MeshFeat: Multi-Resolution Features for Neural Fields on Meshes

Mihir Mahajan[*,1] ⬡, Florian Hofherr[*,1,2] ⬡, and Daniel Cremers[1,2] ⬡

[1] Technical University of Munich
[2] Munich Center for Machine Learning

Abstract. Parametric feature grid encodings have gained significant attention as an encoding approach for neural fields since they allow for much smaller MLPs, which significantly decreases the inference time of the models. In this work, we propose MeshFeat, a parametric feature encoding tailored to meshes, for which we adapt the idea of multi-resolution feature grids from Euclidean space. We start from the structure provided by the given vertex topology and use a mesh simplification algorithm to construct a multi-resolution feature representation directly on the mesh. The approach allows the usage of small MLPs for neural fields on meshes, and we show a significant speed-up compared to previous representations while maintaining comparable reconstruction quality for texture reconstruction and BRDF representation. Given its intrinsic coupling to the vertices, the method is particularly well-suited for representations on deforming meshes, making it a good fit for object animation.

Keywords: Feature Encodings · Multi-Resolution · Meshes

1 Introduction

Capturing and modeling our 3D world realistically and effectively is becoming increasingly relevant for AR, VR, CGI, simulations, and computer games. Classical approaches for scene representation often rely on explicit discrete data structures like voxel grids to store function values like SDF. Alternatively, point clouds, and meshes are used to store geometry or UV-Maps, and discrete texture maps to represent appearance. For all these approaches, the memory footprint is strongly coupled to the resolution, making high-resolution representations often impractical. Since the breakthrough work by Mildenhall et al. [30], the focus has shifted towards neural implicit representations, which use Multilayer Perceptrons (MLP) to represent scenes in a continuous and resolution-free manner.

The key to high-quality reconstructions is to use input encodings to overcome the spectral bias of neural networks [35] and enable the model to learn high-frequency details. Models based on frequency encodings like positional encoding [30, 44] or Fourier features [36, 42] tend to be costly and slow to evaluate since all the information on the scene is stored in the weights of the MLP. This results

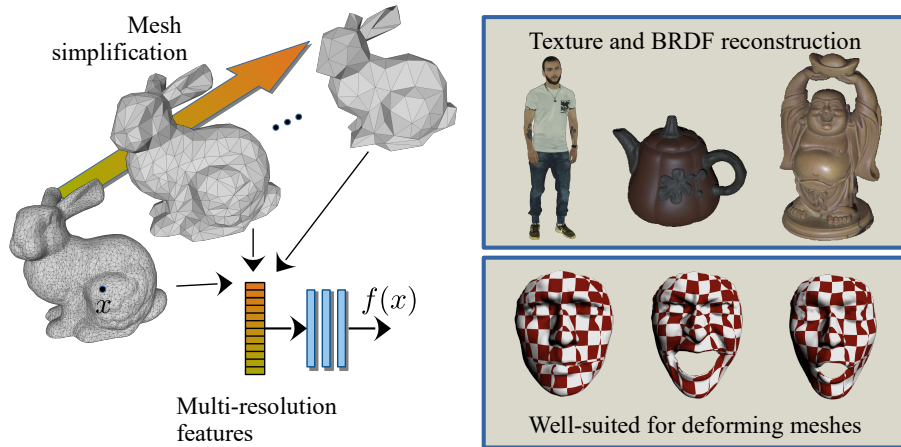---

[*] Equal contribution

Fig. 1: We present MeshFeat, a parametric encoding strategy for neural fields on meshes. We propose a multi-resolution strategy based on mesh simplification to enhance the efficiency of the encoding. Our approach allows for much smaller MLPs than previous frequency-based encodings, resulting in significantly faster inference times. We evaluate the method for texture reconstruction and BRDF estimation and demonstrate, that the encoding is well suited to represent signals on deforming meshes.

in a strong coupling of different parts of the scene and necessitates large neural networks. On the other hand, the use of multi-resolution feature grids [5, 7, 16, 22, 25, 26, 34, 40, 43] as input encoding has led to a large gain in evaluation speed, making the models real-time capable. The idea of feature grids is to store local information about the scene in spatially distributed features and use only a small MLP to decode this information to the quantity of interest. This decoupling of spatial information from information decoding, combined with the resulting possibility of smaller networks, is the reason for the performance gain for those approaches.

While neural representations have demonstrated impressive performance, numerous 3D computer graphics pipelines continue to utilize meshes as a fundamental data structure. Meshes remain a preferred choice in many applications due to the existence of efficient algorithms, intuitive editing capabilities, and convincing animation possibilities. Consequently, a noteworthy avenue of research explores the integration of mesh-based representations with neural fields, aiming to capitalize on the strengths of both approaches. Texture fields by Oechsle *et al.* [32] embed the points on the mesh into the surrounding Euclidean space and leverages neural fields to regress texture values. NeuTex [48] and other methods [2, 42] follow the same idea, but additionally use frequency encodings to capture higher frequency details and circumvent the spectral bias issue. Unfortunately, these Euclidean embeddings do not take the mesh geometry into account, and for points where Euclidean and geodesic distance varies significantly, these methods can show bleeding artifacts [19]. Intrinsic neural fields by Koestler *et al.* [19] ad-

dresses this by transferring the idea of frequency encodings from the Euclidean space to manifolds by using the eigenfunctions of the Laplace-Beltrami Operator. This makes the encoding intrinsic to the mesh. While they show promising results in reconstruction quality, they require heavy preprocessing to calculate the eigenfunctions. Moreover, they require a large number of eigenfunctions, resulting in a substantial memory footprint of the model. All methods employ frequency-based encodings and, therefore, suffer from the requirement of large networks, hindering efficient inference and rendering.

In this work, we propose MeshFeat, a novel parametric multi-resolution feature encoding for neural fields on meshes, for which we adopt ideas of feature grids from Euclidean space. We leverage the spatial structure given by the mesh and employ a mesh simplification algorithm for our multi-resolution strategy. Similarly to feature grids in Euclidean space, decoupling spatial information and decoding allows for much smaller MLPs and significantly accelerates the inference compared to previous methods. A schematic overview of our approach can be seen in Fig. 1.

We summarize our main contributions as follows:

- We introduce MeshFeat, a parametric feature encoding for neural fields on meshes which includes an effective multi-resolution strategy based on mesh simplification.
- We demonstrate that this feature encoding allows for significantly smaller MLPs compared to previous approaches, leading to a large speed-up in inference time.
- We perform a thorough comparison of our approach to state-of-the-art methods for the tasks of single-object texture reconstruction and BRDF estimation. Moreover, we demonstrate the native applicability of our approach to signal representation on deforming meshes.

Please see our project page at https://maharajamihir.github.io/MeshFeat/ for the source code.

2   Related Work

Neural Fields Scene representation based on coordinate-based networks has been used extensively in recent years. The idea is to use a simple multi-layer perception (MLP) to map a spatial position to quantities of interest like occupancy [6, 27, 34], signed distance [11, 33], texture [32], density and color for volume rendering [30, 46, 50] and more. In contrast to classical methods for scene representation purely based on explicit and discrete data structures like voxel grids, these implicit approaches are not limited by a discrete resolution resulting in a much more memory-efficient representation. The key to high-fidelity reconstructions with neural fields is to encode the input to a higher dimensional space to overcome the low-frequency bias of neural networks [3, 35].

