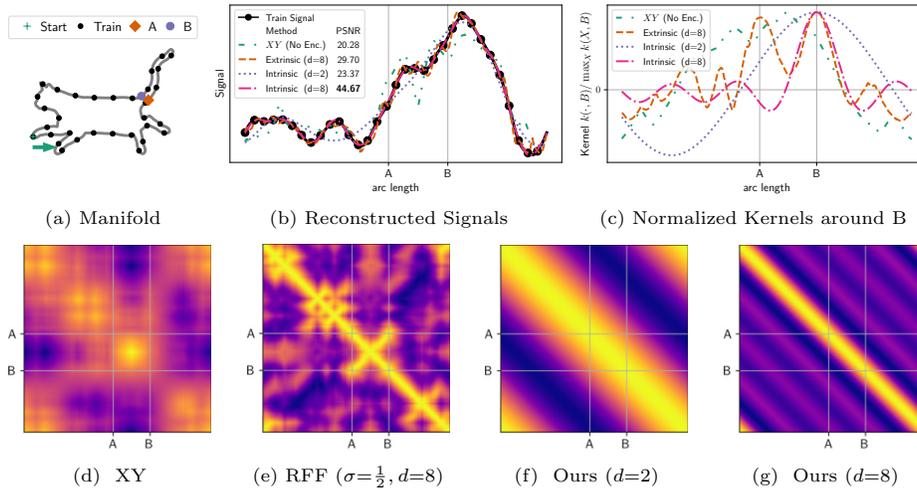


Fig. 2: Signal reconstruction. (2b) The target is sampled at 32 points and MLPs with three layers, 1024 channels, and different embeddings are trained using L^2 loss. The intrinsic neural field with $d=8$ eigenfunctions performs best. Using only two eigenfunctions leads to oversmoothing. The reconstruction with the extrinsic embedding and random Fourier features (RFF) [50] can capture the high-frequency details, but introduces artifacts when the Euclidean distance is not a good approximation of the geodesic distance, for example, at points A & B. (2d-2g) The second row of subfigures shows the pairwise neural tangent kernel (NTK) [20,31] between all points on the manifold. (2d) The NTK using the extrinsic Euclidean embedding is not maximal along the diagonal. (2e) For the NTK with RFF embedding the maximum is at the diagonal because each point’s influence is maximal onto itself. However, it has many spurious correlations between points that are close in Euclidean space but not along the manifold, for example, around B. (2f,2g) The NTK with our intrinsic embedding is localized correctly and is stationary (c.f. Thm. 1), which makes it most suitable for interpolation.

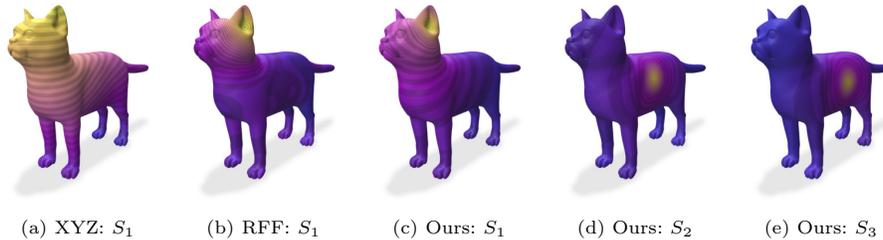


Fig. 3: Neural tangent kernels (NTKs) [20,31] with different embedding functions. The source S_1 lies directly inside the ear of the cat. (3a) The NTK using the extrinsic Euclidean embedding is not maximal at the source. (3b) The NTK using random Fourier features (RFF) [50] is localized correctly, but shows wrong behavior on the cat’s body. (3c) The NTK with our intrinsic embedding is localized correctly and adapts to the local and global geometry. (3d,3e) Additionally, the NTK with our intrinsic embedding is nearly shift-invariant, if the local geometry is approximately Euclidean: When the source is shifted from S_2 to S_3 the kernel is approximately shifted as well.

in novel view synthesis [50, App. C]. Fourier features induce a stationary (shift-invariant) neural tangent kernel (NTK). Namely, the composed NTK for two points in Euclidean space $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ is given by $k_{\text{NTK}}(\mathbf{x}, \mathbf{y}) = (\text{h}_{\text{NTK}} \circ \text{h}_\gamma)(\mathbf{x} - \mathbf{y})$ where $\text{h}_{\text{NTK}} : \mathbb{R} \rightarrow \mathbb{R}$ is a scalar function related to the NTK of the MLP and $\text{h}_\gamma : \mathbb{R}^m \rightarrow \mathbb{R}$ is a scalar function related to the Fourier feature embedding [50, Eqn. 7,8]. Extending this result to cover inputs $\mathbf{p}, \mathbf{q} \in \mathcal{M}$ on a manifold is challenging because the point difference $\mathbf{p} - \mathbf{q}$ and, therefore, the concept of stationary is not defined intrinsically.

Stationarity on Manifolds. While one could use the Euclidean embedding of the manifold to define the difference $\mathbf{p} - \mathbf{q}$, this would ignore the local connectivity and can change under extrinsic deformations. Instead, we make use of an equivalent definition from Bochner’s theorem which implies that for Euclidean space any continuous, *stationary* kernel is the Fourier transform of a non-negative measure [39, Thm. 1]. This definition can be directly used on manifolds, and we define a kernel $k : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}$ to be **stationary** if it can be written as

$$k(\mathbf{p}, \mathbf{q}) = \sum_i \hat{k}(\lambda_i) \phi_i(\mathbf{p}) \phi_i(\mathbf{q}), \quad \hat{k}(\lambda_i) \geq 0 \quad \forall i, \quad (4)$$

where the function $\hat{k} : \mathbb{R} \rightarrow \mathbb{R}_0^+$ is akin to the Fourier transform. This implies that $\hat{k}(\lambda_i)$ and $\hat{k}(\lambda_j)$ for identical eigenvalues $\lambda_i = \lambda_j$ must be identical.

First, we want to point out that for inputs with $\|\mathbf{x}\| = \|\mathbf{y}\| = r$ the result of $k_{\text{NTK}}(\mathbf{x}, \mathbf{y}) = \text{h}_{\text{NTK}}(\langle \mathbf{x}, \mathbf{y} \rangle)$ shown by [20] for $r = 1$ and used in [50] still holds. This slight extension is given as **Lem. 1**. It is a prerequisite for the following theorem which requires the same setting as used in [20].