Input Encodings The different input encoding strategies can be divided into two categories: frequency-based encodings and parametric approaches. Among the most popular choices for the first type are positional encodings, which originate from transformer architectures [44] and have been first introduced in the seminal NeRF paper [30] for neural fields. The idea is to use a sequence of sine and cosine functions to encode the input coordinates. Tancik *et al.* analyze these positional encodings using neural tangent kernel analysis and generalize the idea to the Fourier feature encoding [42]. A similar effect is achieved by SIRENs, which are MLP with periodic activation functions [38]. Hertz *et al.* complement this idea with an adaptive frequency masking scheme [13].

In contrast, parametric encodings do not directly apply an encoding function to the input. Instead, they use the input coordinate as an interpolation point for learnable features stored in more classical data structures like grids [5,7,16, 22,25,26,34,40,43]. To evaluate the model, the grid features are interpolated at the input coordinate, and the result is used as an input for the MLP. Extending a single-resolution grid to a multi-resolution approach adds different scales of locality to the model. This decoupling of local and global information helps to improve efficiency and convergence properties. As further improvements, the use of hash grids [31] and octrees [41] have been proposed.

Parametric encodings allow for much smaller MLPs than their frequency-based counterparts, making evaluating these models significantly faster. Intuitively, feature encodings contain the information "What is at this position" and the MLP decodes this into quantities of interest. In contrast, frequency-based encodings do not contain learned information, and the MLP needs to store all the information on the scene in its weights. The smaller computational cost for parametric encodings comes at the price of a larger memory footprint.

Input Encodings for Neural Fields on Meshes Encoding strategies specifically for meshes have received much less attention than encodings for neural fields in the Euclidean space. A common strategy, used *e.g.* by Texture Fields [32], is to embed the points on the mesh into the surrounding Euclidean space. This enables the use of established encoding strategies like Fourier features, as done by Text2Mesh [28] or variants of positional encoding, as done by Hertz *et al.* [14].

The drawback of this strategy is that the embedding to the surrounding space makes the encodings *extrinsic* and the properties of the underlying manifold represented by the mesh are not considered, which can lead to artifacts [19]. Koestler *et al.* show, that the equivalent to positional encoding in the Euclidean space are the eigenfunctions of the Laplace-Beltrami Operator on manifolds [19]. Grattarola and Vandergheynst investigate a similar idea [10]. This results in an *intrinsic* encoding that respects the properties of the manifold and enables applications like texture transfer. This approach has been used successfully to model deformation fields on meshes [23,45] and for scene stylization [15]. While intrinsic encodings are elegant, in practice, many eigenfunctions have to be stored per vertex, resulting in a large model size. Also, the computation of the eigenfunctions is costly since the eigenvalues of a large matrix need to be computed.

All approaches for encodings on meshes discussed so far follow the idea of frequency encodings and thus require large MLPs for high-quality reconstruction, resulting in slow evaluation times. In contrast, we present a parametric encoding on a mesh, enhanced by a multi-resolution strategy. Only very few previous works consider this form of encoding. Yang *et al.* [49] train features on a mesh, which they use as a scaffold for a neural scene representation in 3D. In contrast to our work, their approach does not contain hierarchical features, and moreover, they do not consider a signal restricted to the mesh surface. While Kim *et al.* propose multi-resolution features, they only consider spherical meshes [18]. In contrast, we are not restricted to a certain class of meshes.

Multi-Resolution Approaches on Meshes Multi-resolution approaches for meshes have been used for various tasks. Lee *et al.* use mesh simplification based on vertex removal for adaptive re-meshing [20]. Liu *et al.* investigate mesh simplification for multi-grid solvers on curved surfaces [24]. Jiang *et al.* propose a prismatic shell for meshes that can help to avoid geometric artifacts for mesh simplification algorithms for extreme vertex reductions at the cost of additional computational overhead [17]. MeshCNN uses a learned mesh-pooling for a multi-resolution approach to mesh analysis tasks [12].

3   Method

In this section we present MeshFeat, a parametric encoding strategy for the parametrization of neural fields directly on meshes. We adapt the idea of multi-resolution grids for the Euclidean space to meshes to obtain an efficient encoding. Instead of using a regular voxel grid in Euclidean space, we use the mesh vertices as pre-defined locations to store the feature vectors. We apply a mesh simplification algorithm to obtain different resolutions of the initial mesh for our multi-resolution approach. Similar to feature grids in Euclidean space, our approach allows to use very small MLPs resulting in a fast evaluation time.

3.1   Multi-Resolution Feature Encoding

Mesh Simplification To obtain our multi-resolution feature encoding, we simplify the initial mesh $M = (V, F)$ to multiple resolutions using the quadric error metric decimation by Garland and Heckbert [8,9]. We denote the sequence of resolutions for the mesh simplification by $(r^{(i)})_i$, where $r^{(i)} \in [0, 1]$ means, that the i-th resolution has $|V^{(i)}| = r^{(i)}|V|$ number of vertices. This yields the sequence of meshes $((V^{(i)}, F^{(i)}))_i$. Moreover, we store the mapping

$$m^{(i)} : V \to V^{(i)} \quad v \mapsto m^{(i)}(v) = v^{(i)} \tag{1}$$

that assigns a vertex $v \in V$ in the original mesh to the vertex $v^{(i)} \in V^{(i)}$ in resolution i to which v was collapsed in the decimation algorithm. Note that in contrast to other simplification-based multi-resolution schemes, we do not compute a geometric mapping between the resolutions, which reduces the computational overhead. Please see the appendix for further information.
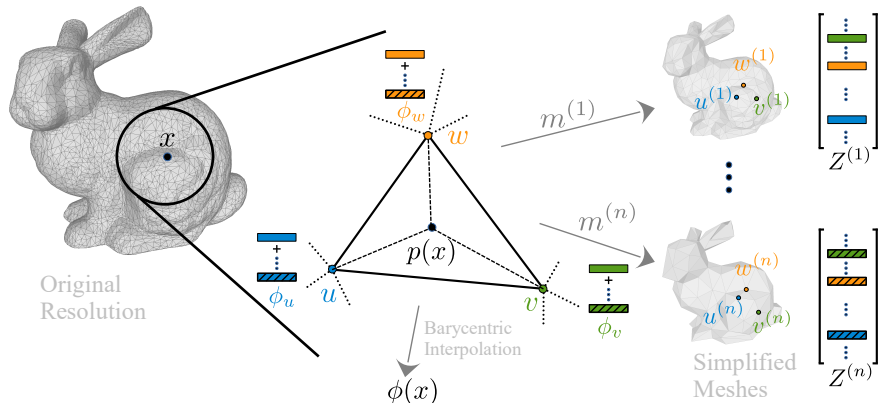
Fig. 2: Overview of our multi-resolution feature approach on the mesh. To get the feature encoding $\phi(x)$ for a point x on the original mesh, we determine the vertices u, v, w of the respective triangle. Using the mappings $m^{(i)}$, we gather the corresponding features from the different resolutions. By summing them, we obtain the features ϕ_u, ϕ_v, ϕ_w at the vertices in the original mesh. We receive the final feature encoding $\phi(x)$ by barycentric interpolation of the features at the vertices.