Theorem 1. *Let \mathcal{M} be \mathbb{S}^n or a closed 1-manifold. Let $(\lambda_i, \phi_i)_{i=1, \dots, d}$ be the positive, non-decreasing eigenvalues with associated eigenfunctions of the Laplace-Beltrami operator on \mathcal{M} . Let $a_i \geq 0$ be coefficients s.t. $\lambda_i = \lambda_j \Rightarrow a_i = a_j$, which define the embedding function $\gamma : \mathcal{M} \rightarrow \mathbb{R}^d$ with $\gamma(\mathbf{p}) = (a_1 \phi_1(\mathbf{p}), \dots, a_d \phi_d(\mathbf{p}))$. Then, the composed neural tangent kernel $k_{\text{NTK}} : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}$ of an MLP with the embedding γ is stationary as defined in **Eq. 4**.*

Proof. Let $\mathcal{M} = \mathbb{S}^n$ and let \mathbf{H}_l^n be the space of degree l spherical harmonics on \mathbb{S}^n . Let $Y_{lm} \in \mathbf{H}_l^n$ be the m -th real spherical harmonic of degree l with $m = 1, \dots, \dim \mathbf{H}_l^n$. Notice that the spherical harmonics are the eigenfunctions of the LBO. We will use j to linearly index the spherical harmonics and $l(j)$ for the degree. Spherical harmonics of the same degree have the same eigenvalues, thus we use $c_{l(j)} = a_j = a_i$ for $\lambda_i = \lambda_j$ to denote the equal coefficients for same degree harmonics. First, the norm of the embedding function is constant:

$$\|\gamma(\mathbf{q})\|^2 = \sum_j c_{l(j)}^2 \phi_j^2(\mathbf{q}) = \sum_l c_l^2 \sum_{m=1}^{\dim \mathbf{H}_l^n} Y_{lm}^2(\mathbf{q}) \stackrel{(a)}{=} \sum_l c_l^2 Z_l(\mathbf{q}, \mathbf{q}) \stackrel{(b)}{=} \text{const.} \quad (5)$$

Here, $Z_l(\mathbf{q}, \mathbf{q})$ is the degree l zonal harmonic and (a), (b) are properties of zonal harmonics [13, Lem. 1.2.3, Lem. 1.2.7]. Due to **Eq. 5** and **Lem. 1** $k_{\text{NTK}}(\gamma(\mathbf{p}), \gamma(\mathbf{q})) =$

$\mathbf{h}_{\text{NTK}}(\langle \gamma(\mathbf{p}), \gamma(\mathbf{q}) \rangle) \forall \mathbf{p}, \mathbf{q} \in \mathcal{M}$ holds. We can rewrite the scalar product as follows

$$\langle \gamma(\mathbf{p}), \gamma(\mathbf{q}) \rangle = \sum_j c_{l(j)}^2 \phi_j(\mathbf{p}) \phi_j(\mathbf{q}) = \sum_l c_l^2 \sum_{m=1}^{\dim \mathbf{H}_l^n} Y_{lm}(\mathbf{p}) Y_{lm}(\mathbf{q}) \quad (6)$$

$$\stackrel{(c)}{=} \sum_l c_l^2 Z_{\mathbf{p}}^l(\mathbf{q}) \stackrel{(d)}{=} \sum_l c_l^2 (1 + l/\alpha) C_l^\alpha(\langle \mathbf{p}, \mathbf{q} \rangle), \quad (7)$$

where $C_l^\alpha : [-1, 1] \rightarrow \mathbb{R}$ are the Gegenbauer polynomials which are orthogonal on $[-1, 1]$ for the weighting function $w_\alpha(z) = (1 - z^2)^{\alpha-1/2}$ with $\alpha = (n - 1)/2$ [13, B.2]. Equality (c) holds again due to [13, Lem. 1.2.3]. Equality (d) holds due to a property of Gegenbauer polynomials [13, Thm. 1.2.6], here $\langle \mathbf{p}, \mathbf{q} \rangle$ denotes the extrinsic Euclidean inner product. For the composed NTK we obtain

$$\mathbf{k}_{\text{NTK}}(\gamma(\mathbf{p}), \gamma(\mathbf{q})) = \mathbf{h}_{\text{NTK}} \left(\sum_l c_l^2 (1 + l/\alpha) C_l^\alpha(\langle \mathbf{p}, \mathbf{q} \rangle) \right). \quad (8)$$

We see that $\mathbf{k}_{\text{NTK}}(\gamma(\mathbf{p}), \gamma(\mathbf{q}))$ is a function depending only on $\langle \mathbf{p}, \mathbf{q} \rangle$. Because the Gegenbauer polynomials are orthogonal on $[-1, 1]$, this function can be expanded with coefficients $\hat{c}_l \in \mathbb{R}$, which yields

$$\mathbf{k}_{\text{NTK}}(\gamma(\mathbf{p}), \gamma(\mathbf{q})) = \sum_l \hat{c}_l (1 + l/\alpha) C_l^\alpha(\langle \mathbf{p}, \mathbf{q} \rangle) = \sum_l \hat{c}_l Z^l(\mathbf{p}, \mathbf{q}) \quad (9)$$

$$= \sum_l \hat{c}_l \sum_{m=1}^{\dim \mathbf{H}_l^n} Y_{lm}(\mathbf{p}) Y_{lm}(\mathbf{q}) = \sum_j \hat{c}_{l(j)} \phi_j(\mathbf{p}) \phi_j(\mathbf{q}). \quad (10)$$

The coefficients $\hat{c}_{l(j)}$ are non-negative as a consequence of the positive definiteness of the NTK [20, Prop. 2] and a classic result by Schoenberg [13, Thm. 14.3.3]. This shows that $\mathbf{k}_{\text{NTK}}(\gamma(\mathbf{p}), \gamma(\mathbf{q}))$ is stationary as defined in Equation 4. \square

The adapted proof for 1-manifolds can be found in Sec. D.1. A qualitative example of the stationary kernels can be seen in Fig. 2. The theorem does not hold for general manifolds but we do not consider this a shortcoming. The composed kernel adapts to the intrinsic geometry of the underlying manifold and is approximately shift-invariant between points that share a similar local neighborhood, see Fig. 3. Hence, the proposed method naturally and unambiguously integrates the geometry of the manifold based on an intrinsic representation.

5 Experiments

Due to the large number of experiments presented within this section, we refer to Sec. B for all experimental details and hyperparameter settings as well as further results. To facilitate fair comparisons, all methods use the same hyperparameters like learning rate, optimizer, number of training epochs, or MLP architecture except when noted otherwise. For baselines using random Fourier features (RFF), we follow [50] and tune the standard deviation σ (c.f. Eq. 1) of the random frequency matrix to obtain optimal results.

5.1 Texture Reconstruction from Images

To investigate the representation power of the proposed intrinsic neural fields, we consider the task of texture reconstruction from posed images as proposed by Oechsle et al. [32] in Tab. 1 and Fig. 4. The input to our algorithms is a set of five 512×512 images with their camera poses and the triangle mesh of the shape. After fitting the intrinsic neural field to the data, we render images from 200 novel viewpoints and compare them to ground-truth images for evaluation.

For each pixel, we perform ray mesh intersection between the ray through the pixel and the mesh. The eigenfunctions of the Laplace-Beltrami operator are defined only on vertices of the mesh [44]. Within triangles, we use barycentric interpolation. We employ the mean L^1 loss across a batch of rays and the RGB color channels. The eigenfunction computation and ray-mesh intersection are performed once at the start of training. Hence, our training speed is similar to the baseline method that uses random Fourier features. Training takes approx. one hour on an Nvidia Titan X with 12 GB memory.

Comparison with State of the Art Methods. We compare against Texture Fields [32] enhanced with random Fourier features (RFF) [50]. Additionally, we compare against NeuTex [57], which uses a network to map a shape to the sphere and represents the texture on this sphere. We adapt NeuTex s.t. it takes advantage of the given geometry, see Sec. B.1. Tab. 1 and Fig. 4 show that intrinsic neural fields can reconstruct texture with state-of-the-art quality. This is also true if the number of training epochs is decreased from 1000 to 200.

Ablation Study. We investigate the effect of different hyperparameters on the quality of the intrinsic neural texture field. The results in Tab. 2 show that the number of eigenfunctions is more important than the size of the MLP, which is promising for real-time applications. A model using only 64 eigenfunctions and $17k$ parameters³ still achieves a PSNR of 29.20 for the cat showing that intrinsic neural fields can be a promising approach for compressing manifold data.

5.2 Discretization-agnostic Intrinsic Neural Fields

For real-world applications, it is desirable that intrinsic neural fields can be trained for different discretizations of the same manifold. First, the training process of the intrinsic neural field should be robust to the sampling in the discretization. Second, it would be beneficial if an intrinsic neural field trained on one discretization could be transferred to another, which we show in Sec. 5.3. To quantify the discretization dependence of intrinsic neural fields, we follow the procedure proposed by Sharp et al. [43, Sec. 5.4] and discretize the meshes used in Sec. 5.1. The qualitative results in Fig. 5 and the quantitative results in Tab. 3 show that intrinsic neural fields work across various discretizations. Furthermore, Fig. 6 shows that transferring pre-trained intrinsic neural fields across discretizations is possible with minimal loss in visual quality.

³ For reference: a 80×80 3-channel color texture image has over $17k$ pixel values.

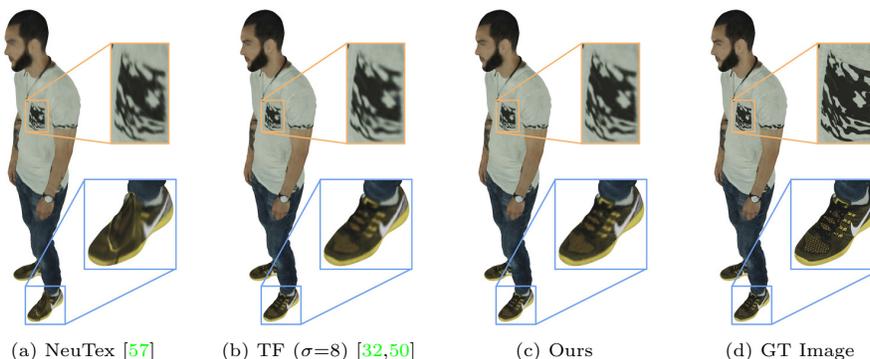


Fig. 4: Texture reconstruction from images. (4a) NeuTex uses a network to map from the shape to the sphere and represents the texture on the sphere, which yields distortions around the shoe. (4b) Texture Fields (TF) [32] with random Fourier Features (RFF) [50] learns the texture well and only around the breast pocket our method shows slightly better results. (4c) Intrinsic neural fields can reconstruct texture from images with state-of-the-art quality, which we show quantitatively in Tab. 1.

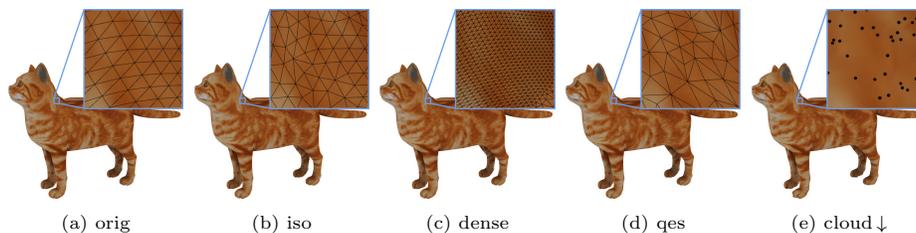


Fig. 5: Discretization-agnostic intrinsic neural fields. Our method produces identical results for a wide variety of triangular meshings and even point cloud data. For the point cloud, we use local triangulations [44, Sec. 5.7] for ray-mesh intersection. Pre-trained intrinsic neural fields can be transferred across discretizations as shown in Fig. 6.

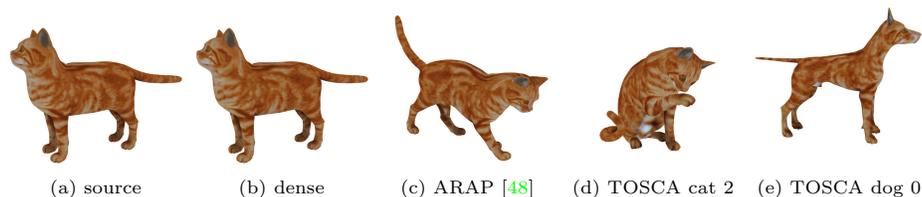


Fig. 6: Intrinsic neural field transfer. (6a) The pre-trained intrinsic neural texture field from the source mesh is transferred to the target shapes using functional maps [33,15]. (6b,6c) The transfer across discretization (c.f. Fig. 5) and deformation gives nearly perfect visual quality. (6d,6e) As a proof of concept, we show artistic transfer to a different cat shape and a dog shape from the TOSCA dataset [8]. Both transfers work well but the transfer to the dog shows small visual artifacts in the snout area due to locally different geometry. Overall, the experiment shows the advantage of the intrinsic formulation which naturally incorporates field transfer through functional maps.

Table 1: Texture reconstruction from images. Our intrinsic neural fields show state-of-the-art performance (*first row block*), which also holds for fewer training epochs (*Ep. ↓*, *second row block*). For a fair comparison, we improve the original Texture Fields by employing the same MLP architecture as our model and by additionally using random Fourier features (*TF+RFF*). NeuTex already has more parameters than our model and we increase the embedding size (*Em. ↑*). We adapt NeuTex s.t. it takes advantage of the given geometry, which we detail in [Sec. B.1](#). The methods are evaluated on novel views using the PSNR, DSSIM [55], and LPIPS [62]. DSSIM and LPIPS are scaled by 100. For each row block, the best number is in bold font. The intrinsic representation shows better results than the extrinsic representation (*TF+RFF*) as well as when mapping to a sphere and representing the texture there (*NeuTex*). For qualitative results, see [Fig. 4](#).