Multi-Resolution Strategy For each of the resolutions, we use a learnable feature matrix $Z^{(i)} \in \mathbb{R}^{|V^{(i)}| \times d}$, where $d \in \mathbb{N}$ is the feature dimension. We denote the feature vector of vertex $v^{(i)} \in V^{(i)}$ by $Z^{(i)}(v^{(i)}) \in \mathbb{R}^d$.

In contrast to parametric encodings on Euclidean grids, we do not interpolate the features for each resolution but rather accumulate the features from all resolutions on the original resolution and interpolate the results only once. The reason is that a point on the mesh for a certain resolution does not have a direct meaning on the meshes of the other resolutions. The Euclidean embedding of the point, for example, does not even need to be on the other meshes. Therefore, we use the mappings $m^{(i)}$ to "pull" all features of the coarser resolutions to the finest resolution (which is where we want to evaluate the function on the mesh). We obtain the combined feature vector ϕ_v for a vertex $v \in V$ in the original mesh by summing the feature vectors from all resolutions

$$\phi_v = \sum_i Z^{(i)}(m^{(i)}(v)). \tag{2}$$

See Fig. 2 for an overview of our multi-resolution feature pipeline.

Feature Interpolation To obtain the feature vector for an arbitrary point x on the mesh, we compute the barycentric coordinates $p(x) = [\lambda_1, \lambda_2, \lambda_3]^\top$ within the respective triangle of the original mesh. Let $v_1, v_2, v_3 \in V$ be the vertices of this triangle. Then the encoding $\phi(x)$ for the point x reads

$$\phi(x) = \sum_i \lambda_i \phi_{v_i} = [\phi_{v_1}, \phi_{v_2}, \phi_{v_3}] \, p(x). \tag{3}$$

3.2   Feature Regularization

Depending on the training setup, it can happen that the features for some vertices will never receive a training signal. For example, for training from images and very fine meshes, some triangles might not be hit during the ray mesh intersection leading to missing supervision for vertices in that region. We use a regularization of the features based on the mesh Laplacian to avoid the resulting artifacts. We denote the mesh Laplacian of the mesh in the original resolution by $L \in \mathbb{R}^{|V| \times |V|}$. To remove the dependency on the mesh scale and avoid the necessity to tune the regularization weight to each mesh individually, we normalize L by its spectral norm, $i.e.$ we consider $\hat{L} = L/\|L\|_2$. For the regularization, we accumulate the features for all vertices of the original resolution in the matrix $\Phi \in \mathbb{R}^{|V| \times d}$ and sum the absolute values of the normalized Laplacian applied to the features, $i.e.$

$$\mathcal{L}_{\text{reg}} = \sum_{i,j} |(\hat{L}\Phi)_{i,j}|. \tag{4}$$

This can be seen as the sum over the 1-norms of the Laplacian applied to the individual entries of the latent codes. This loss term penalizes large variations of neighboring feature values since the Laplacian computes the deviation from the local average. Still, due to the 1-norm, we allow for sparse larger changes. Note that while we compute the loss term only for the original resolution, the regularization affects all resolutions since we apply the Laplacian to the accumulated features. In practice, we use the robust Laplacian by Sharp and Crane [37].

3.3   Model Architecture and Training Details

To decode the feature vectors into the final function value, we employ a standard MLP with ReLU activation functions. For the experiments in this paper, we use a hidden dimension of 32 with 2 hidden layers. Also, we employ a sigmoid output-nonlinearity since all of our reconstructed signals are in the range $[0, 1]$. To prevent overfitting, a weak L2 regularization (with weight 10^{-5}) proved to be helpful.

We found that a total of 4 resolutions $r^{(i)} \in \{1, 0.1, 0.05, 0.01\}$ for the mesh simplification achieved optimal results for all experiments. Note, that for $r^{(1)} = 1$, the mapping $m^{(1)}$ yields the identity function, $i.e.$ we use the original mesh as our finest resolution. We initialize the feature matrices using a normal distribution with $\sigma = 5 \cdot 10^{-4}$.

For training, we use a batch size of $b = 8000$. We train for 1000 epochs for the texture reconstruction task and for 500 epochs for the BRDF estimation. We found that different learning rates for the MLP parameters and the features are crucial. We use $\mathrm{lr}_\theta = 2 \cdot 10^{-4}$ for the weights of the neural network and $\mathrm{lr}_Z = 5 \cdot 10^{-3}$ for our latent codes. We use a factor $\lambda_{\text{reg}} = 1.5 \cdot 10^{-6}$ to balance the regularization loss with the data loss.

4   Experiments

In this section, we present a detailed evaluation of our proposed approach. First, we evaluate our method for single-object texture reconstruction from multi-view images. We compare against state-of-the-art methods [19, 32, 48]. Moreover, we perform an ablation study to evaluate the significance of our modeling choices. As a second application of our method, we consider the reconstruction of a parametric BRDF from multi-view images of a single object with calibrated lighting. We use our approach to estimate the parameters of the well-known Disney BRDF [4] spatially varying on the mesh. Furthermore, we demonstrate that our method is well-suited to represent quantities on deforming meshes due to the tight coupling with the mesh. We refer to the supplementary material for additional experiments and further training information.

Concurrent Models We compare our encoding strategy to state-of-the-art work for neural fields on meshes with different encoding strategies. We use Texture Fields (TF) [32] as a method with an *extrinsic* encoding strategy. Consistent with the experiments done by Koestler *et al.* [19], we augment the method with random Fourier features (RFF). Moreover, we compare against Intrinsic Neural Fields (INF) [19], which uses an intrinsic and frequency-based encoding based on the eigenfunctions of the Laplace-Beltrami operator. Both methods use an MLP with 6 hidden layers of width 128. To keep the experiments consistent with previous work [19], we also include a modified version of NeuTex [48] in our experiments on texture reconstruction, even though the method was originally designed for geometry estimation along with texture. NeuTex follows a UV-mapping-based approach and learns separate networks for geometry, UV-mapping and texture. Following Koestler *et al.* [19], we allow it to take advantage of the given geometry. We refer the reader to their work for details on this modification. Finally, as a reference, we include a non-neural baseline, for which we learn color values directly on the vertices of the original mesh and use barycentric interpolation to obtain the values on the triangles. Again, we use our regularization term to account for unsupervised vertices. All methods are adapted into the same pipeline for a fair comparison.