	Em.	Ep.	cat			human				
			PSNR ↑	DSSIM ↓	LPIPS ↓	PSNR ↑	DSSIM ↓	LPIPS ↓		
NeuTex [57]	63	1000	31.60	0.242	0.504	29.49	0.329	0.715		
NeuTex Em. ↑	1023	1000	31.96	0.212	0.266	29.22	0.306	0.669		
TF+RFF ($\sigma=4$) [32,50]	1023	1000	33.86	0.125	0.444	32.04	0.130	0.420		
TF+RFF ($\sigma=16$)	1023	1000	34.19	0.105	0.167	31.53	0.193	0.414		
TF+RFF ($\sigma=8$)	1023	1000	34.39	0.097	0.205	32.26	0.129	0.336		
Intrinsic (Ours)	1023	1000	34.82	0.095	0.153	32.48	0.121	0.306		
NeuTex		Ep. ↓	1023	200	30.96	0.290	0.355	28.02	0.418	0.900
TF+RFF ($\sigma=8$)		Ep. ↓	1023	200	34.07	0.116	0.346	31.85	0.142	0.427
Intrinsic (Ours)		Ep. ↓	1023	200	34.79	0.100	0.196	32.37	0.126	0.346

5.3 Intrinsic Neural Field Transfer

One advantage of the Laplace-Beltrami operator is its invariance under isometries which allows for transferring a pre-trained intrinsic neural field from one manifold to another. However, this theoretic invariance does not always perfectly hold in practice, for example, due to discretization artifacts as discussed by Kovnatsky et al. [21]. Hence, we employ functional maps [33] to transfer the eigenfunctions of the source to the target shape, computed with the method proposed by Eisenberger et al. [15]. [Fig. 6](#) shows that intrinsic neural fields can be transferred between shapes. Specifically, the transfer is possible between different discretizations and deformations [48] of the same shape. As a proof of concept, we also show artistic transfer, which yields satisfying results for shapes from the same category, but small visual artifacts for shapes from different categories. It is, of course, possible to generate similar results with extrinsic fields by calculating a point-to-point correspondence and mapping the coordinate values. However, functional maps are naturally low-dimensional, continuous, and differentiable. This makes them a beneficial choice in many applications, especially related to learning.

5.4 Real-world Data & View Dependence

We validate the effectiveness of intrinsic neural fields under real-world settings on the BigBIRD dataset [46]. The dataset provides posed images and reconstructed

Table 2: Ablation study based on the texture reconstruction experiment (c.f. Sec. 5.1). The number of eigenfunctions is more important than the size of the MLP which is promising for real-time applications. A model using only 64 eigenfunctions and only 17k parameters still achieves a PSNR of 29.20 for the cat, which shows that intrinsic neural fields can be a promising approach for compressing manifold data.

	#Params	# ϕ	cat			human		
			PSNR \uparrow	DSSIM \downarrow	LPIPS \downarrow	PSNR \uparrow	DSSIM \downarrow	LPIPS \downarrow
Full model	329k	1023	34.82	0.095	0.153	32.48	0.121	0.306
Smaller MLP	140k	1023	34.57	0.108	0.205	32.20	0.134	0.379
Fewer eigenfunctions	83k	64	31.18	0.284	0.927	28.95	0.312	1.090
Smaller MLP & fewer efs	17k	64	29.20	0.473	1.428	26.72	0.493	1.766
Just 4 eigenfunctions	68k	4	22.84	1.367	3.299	20.60	1.033	2.756

Table 3: Discretization-agnostic intrinsic neural fields. We employ the procedure proposed by Sharp et al. [43, Sec. 5.4] to generate different discretizations of the original meshes (*orig*): uniform isotropic remeshing (*iso*), densification around random vertices (*dense*), refinement and subsequent quadric error simplification [17] (*qes*), and point clouds sampled from the surfaces with more points than vertices (*cloud* \uparrow) and with fewer points (*cloud* \downarrow). The discretizations are then used for texture reconstruction as in Sec. 5.1. For the point clouds, we use local triangulations [44, Sec. 5.7] for ray-mesh intersection. This table and the qualitative results in Fig. 5 show that intrinsic neural fields can be trained for a wide variety of discretizations. Furthermore, pre-trained intrinsic neural fields can be transferred across discretizations as shown in Fig. 6.

Method	cat						human					
	orig	iso	dense	qes	cloud \uparrow	cloud \downarrow	orig	iso	dense	qes	cloud \uparrow	cloud \downarrow
PSNR \uparrow	34.82	34.85	34.74	35.07	34.91	33.17	32.48	32.63	32.57	32.49	32.45	31.99
DSSIM \downarrow	0.095	0.093	0.096	0.096	0.096	0.130	0.121	0.117	0.120	0.121	0.123	0.135
LPIPS \downarrow	0.153	0.152	0.159	0.147	0.152	0.220	0.306	0.300	0.301	0.297	0.307	0.323

meshes, and we apply a similar pipeline as in Sec. 5.1. However, the objects under consideration are not perfectly Lambertian, and thus, view dependence must be considered. This is achieved by using the viewing direction as an additional input to the network, as done in [29]. At first glance, using the viewing direction in its extrinsic representation is in contrast to our intrinsic definition of neural fields. However, view dependence arises from the extrinsic scene of the object, such as the lighting, which cannot be represented purely intrinsically. Decomposing the scene into the intrinsic properties, like the BRDF, of the object and the influence of the environment, like light sources, is an interesting future application for intrinsic neural fields. Such decomposition has recently been studied in the context of neural radiance fields [10,61,6,56,7]. Intrinsic neural fields can reconstruct high-quality textures from real-world data with imprecise calibration and imprecise meshes, as shown in Fig. 7.



Fig. 7: Texture reconstruction from real-world data. (7b,7e) Intrinsic neural fields can reconstruct high quality textures from the real-world BigBIRD dataset [46] with imprecise calibration and imprecise meshes. (7a,7d) The baseline texture mapped meshes provided in the dataset show notable seams due to the non-Lambertian material, which are not present in our reconstruction that utilizes view dependence as proposed by [29].

6 Conclusion

Discussion. The proposed intrinsic formulation of neural fields outperforms the extrinsic formulation in the presented experiments. However, if the data is very densely sampled from the manifold, and the kernel is thus locally limited, the extrinsic method can overcome many of its weaknesses shown before. In practice, dense sampling often leads to an increase in runtime of further processing steps, and therefore, we consider our intrinsic approach still to be superior. Further, we provided the proof for a stationary NTK on n-spheres. Our good results and intuition imply that for general manifolds, it is advantageous how the NTK takes local geometry into account. We believe this opens up an interesting direction for further theoretical analysis.