4.1   Texture Reconstruction from Multi-View Images

We follow the experimental setup of Intrinsic Neural Fields [19], initially proposed by Oechsle et al. [32] and use the same dataset. The input to all methods is a set of five 512x512 pixel posed images alongside their respective intrinsic and extrinsic camera matrices. Additionally, the triangle mesh of the object is given. In the preprocessing stage, we compute the ray-mesh intersection for each pixel to obtain the corresponding point on the mesh. We use the Euclidean embedding of this point as input for TF and use barycentric coordinates in the respective triangle for the other methods. Additionally, for INF, the values of the eigenfunctions of the Laplace-Beltrami operator (LBO) must be computed at the

Table 1: Texture reconstruction from multi-view images. Our multi-resolution feature encoding significantly improves inference speed compared to state-of-the-art neural methods while maintaining similar reconstruction quality. Note that DSSIM and LPIPS are scaled by 100. The trade-off for the speed-up is a slight increase in the number of parameters; however, we still have a significantly smaller model compared to INF, which generally produces the best results. While the non-neural reference baseline has the fastest evaluation time, it shows significantly decreased reconstruction quality, particularly for the coarser mesh. For our method, using the regularizer shows a substantial improvement in reconstruction quality. While for coarser meshes like the *cat*, we obtain better reconstruction results for an increase in feature dimension ($d \uparrow$) to $d = 10$, we did observe slight overfitting for finer meshes like the *human*.

Object	Method	PSNR ↑	DSSIM ↓	LPIPS ↓	# Params. ↓	Speedup ↑
human $\|V\| = 129$k	NeuTex [48]	27.32	0.549	0.954	793k	1.0x
	TF+RFF [32,42]	32.10	0.232	0.423	**331k** 🏆	1.96x
	INF [19]	32.46 🏆	0.215	**0.390** 🏆	133130k	3.06x 🏆
	Ours ($\lambda_{reg} = 0$)	31.25	0.323	0.510	604k	13.49x
	Ours ($d \uparrow$)	32.46	0.203 🏆	0.410	1058k	12.08x
	Ours ($d = 4$)	**32.51** 🏆	**0.202** 🏆	0.400 🏆	604k 🏆	**13.49x** 🏆
	Non-neural (ref.)	32.01	0.225	0.432	391k	28.32x
cat $\|V\| = 33$k	NeuTex	31.56	0.338	0.336	793k	1.0x
	TF+RFF	34.33	**0.162** 🏆	0.247 🏆	331k 🏆	1.96x
	INF	**34.76** 🏆	0.166 🏆	**0.202** 🏆	36430k	3.07x
	Ours ($\lambda_{reg} = 0$)	33.27	0.305	0.453	166k	13.33x
	Ours ($d \uparrow$)	34.65 🏆	0.191	0.295	411k	11.94x 🏆
	Ours ($d = 4$)	34.23	0.238	0.349	**166k** 🏆	**13.33x** 🏆
	Non-neural (ref.)	33.01	0.400	0.775	123k	32.52x

mesh vertices. We would like to point out that this computation can take *hours*, depending on the mesh size, while the other methods do not require any preprocessing. For the training, we employ an L1 loss on the color values. We render 200 images for evaluation and report PSNR, DSSIM[3] [47] and LPIPS [51]. All experiments were done on an Nvidia A10G Tensor Core with 24GiB of memory.

The results in Tab. 1 and Fig. 3 show that our model achieves results on par with the state-of-the-art while showing a notable speed-up compared to previous approaches.

Reconstruction Quality Tab. 1 shows that our method yields quantitative results that are on par with the other methods for the reconstruction quality. As can be seen in Fig. 3, we even achieve a noticeable qualitative improvement for some regions, while other regions are slightly worse. We found that areas of high quality often correspond with a finer mesh in that area, which is to be expected since, for those regions, more features are available.

---

[3] For better readability we use the Structural Dissimilarity: $\text{DSSIM} = (1 - \text{SSIM})/2$

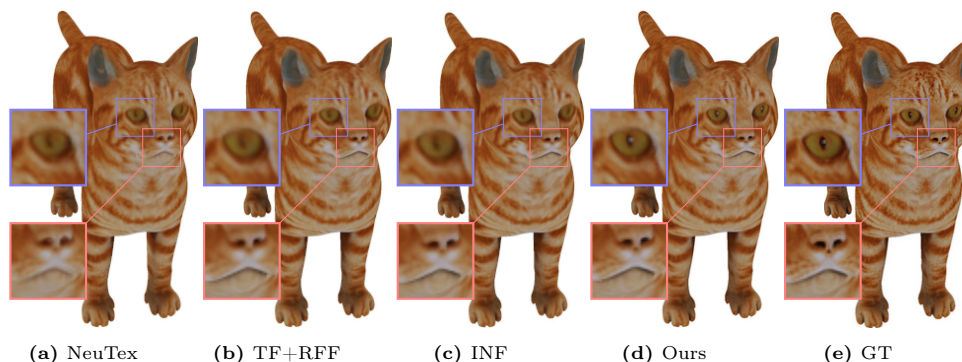| **(a)** NeuTex | **(b)** TF+RFF | **(c)** INF | **(d)** Ours | **(e)** GT |

Fig. 3: Qualitative results for texture reconstruction from multi-view images on the *cat*. Our method enables high-quality reconstructions, matching state-of-the-art methods in visual fidelity while offering a significant speedup. Because the baseline methods are based on frequency encodings, they lead to an over-smoothening of intricate details around the eye, which only our method can capture. Furthermore, NeuTex is unable to capture spatially fast changing color like on the mouth of the *cat* and shows distortions inside the ear.

Inference Time To benchmark the inference speed, we measure the evaluation time of the encoding stage and the MLP for 2^{15} points for each model. Note that this does not include the ray mesh intersection or the computation of the eigenfunctions. To reduce variance, we average the results over 300 measurements of the same batch. Tab. 1 shows a significant speed-up of our method compared to the other methods, which we attribute to the much smaller MLP size. Also, it is important to note that our encoding strategy is computationally lightweight since it involves a single matrix multiplication with low dimensions due to the small feature size. As expected, the non-neural baseline is the fastest.

Number of Parameters For the number of parameters, we count all parameters that are necessary to evaluate the model. This includes learnable components like the weights of the network and the features, as well as non-learnable ones like the Fourier feature matrix and the LBO eigenfunctions values. We observe that the trade-off for the evaluation speed of our method is a slightly increased number of parameters[4] due to the feature matrices. This observation is similar to feature-grid-based methods in the Euclidean space.