Conclusion. We present intrinsic neural fields, an elegant and direct generalization of neural fields for manifolds. Intrinsic neural fields can represent high-frequency functions on the manifold surface independent of discretization by making use of the Laplace-Beltrami eigenfunctions. As a result, they also inherit beneficial properties of the LBO, like isometry invariance, a natural frequency filter, and are directly compatible with the popular functional map framework. We introduce a new definition for stationary kernels on manifolds, and our theoretic analysis shows that the derived neural tangent kernel is stationary under specific conditions.

We conduct experiments to investigate the capabilities of our framework on the application of texture reconstruction from a limited number of views. Our results show that intrinsic neural fields can represent high-frequency functions on the surface independent of sampling density and surface discretization. Furthermore, the learned functions can be transferred to new examples using functional maps without any retraining, and view-dependent changes can be incorporated. Intrinsic neural fields outperform competing methods in all settings. Additionally, they add flexibility, especially in settings with deformable objects due to the intrinsic nature of our approach.

Acknowledgements

We express our appreciation to our colleagues who have supported us. Specifically we thank Simon Klenk, Tarun Yenamandra, Björn Häfner, and Dominik Muhle for proofreading and helpful discussions. We want to thank the research community at large and contributors to open source projects for openly and freely sharing their knowledge and work.

References

1. Aubry, M., Schlickewei, U., Cremers, D.: The wave kernel signature: A quantum mechanical approach to shape analysis. In: IEEE International Conference on Computer Vision (ICCV) (2011)
2. Baatz, H., Granskog, J., Papas, M., Rousselle, F., Novák, J.: Nerf-tex: Neural reflectance field textures. In: Eurographics Symposium on Rendering (EGSR) (2021)
3. Basri, R., Galun, M., Geifman, A., Jacobs, D.W., Kasten, Y., Kritchman, S.: Frequency bias in neural networks for input of non-uniform density. In: International Conference on Machine Learning (ICML) (2020)
4. Benbarka, N., Höfer, T., ul Moqet Riaz, H., Zell, A.: Seeing implicit neural representations as fourier series. In: IEEE Winter Conference of Applications on Computer Vision (WACV) (2022)
5. Boscaini, D., Masci, J., Rodolà, E., Bronstein, M.M., Cremers, D.: Anisotropic diffusion descriptors. In: Computer Graphics Forum (CGF). vol. 35 (2016)
6. Boss, M., Braun, R., Jampani, V., Barron, J.T., Liu, C., Lensch, H.: Nerd: Neural reflectance decomposition from image collections. In: IEEE International Conference on Computer Vision (ICCV) (2021)
7. Boss, M., Jampani, V., Braun, R., Liu, C., Barron, J.T., Lensch, H.P.A.: Neural-pil: Neural pre-integrated lighting for reflectance decomposition. *CoRR* **abs/2110.14373** (2021)
8. Bronstein, A.M., Bronstein, M.M., Kimmel, R.: Numerical geometry of non-rigid shapes. Springer Science & Business Media (2008)
9. Burghard, O., Dieckmann, A., Klein, R.: Embedding shapes with green’s functions for global shape matching. *Computers & Graphics* **68C** (2017)
10. Chen, Z., Nobuhara, S., Nishino, K.: Invertible neural BRDF for object inverse rendering. In: European Conference on Computer Vision (ECCV) (2020)
11. Chibane, J., Pons-Moll, G.: Implicit feature networks for texture completion from partial 3d data. In: European Conference on Computer Vision (ECCV) SHARP Workshop (2020)
12. Crane, K., Weischedel, C., Wardetzky, M.: The heat method for distance computation. *Communications of the ACM* **60** (2017)
13. Dai, F., Xu, Y.: Approximation theory and harmonic analysis on spheres and balls. Springer (2013)
14. Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., Katam, H.: Blenderproc. arXiv preprint arXiv:1911.01911 (2019)
15. 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)

16. Gargan, D., Neelamkavil, F.: Approximating reflectance functions using neural networks. In: Eurographics Workshop on Rendering Techniques (1998)
17. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH) (1997)
18. Hertz, A., Perel, O., Giryès, R., Sorkine-Hornung, O., Cohen-Or, D.: Mesh draping: Parametrization-free neural mesh transfer. CoRR [abs/2110.05433](#) (2021)
19. Hertz, A., Perel, O., Giryès, R., Sorkine-Hornung, O., Cohen-Or, D.: Sape: Spatially-adaptive progressive encoding for neural optimization. In: Conference on Neural Information Processing Systems (NeurIPS) (2021)
20. Jacot, A., Hongler, C., Gabriel, F.: Neural tangent kernel: Convergence and generalization in neural networks. In: Conference on Neural Information Processing Systems (NeurIPS) (2018)
21. Kovnatsky, A., Bronstein, M.M., Bronstein, A.M., Glashoff, K., Kimmel, R.: Coupled quasi-harmonic bases. Computer Graphics Forum **32** (2013)
22. Lee, J., Jin, K.H.: Local texture estimator for implicit representation function. CoRR [abs/2111.08918](#) (2021)
23. Liu, X., Donate, A., Jemison, M., Mio, W.: Kernel functions for robust 3d surface registration with spectral embeddings. In: International Conference on Pattern Recognition (ICPR) (2008)
24. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: a skinned multi-person linear model. ACM Transactions on Graphics (TOG) **34** (2015)
25. Marin, R., Cosmo, L., Melzi, S., Rampini, A., Rodolá, E.: Spectral geometry in practice. 3DV Tutorial (2021)
26. Mehta, I., Gharbi, M., Barnes, C., Shechtman, E., Ramamoorthi, R., Chandraker, M.: Modulated periodic activations for generalizable local functional representations. CoRR [abs/2104.03960](#) (2021)
27. Meronen, L., Trapp, M., Solin, A.: Periodic activation functions induce stationarity. CoRR [abs/2110.13572](#) (2021)
28. Michel, O., Bar-On, R., Liu, R., Benaim, S., Hanocka, R.: Text2mesh: Text-driven neural stylization for meshes. CoRR [abs/2112.03221](#) (2021)
29. 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)
30. Morreale, L., Aigerman, N., Kim, V.G., Mitra, N.J.: Neural surface maps. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
31. Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A.A., Sohl-Dickstein, J., Schoenholz, S.S.: Neural tangents: Fast and easy infinite neural networks in python. In: International Conference on Learning Representations (ICLR) (2020)
32. 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)
33. 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)
34. Palafox, P., Bozic, A., Thies, J., Nießner, M., Dai, A.: Neural parametric models for 3d deformable shapes. In: IEEE International Conference on Computer Vision (ICCV) (2021)
35. Parlett, B.N.: The symmetric eigenvalue problem. Siam (1998)