4.2   Ablation Study

Multi-Resolution Features We accumulate information distributed over multiple resolutions of our mesh to enable different scales of locality. This way, coarse resolutions can capture more global information, and fine resolutions can act as

---

[4] As a reference: a 512x512 3-channel color image consists of over 786k pixel values.
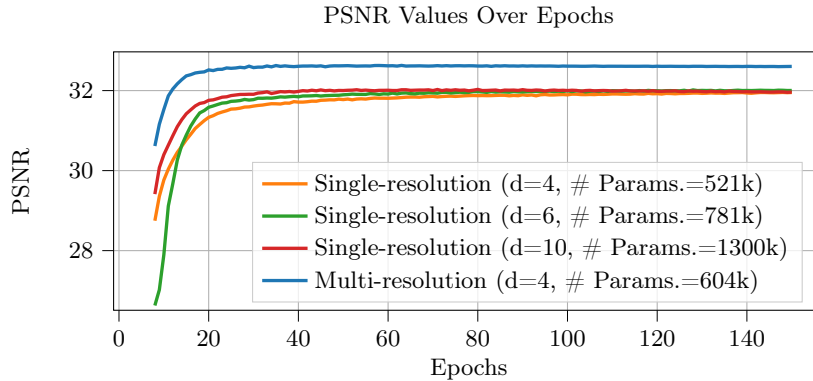
PSNR Values Over Epochs

Fig. 4: Validation PSNR over training time in epochs. Our multi-resolution feature encoding leads to higher reconstruction quality despite having fewer parameters than a single-resolution encoding. For the latter, we use the finest resolution and arrange parameters the same way as in the multi-resolution approach. While a higher feature dimension d leads to faster convergence in the single-resolution setting, it does not improve the reconstruction quality despite the increased number of parameters.

correcting terms for local changes. Fig. 4 shows the effectivity of our multi-resolution approach and demonstrates that a single-resolution setup struggles to achieve high visual fidelity, even if the number of parameters exceeds the multi-resolution approach. It confirms that our multi-resolution approach allows an effective sharing of global information on different scales and enables reconstructions of higher quality while maintaining rapid convergence.

Regularization Frequency encodings require the MLP to map encoded spatial information to function values. Due to strong coupling of spatial information in a single network, frequency encodings can interpolate for regions, where training data is sparse. For our parametric encoding approach, working with fine meshes and sparse training data can lead to unsupervised latent codes, creating visual artifacts in our reconstructions. Our latent code regularization (Eq. (4)) uses the mesh Laplacian to circumvent this issue. Fig. 5 shows how our regularization reduces these artifacts significantly. The positive effect or our regularization scheme is also underlined quantitatively in Tab. 1.

4.3   Deforming Meshes

For many real-world applications of meshes that include animations, it is crucial that the representation of *e.g.* texture can also be used under deformations of the mesh. Since our features are stored at the mesh vertices and are therefore intrinsic to the mesh, they are unaffected by deformations. Consequently, our method supports mesh deformations natively without any additional computational overhead. While the same is true for other intrinsic encodings [19],
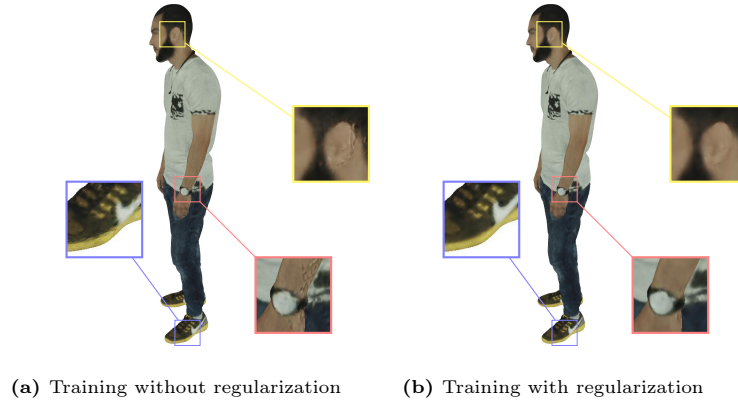
(a) Training without regularization   (b) Training with regularization

Fig. 5: Qualitative results of our method for texture reconstruction with and without the regularization based on the mesh Laplacian. The results on the left, without the regularization, show visual artifacts around the ear, and on the shoe and arm. This is a direct result of sparse training data resulting in untrained feature vectors. Our regularization acts as a smoothing term that enables feature information to be diffused to the unsupervised areas, significantly reducing the noise.
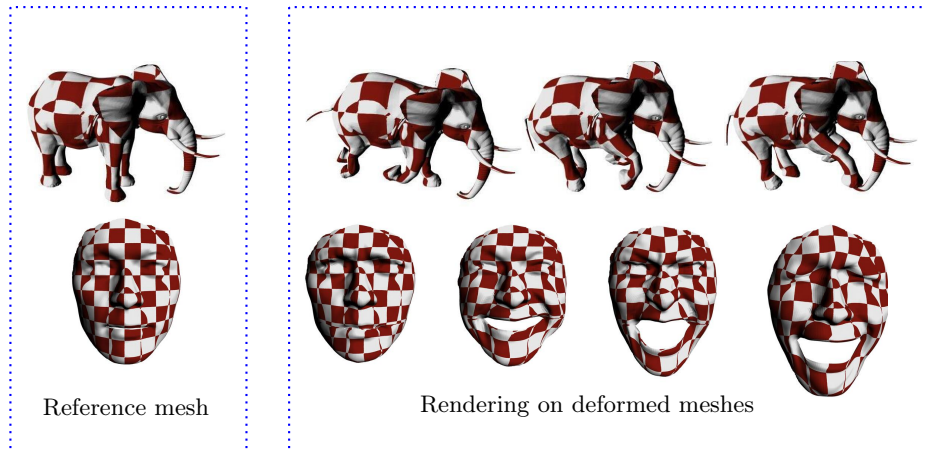

Reference mesh   Rendering on deformed meshes

Fig. 6: Qualitative evaluation of texture represented by our method under mesh deformation. We train our network for a checker texture on the reference mesh (*left*) and render images for various deformations. Since the mesh topology remains unchanged under deformations, and our features are intrinsic to the mesh, our representation is unaffected by the deformation and can be evaluated similarly to the reference configuration. The results show that the representation is consistent under deformations and produces no visible artifacts.

extrinsic methods like [32,42] need to map the intersection point in the deformed configuration back to the reference configuration, which adds a slight computa-

tional overhead. In Fig. 6, we show qualitatively that a texture represented by our method is consistent when the mesh is deformed and exhibits no noticeable artifacts. For the experiments, we use a checker texture on two meshes from the data provided by Sumner and Popovic [39], which we learn directly on the respective reference mesh and evaluate on some of its deformations.

4.4 BRDF Reconstruction from Images with Calibrated Lighting

As a second application of our method, we consider the estimation of a spatially varying *bidirectional reflectance distribution function* (BRDF) on a mesh. The BRDF $f(x, l, v)$ describes how much of the irradiance incident from the light direction $l \in \mathbb{S}^2$ is reflected in the viewing direction $v \in \mathbb{S}^2$. To obtain the total outgoing radiance $L_o(x, v)$ at position x in the viewing direction v, the incoming irradiance $L_i(x, l)$ needs to be integrated over the hemisphere $\mathbb{H}$, which is known as the rendering equation

$$L_o(x, v) = \int_{\mathbb{H}} f(x, l, v) L_i(x, l) \cos \theta_l \, \mathrm{d}l. \tag{5}$$

For the estimation of the BRDF, we consider the special case of a single, directional light. As a result, the irradiance L_i is independent of the position x, and the rendering equation Eq. (5) reduces to a single evaluation for the direction l of the directional light. The simplified equation reads

$$L_o(x, v) = f(x, l, v) L_i \mathbb{I}_s(x, l) \cos \theta_l, \tag{6}$$

where we have introduced the indicator function $\mathbb{I}_s(x, l)$ to account for cast shadows.

Several models have been proposed to parametrize the BRDF. In this work, we choose the isotropic variant of the state-of-the-art Disney BRDF, which uses 12 parameters with values in the unit interval [4]. We compare against texture fields and intrinsic neural fields and modify the methods to predict these parameters spatially varying on the mesh by adjusting the output dimension accordingly. For Texture Fields, we found that an exponential learning rate scheduler with $\gamma = 0.9$ is necessary to prevent overfitting. All other methods remain unchanged. We employ a batch size of $b = 2^{14}$ color values under a given light direction for the training of all methods.

We use the established DiLiGenT-MV real-world dataset for our experiments [21]. It contains HDR images of 5 objects with complex reflective behavior. The images are taken from 20 calibrated views, captured under 96 calibrated directional lights each. Moreover, the dataset contains meshes of the objects. For training, we use 10 views with 30 lights each and evaluate the models on 20 lights for each of the remaining 10 views. We follow [29] and combine an L1 loss with a gamma correction function applied to the color values. Since the dataset contains HDR images, very bright regions would otherwise dominate the loss, and information from darker regions would be suppressed. We refer to the supplement for more details. The results in Tab. 2 and Fig. 7 show, that
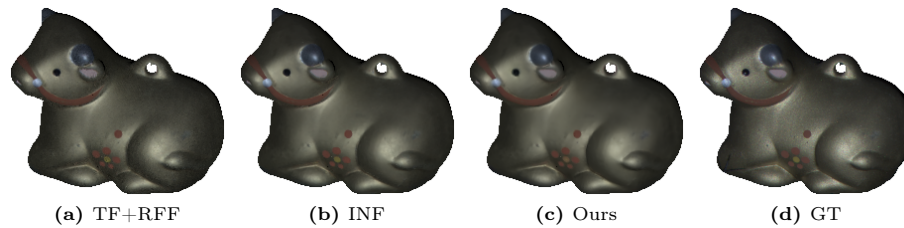
(a) TF+RFF   (b) INF   (c) Ours   (d) GT

Fig. 7: Qualitative Results of the BRDF reconstruction for the *cow* of the DiLiGenT-MV real-world dataset. The renderings show, that our method is able to reconstruct sharp boundaries in the material and yields results that are practically indistinguishable from the baseline methods, while being almost an order of magnitude faster.

Table 2: Results of the BRDF reconstruction. All metrics are averaged over the experiments for the 5 objects of the DiLiGenT-MV real-world dataset. Note that DSSIM and LPIPS are scaled by 100. Our method archives again similar reconstruction quality to the other methods while maintaining the large speed-up observed previously.

Method	PSNR ↑	DSSIM ↓	LPIPS ↓	# Params. ↓	Speedup ↑
TF+RFF	42.13	0.6718	**1.50** 🏆	**332k** 🏆	1.00x
INF	**42.21** 🏆	**0.666** 🏆	1.53 🏆	204930k	1.08x 🏆
Ours	42.17 🏆	0.6700 🏆	1.60	930k 🏆	**7.58x** 🏆

for BRDF estimation, we achieve similar reconstruction quality to the other methods while maintaining the significant speed-up in evaluation time. For more qualitative results, we refer to the supplement.

5 Conclusion

We have introduced MeshFeat as a multi-resolution parametric feature encoding for neural fields on meshes. The approach adapts the idea of parametric feature grids from the Euclidean space to meshes. Compared to existing approaches, this enables us to work with considerably smaller MLPs, leading to a significant speed-up. Our proposed multi-resolution approach is based on mesh simplification and enables the effective sharing of global feature information. This allows for higher reconstruction quality than a single-resolution approach while simultaneously requiring a smaller feature dimension. We have demonstrated the computational efficiency of our encoding for texture and BRDF representation on meshes, where we achieve reconstruction results that are on par with competing methods based on frequency-based encodings with significantly longer evaluation times. Additionally, we showcased our method's native applicability to signals on deforming meshes. We identify a more texture-adaptive multi-resolution feature approach as a promising direction for future research. We hope our work provides valuable insights applicable to various areas, including virtual and augmented reality, animations, and computer graphics engines.

References

1. Akenine-Möller, T., Haines, E., Hoffman, N., Pesce, A., Iwanicki, M., Hillaire, S.: Real-Time Rendering 4th Edition. A K Peters/CRC Press (2018)
2. Baatz, H., Granskog, J., Papas, M., Rousselle, F., Novák, J.: Nerf-tex: Neural reflectance field textures. In: Comput. Graph. Forum (2022)
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: ICML (2020)
4. Burley, B., Studios, W.D.A.: Physically-based shading at disney. In: Acm Siggraph (2012)
5. Chabra, R., Lenssen, J.E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., Newcombe, R.: Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In: ECCV (2020)
6. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: CVPR (2019). https://doi.org/10.1109/CVPR.2019.00609
7. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR (2022)
8. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: SIGGRAPH (1997). https://doi.org/10.1145/258734.258849
9. Garland, M., Heckbert, P.S.: Simplifying surfaces with color and texture using quadric error metrics. In: IEEE Visualization Conference (1998). https://doi.org/10.1109/VISUAL.1998.745312
10. Grattarola, D., Vandergheynst, P.: Generalised implicit neural representations. In: NeurIPS (2022)
11. Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y.: Implicit geometric regularization for learning shapes. In: ICML (2020)
12. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: Meshcnn: a network with an edge. ACM TOG (2019). https://doi.org/10.1145/3306346.3322959
13. Hertz, A., Perel, O., Giryes, R., Sorkine-Hornung, O., Cohen-Or, D.: SAPE: spatially-adaptive progressive encoding for neural optimization. In: NeurIPS (2021)
14. Hertz, A., Perel, O., Giryes, R., Sorkine-Hornung, O., Cohen-Or, D.: Mesh draping: Parametrization-free neural mesh transfer. Comput. Graph. Forum (2023). https://doi.org/10.1111/CGF.14721
15. Hwang, I., Kim, H., Kim, Y.M.: Text2scene: Text-driven indoor scene stylization with part-aware details. In: CVPR (2023). https://doi.org/10.1109/CVPR52729.2023.00188
16. Jiang, C.M., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T.: Local implicit grid representations for 3d scenes. In: CVPR (2020)
17. Jiang, Z., Schneider, T., Zorin, D., Panozzo, D.: Bijective projection in a shell. ACM TOG (2020). https://doi.org/10.1145/3414685.3417769
18. Kim, H., Jang, Y., Lee, J., Ahn, S.: Hybrid neural representations for spherical data. CoRR (2024). https://doi.org/10.48550/ARXIV.2402.05965
19. Koestler, L., Grittner, D., Möller, M., Cremers, D., Lähner, Z.: Intrinsic neural fields: Learning functions on manifolds. In: ECCV (2022). https://doi.org/10.1007/978-3-031-20086-1_36

20. Lee, A.W.F., Sweldens, W., Schröder, P., Cowsar, L.C., Dobkin, D.P.: MAPS: multiresolution adaptive parameterization of surfaces. In: SIGGRAPH (1998). https://doi.org/10.1145/280814.280828