36. Peng, L.W., Shamsuddin, S.M.H.: 3d object reconstruction and representation using neural networks. In: International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia (GRAPHITE) (2004)
37. Piperakis, E., Kumazawa, I.: Affine transformations of 3d objects represented with neural networks. In: IEEE International Conference on 3-D Digital Imaging and Modeling (2001)
38. Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F.A., Bengio, Y., Courville, A.C.: On the spectral bias of neural networks. In: International Conference on Machine Learning (ICML) (2019)
39. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Conference on Neural Information Processing Systems (NeurIPS) (2007)
40. Ramasinghe, S., Lucey, S.: Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. CoRR [abs/2111.15135](#) (2021)
41. Ramasinghe, S., Lucey, S.: Learning positional embeddings for coordinate-mlps. CoRR [abs/2112.11577](#) (2021)
42. Rustamov, R.M.: Laplace-beltrami eigenfunctions for deformation invariant shape representation. In: Symposium on Geometry Processing (SGP) (2007)
43. Sharp, N., Attaiki, S., Crane, K., Ovsjanikov, M.: Diffusionnet: Discretization agnostic learning on surfaces. ACM Transactions on Graphics (TOG) **XX** (20XX)
44. Sharp, N., Crane, K.: A laplacian for nonmanifold triangle meshes. Computer Graphics Forum **39** (2020)
45. Sharp, N., Soliman, Y., Crane, K.: The vector heat method. ACM Transactions on Graphics (TOG) **38** (2019)
46. 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)
47. Sitzmann, V., Martel, J.N.P., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions. In: Conference on Neural Information Processing Systems (NeurIPS) (2020)
48. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Symposium on Geometry Processing (SGP) (2007)
49. Sun, J., Ovsjanikov, M., Guibas, L.: A concise and provably informative multi-scale signature based on heat diffusion. In: Symposium on Geometry Processing (SGP) (2009)
50. 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)
51. Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P.P., Tretschk, E., Wang, Y., Lassner, C., Sitzmann, V., Martin-Brualla, R., Lombardi, S., Simon, T., Theobalt, C., Nießner, M., Barron, J.T., Wetzstein, G., Zollhöfer, M., Golyanik, V.: Advances in neural rendering. CoRR [abs/2111.05849](#) (2021)
52. Vaxman, A., Ben-Chen, M., Gotsman, C.: A multi-resolution approach to heat kernels on discrete surfaces. ACM Transactions on Graphics (TOG) **29** (2010)
53. Vestner, M., Lähner, Z., Boyarski, A., Litany, O., Slossberg, R., Remez, T., Rodolà, E., Bronstein, A.M., Bronstein, M.M., Kimmel, R., Cremers, D.: Efficient deformable shape correspondence via kernel matching. In: International Conference on 3D Vision (3DV) (2017)
54. Wang, P., Liu, Y., Yang, Y., Tong, X.: Spline positional encoding for learning 3d implicit signed distance fields. In: International Joint Conference on Artificial Intelligence (IJCAI) (2021)

55. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* **13** (2004)
56. Wimbauer, F., Wu, S., Rupprecht, C.: De-rendering 3d objects in the wild. *CoRR* **abs/2201.02279** (2022)
57. 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)
58. Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., Sridhar, S.: Neural fields in visual computing and beyond (2021)
59. Yifan, W., Rahmann, L., Sorkine-hornung, O.: Geometry-consistent neural shape representation with implicit displacement fields. In: *International Conference on Learning Representations (ICLR)* (2022)
60. Yüce, G., Ortiz-Jiménez, G., Besbinar, B., Frossard, P.: A structured dictionary perspective on implicit neural representations. *CoRR* **abs/2112.01917** (2021)
61. Zhang, K., Luan, F., Wang, Q., Bala, K., Snavely, N.: Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2021)
62. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018)
63. Zheng, J., Ramasinghe, S., Lucey, S.: Rethinking positional encoding. *CoRR* **abs/2107.02561** (2021)

Supplementary Material

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

The high-resolution intrinsic neural texture field on the human model that we showcase in Fig. 1 is available as a textured mesh on [sketchfab](https://skfb.ly/otvFK)⁴. 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.

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 [44]. 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 [44] supports calculating eigenfunctions on pointclouds.

Depending on whether the viewing direction is taken into account, we use the respective network architectures shown in Fig. 8 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 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 [29].

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}

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

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

⁴ <https://skfb.ly/otvFK>

B Experimental Details

B.1 Modification of NeuTex

In order to make the comparison between our method and NeuTex [57] 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 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 Fig. 9.

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

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

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 given geometry:

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

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 $\sigma = 8$.

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

B.2 Texture Reconstruction

In this experiment, we use the same [cat](#)⁶ and [human](#)⁷ mesh as in [32]. The 2D views of the meshes are rendered using Blender and blenderproc [14]. 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 [Fig. 10](#). The hyperparameters are given in [Tab. 4](#). Further qualitative results for [Sec. 5.1](#) can be found in [Fig. 12](#). 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 [Fig. 11](#). In [Tab. 5](#) the hyperparameters for the high-resolution human are shown.

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. [43]. The hyperparameters can be found in [Tab. 4](#). 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 [Tab. 6](#). Since the input to our

⁵ <https://github.com/fbxian/NeuTex>

⁶ <https://free3d.com/3d-model/cat-v1-522281.html>

⁷ <https://www.turbosquid.com/3d-models/water-park-slides-3d-max/1093267>

⁸ <https://github.com/nmwsharp/discretization-robust-correspondence-benchmark>

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

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 [33] do. We use the method of [15], 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 objects detergent and cinnamon toast crunch from the BigBIRD dataset [46]. The dataset provides images from different directions, foreground-background segmentation masks, camera calibration, and a reconstructed mesh of the geometry of each 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 Fig. 8 and uses the hyperparameters shown in Tab. 7. We will release the preprocessing code and the training data used in this experiment along with the final publication.

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

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 [Tab. 8](#). 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 [Fig. 13](#) we demonstrate that the composed neural tangent kernel (NTK) can be non-stationary for a general 2-manifold. In [Sec. D.1](#) the proof for [Thm. 1](#) is given specifically for 1-manifolds. Finally, in [Sec. D.2](#) we provide further theoretical results regarding the NTK, specifically [Lem. 1](#), which was used in the proof of [Thm. 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 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

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.

Table 8: 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 \uparrow	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 \downarrow	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 \downarrow	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 \uparrow	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 \downarrow	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 \downarrow	0.306	0.309	0.305	0.309	0.305	0.306	0.306	0.301	0.303	0.305	0.306	0.763%

D.2 Neural Tangent Kernel

In this section, we prove [Lem. 1](#) 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. [\[20\]](#).

Lemma 1. *Let $k_{NTK} : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ 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*

$$k_{NTK}(x, x') = h_{NTK}(\langle x, x' \rangle), \quad (14)$$

for a scalar function $h_{NTK} : \mathbb{R} \rightarrow \mathbb{R}$.