21. Li, M., Zhou, Z., Wu, Z., Shi, B., Diao, C., Tan, P.: Multi-view photometric stereo: A robust solution and benchmark dataset for spatially varying isotropic materials. IEEE TIP (2020). https://doi.org/10.1109/TIP.2020.2968818

22. Li, Z., Müller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H.: Neuralangelo: High-fidelity neural surface reconstruction. In: CVPR (2023)

23. Lin, S., Zhou, B., Zheng, Z., Zhang, H., Liu, Y.: Leveraging intrinsic properties for non-rigid garment alignment. In: ICCV (2023). https://doi.org/10.1109/ICCV51070.2023.01332

24. Liu, H.D., Zhang, J.E., Ben-Chen, M., Jacobson, A.: Surface multigrid via intrinsic prolongation. ACM TOG (2021). https://doi.org/10.1145/3450626.3459768

25. Liu, L., Gu, J., Lin, K.Z., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. NeurIPS (2020)

26. Mehta, I., Gharbi, M., Barnes, C., Shechtman, E., Ramamoorthi, R., Chandraker, M.: Modulated periodic activations for generalizable local functional representations. In: ICCV (2021)

27. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: CVPR (2019)

28. Michel, O., Bar-On, R., Liu, R., Benaim, S., Hanocka, R.: Text2mesh: Text-driven neural stylization for meshes. In: CVPR (2022). https://doi.org/10.1109/CVPR52688.2022.01313

29. Mildenhall, B., Hedman, P., Martin-Brualla, R., Srinivasan, P.P., Barron, J.T.: Nerf in the dark: High dynamic range view synthesis from noisy raw images. In: CVPR (2022). https://doi.org/10.1109/CVPR52688.2022.01571

30. 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: ECCV (2020)

31. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM TOG (2022). https://doi.org/10.1145/3528223.3530127

32. Oechsle, M., Mescheder, L.M., Niemeyer, M., Strauss, T., Geiger, A.: Texture fields: Learning texture representations in function space. In: ICCV (2019). https://doi.org/10.1109/ICCV.2019.00463

33. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: CVPR (2019)

34. Peng, S., Niemeyer, M., Mescheder, L.M., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: ECCV (2020). https://doi.org/10.1007/978-3-030-58580-8_31

35. Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., Courville, A.: On the spectral bias of neural networks. In: ICML (2019)

36. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. NIPS (2007)

37. Sharp, N., Crane, K.: A Laplacian for Nonmanifold Triangle Meshes. Comput. Graph. Forum (2020)

38. Sitzmann, V., Martel, J.N.P., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions. In: NeurIPS (2020)

39. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. ACM TOG (2004)

40. Sun, C., Sun, M., Chen, H.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In: CVPR (2022)
41. Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C., Nowrouzezahrai, D., Jacobson, A., McGuire, M., Fidler, S.: Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In: CVPR (2021)
42. Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. NeurIPS (2020)
43. Tang, D., Dou, M., Lincoln, P., Davidson, P., Guo, K., Taylor, J., Fanello, S., Keskin, C., Kowdle, A., Bouaziz, S., Izadi, S., Tagliasacchi, A.: Real-time compression and streaming of 4d performances. In: ACM TOG (2018)
44. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NIPS (2017)
45. Walker, T., Mariotti, O., Vaxman, A., Bilen, H.: Explicit neural surfaces: Learning continuous geometry with deformation fields. CoRR (2023). https://doi.org/10.48550/ARXIV.2306.02956
46. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In: NeurIPS (2021)
47. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE TIP (2004). https://doi.org/10.1109/TIP.2003.819861
48. Xiang, F., Xu, Z., Hasan, M., Hold-Geoffroy, Y., Sunkavalli, K., Su, H.: Neutex: Neural texture mapping for volumetric neural rendering. In: CVPR (2021). https://doi.org/10.1109/CVPR46437.2021.00704
49. Yang, B., Bao, C., Zeng, J., Bao, H., Zhang, Y., Cui, Z., Zhang, G.: Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In: ECCV (2022). https://doi.org/10.1007/978-3-031-19787-1_34
50. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. In: NeurIPS (2021)
51. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018). https://doi.org/10.1109/CVPR.2018.00068

Appendix

In this supplementary material, we give additional details on the multi-resolution strategy in Appendix A, on the training in Appendix B, as well as additional experimental results in Appendix C. The latter includes additional renderings for all objects as well as quantitative results for the individual objects of the BRDF reconstruction. Moreover, we present a qualitative analysis of the influence of the different resolutions of our multi-resolution approach in Appendix C.1 and further analysis of the hyperparameters in Appendix C.3.

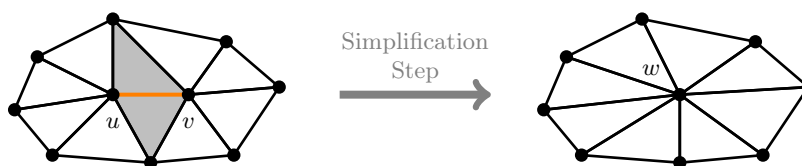## A   Mesh Simplification-Based Multi-Resolution Strategy

Fig. 8: Visualization of a single simplification step. The orange edge has been chosen by the algorithm to be contracted. The two adjacent vertices u and v are collapsed into the vertex w. The two gray triangles are removed in this process. Figure adapted from [8].

As described in the main text, we use the mesh simplification algorithm by Garland and Heckbert [8] to construct our multi-resolution approach. The algorithm iteratively contracts vertex pairs until a specified target number of vertices is reached. The pairs to be contracted are selected based on a geometric error, as shown in the original work. See Fig. 8 for a visualization of a single contraction step on an edge.

By using this simplification algorithm, we obtain the sequence of meshes $((V^{(i)}, F^{(i)}))_i$ with the specified target resolutions. Recall that the multi-resolution strategy is then based on the mapping

$$m^{(i)} : V \to V^{(i)} \quad v \mapsto m^{(i)}(v) = v^{(i)} \tag{7}$$

(Eq. (1) in the main text) that assigns a vertex $v \in V$ in the original to mesh the vertex $v^{(i)} \in V^{(i)}$ in resolution i to which v was collapsed in the decimation algorithm. Consider Fig. 8 as an example and assume that the mesh on the left is the original resolution and the mesh on the right is the i-th resolution. In that case, $m^{(i)}$ would map both vertices u and v to w, i.e. $m^{(i)}(u) = w$ and $m^{(i)}(v) = w$. The idea works analogously for multiple simplification steps between the resolutions.

We interpolate the features *in the original resolution*. To do so, we aggregate the features of the different resolutions based on the mapping in Eq. (7): For each

vertex in the original resolution, we query to which vertex in the coarser resolutions it was collapsed and retrieve the respective features. To obtain the final multi-resolution feature for this vertex, we sum the features from all resolutions according to Eq. (2) in the main text. Note that we do *not* compute a geometric mapping between the resolutions but only use the connectivity information yielded by the mesh simplification algorithm and contained in the mapping. The interpolation of the features is performed based on the Barycentric coordinates of the respective triangle in the original mesh, see Eq. (3) in the main text.