Proof. For this proof we adopt the notation of Jacot et al. [\[20\]](#) 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 [\[20, Prop. 1\]](#). Jacot et al. show that the neural network Gaussian process (NNGP) has covariance $\Sigma^{(L)}$ defined recursively

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

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

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

where $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ 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 [\[20\]](#). 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 $A^{(L)}(x, x')$ 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:

$$\Theta_\infty^{(1)}(x, x') = \Sigma^{(1)}(x, x') \quad (18)$$

$$\Theta_\infty^{(L+1)}(x, x') = \Theta_\infty^{(L)}(x, x') \dot{\Sigma}^{(L+1)}(x, x') + \Sigma^{(L+1)}(x, x') \quad (19)$$

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

where $\dot{\sigma}$ is the derivative of the non-linearity. By a similar induction argument to above we obtain that $\Theta_\infty^{(L)}(x, x')$ only depends on $x^\top x'$ and hence that $\Theta_\infty^{(L)}(x, x)$ does not depend on x . In the notation of our [Lem. 1](#) this means that k_{NTK} only depends on $\langle x, x' \rangle$ and can thus be written as $h_{NTK}(\langle x, x' \rangle)$ for a scalar function $h_{NTK} : \mathbb{R} \rightarrow \mathbb{R}$. \square

Positive-definiteness of the NTK. In the proof of [Thm. 1](#), we used the fact that the NTK is positive definite as shown by [[20](#), 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

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

where $\hat{\mu} : [-1, 1] \rightarrow \mathbb{R}$ is the dual in the sense of [[20](#), Lem. 2] of the function $\mu : \mathbb{R} \rightarrow \mathbb{R}$ defined by $\mu(x) = \sigma \left(x \sqrt{r^2/n_0 + \beta^2} \right)$. Finally, in step 5 of their proof

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

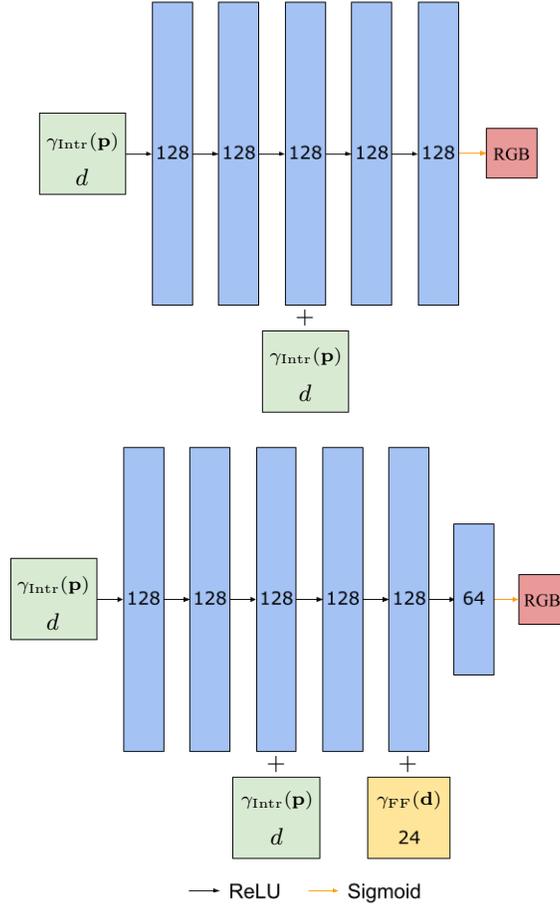


Fig. 8: Network architecture. We use a similar network architecture as used for NeRF [29]. A point on the 2-manifold is described by \mathbf{p} . The unit ray direction is represented by \mathbf{d} . The notations γ_{Intr} and γ_{FF} represent the proposed eigenfunction embedding and the sine/cosine positional encoding [29] 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.

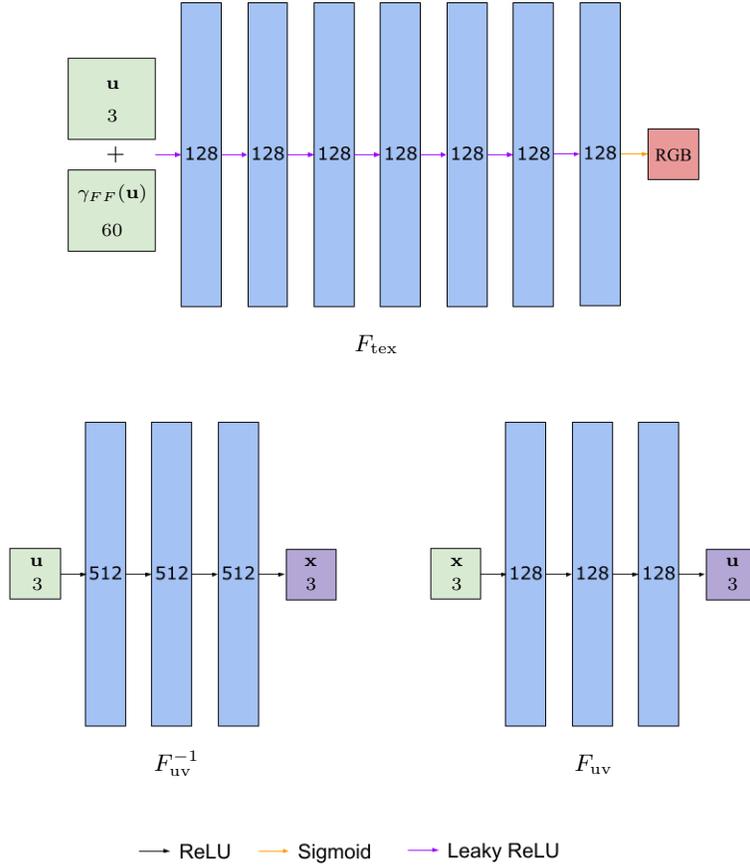


Fig. 9: Modified network architecture for NeuTex [57]. We remove the volume density as well as the view dependence enabling NeuTex to learn a simpler setting because the geometry is known and the textures are diffuse in the experiment of Sec. 5.1. A 3D coordinate on the sphere representing the uv-space 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 [29] is γ_{FF} . The + sign denotes concatenation.

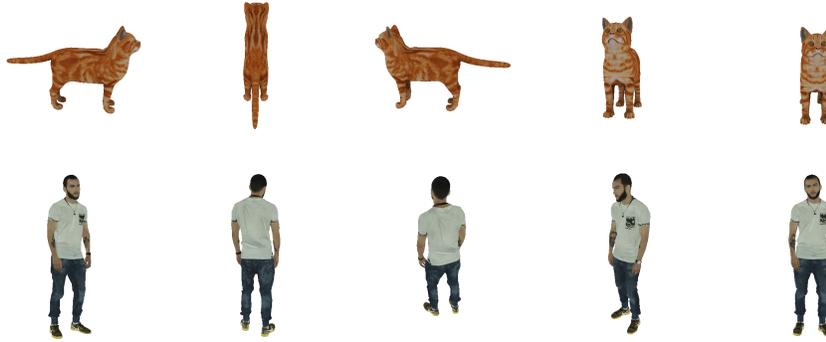


Fig. 10: Training views for the cat and human datasets with 5 views used in [Sec. 5.1](#), [Sec. 5.2](#), and [Sec. 5.3](#).

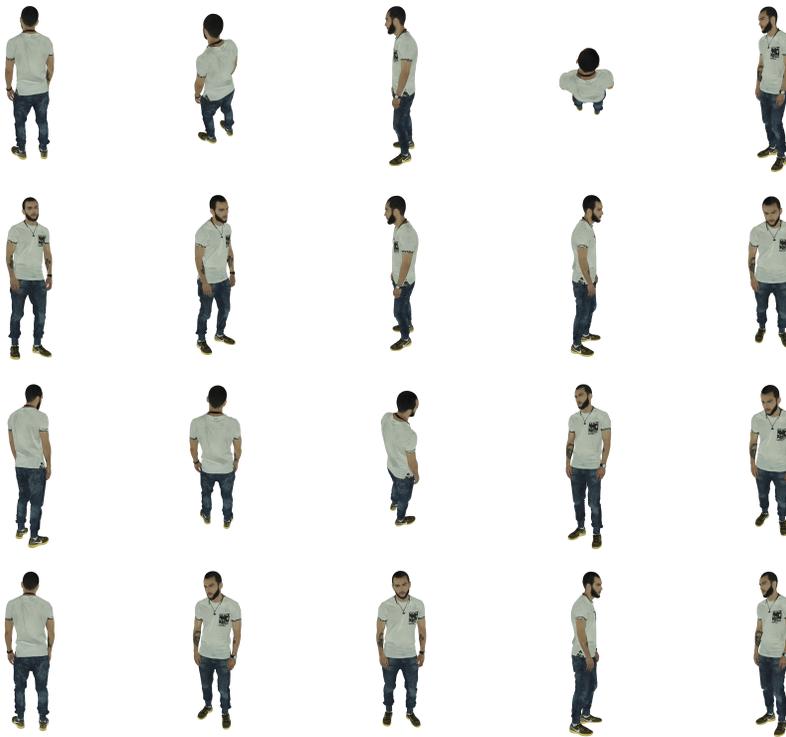


Fig. 11: Training views for the human high resolution dataset. Due to the large size of the 4096×4096 png images, we converted them to jpg and scaled them down to 1024×1024 for this figure.

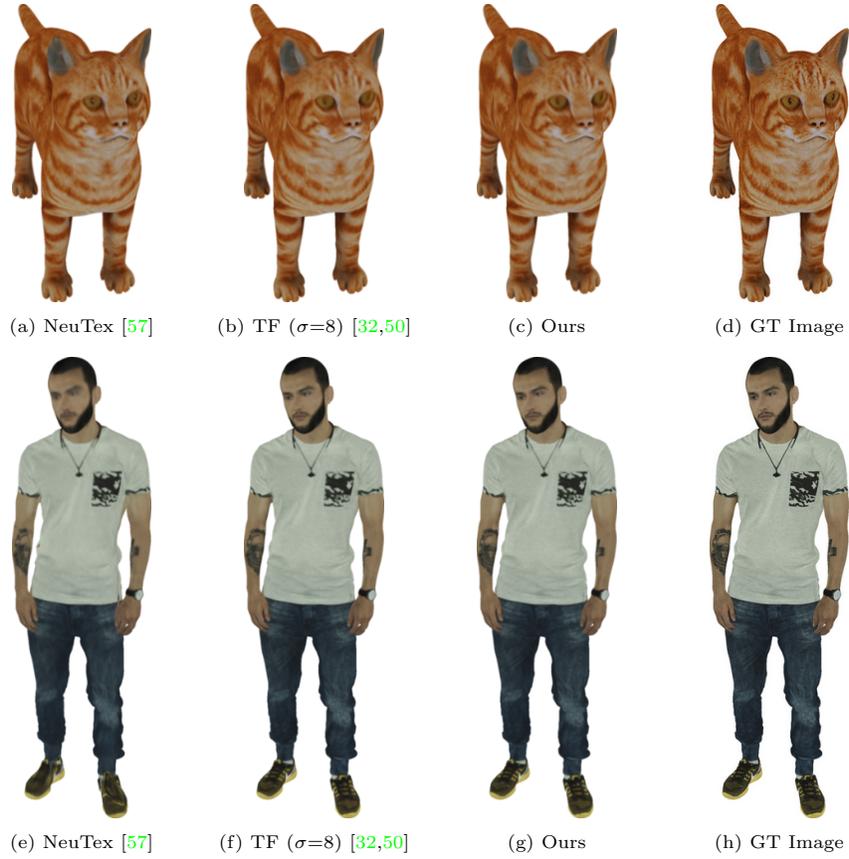


Fig. 12: Further qualitative comparisons of unseen views from the texture reconstruction experiment (Sec. 5.1). All the methods from this figure use an embedding size of 1023. We want to point out that these renderings are not high-quality due to the low resolution of the training views.

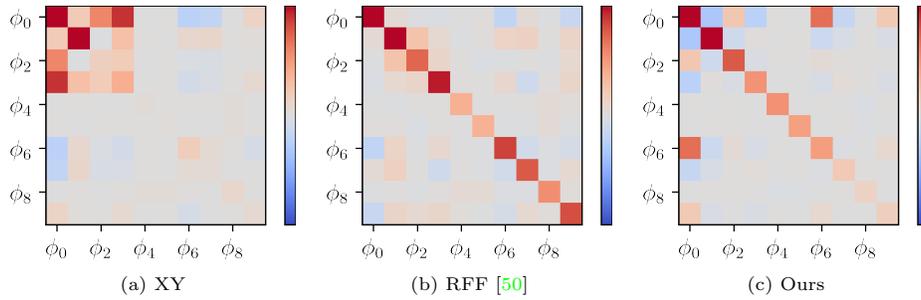


Fig. 13: Non-stationarity of NTKs on a general 2-manifold. We investigate the non-stationarity of the neural tangent kernels (NTKs) on a general 2-manifold, namely the cat shown in Fig. 3. Each small square in the image shows the coefficient c_{ij} when projecting the kernel onto the LBO basis: $c_{ij} = \int_{\mathcal{M}} \int_{\mathcal{M}} \phi_i(\mathbf{p})k(\mathbf{p}, \mathbf{q})\phi_j(\mathbf{q})d\mathbf{p}d\mathbf{q}$. The integral is approximated numerically as the area-weighted sum over the vertices. A stationary kernel as defined in Eq. 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.