B   Additional Training Details

B.1   Training Details for Texture Reconstruction

For the texture reconstruction experiments, we employ the same dataset as in [19,32], which includes the mesh and multiple views of the cat and human object. We use the same views as done in [19] to make the comparison as direct as possible. These contain a set of 5 training, 100 validation and 200 test 512x512 views. The training images can be seen in Fig. 10.

B.2   DiLiGenT-MV Dataset

For the experiments on the BRDF reconstruction, we use the DiLiGenT-MV real-world dataset, which contains HDR images of 5 objects, taken from calibrated viewpoints under calibrated lighting conditions [21]. The triangle meshes provided with the dataset contain an excessively large number of vertices, leading to a significantly prolonged computation time for the LBO eigenfunctions for INF and an increased occurrence of unsupervised vertices in our method. Therefore, we reduce the number of vertices from roughly 10^6 to about $2 \cdot 10^5$. To stay consistent with the simplified mesh, we compute the normals of the simplified mesh for the rendering rather than using the normal maps included in the dataset.

B.3   Loss for the BRDF Estimation

To avoid a dominant influence of the bright regions on the loss, we use a gamma mapping to transform the RGB values from linear to sRGB space before applying the L1 loss, as proposed by [29]. Hence, the loss formulation reads

$$\mathcal{L}_{\text{data}} = \frac{1}{N} \sum_{i=1}^{N} |\gamma(I_o(x,v)) - \gamma(I_{GT}(x,v))|, \tag{8}$$

where $I_o(x,v)$ is the rendered color and $I_{GT}(x,v)$ is the ground truth color of the pixel corresponding to x and v in linear color space. We use the following standard formula for the gamma mapping $g : [0,1] \to [0,1]$, $c_{\text{lin}} \mapsto c_{\text{sRGB}}$ described in [1]:

$$g(c_{\text{lin}}) = \begin{cases} \frac{323}{25} \, c_{\text{lin}} & \text{if } c_{\text{lin}} \leq 0.0031308 \\ \frac{211}{200} \, c_{\text{lin}}^{\frac{5}{12}} - \frac{11}{200} & \text{else} \end{cases} \tag{9}$$

C   Additional Experimental Results

C.1   Visualization of the Muli-Resolution Features

Our multi-resolution strategy enables an effective sharing of common features. Coarser resolutions can learn global features, while finer resolutions act as a correcting term for details. We visualize this, by rendering the trained model with different resolution stages deactivated. More precisely, we modify the feature gathering described in Eq. (2) in the main text, such that we do not sum the contributions over all resolutions but only over a subset of the resolutions. We start from the coarsest one and successively add finer resolutions.

Qualitative results are shown in Fig. 9. The model was trained as described in the main text with 4 resolutions $r^{(i)} \in \{1, 0.1, 0.05, 0.01\}$. The results show that, indeed, the coarser resolutions capture coarse and more global texture features that are then refined by including the finer resolutions in the feature gathering.
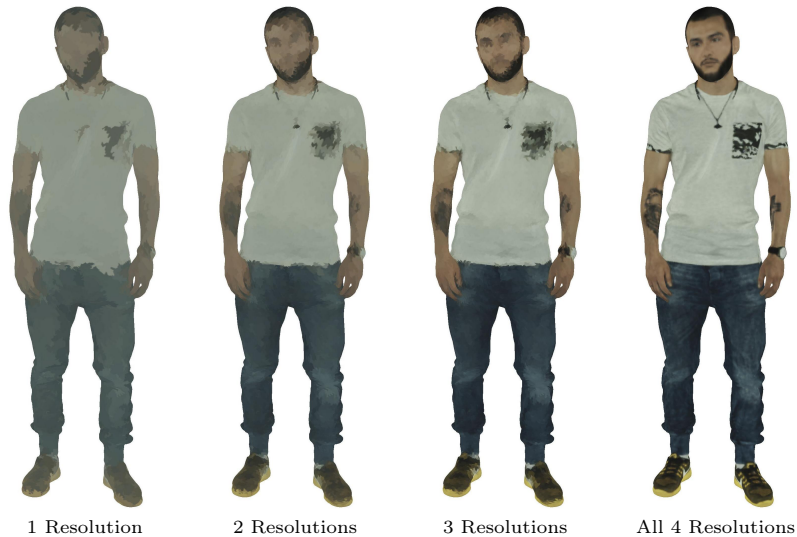

1 Resolution   2 Resolutions   3 Resolutions   All 4 Resolutions

Fig. 9: Renderings with progressive unlocking of finer resolutions. The model was trained as detailed in the main text with 4 resolution levels $r^{(i)} \in \{1, 0.1, 0.05, 0.01\}$. From left to right we successively include finer resolutions in our feature gathering (Eq. (2) in the main text), starting from only the coarsest resolution. The results show that our multi-resolution approach works as expected, and coarser resolutions (*left*) capture global features, while finer resolutions (*right*) enhance details in our representation.

C.2 Additional Results for the Texture Reconstruction

We show additional qualitative comparisons between the methods in Fig. 11. Our method yields high-quality reconstruction results on par with the other methods while admitting the significantly faster inference described in the main paper. In Fig. 12, we present additional views for our method, showing that we consistently achieve a good reconstruction on all mesh parts.
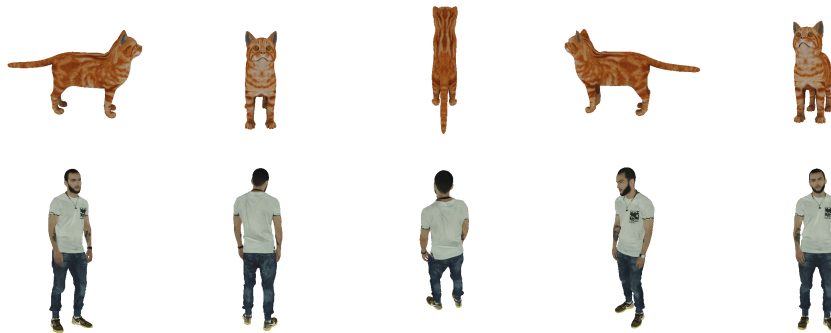

Fig. 10: Training views for the cat and human dataset used for the texture reconstruction experiments. Note that we use the same views as done in the training by [19].

C.3 Further analysis on the choice of hyperparameters

Table 3: Reconstruction quality with different numbers of layers and hidden dimensions. We observe a slight decrease in reconstruction quality after decreasing the hidden dimension or the number of layers even further. For a single layer, the results are slightly worse, even for a significantly increased hidden dimension.

Hidden Layers	Hidden dimension	PSNR↑	DSSIM ↓	LPIPS ↓
2	16	32.31	0.209	0.417
2	32	**32.51**	**0.202**	0.400
1	32	32.39	0.205	0.402
1	64	32.42	0.206	**0.395**
1	128	32.39	0.210	0.398

MLP Parameters One of the main reasons for our significant speedup is the very shallow MLP with 2 hidden layers with a dimension of 32. Since the latent features contain the spatial information of the texture, the MLP only needs to decode them into RGB values. In Tab. 3, we analyze the effect of reducing the MLP size even further. However, reducing the hidden dimension or the number of layers yields diminishing returns.
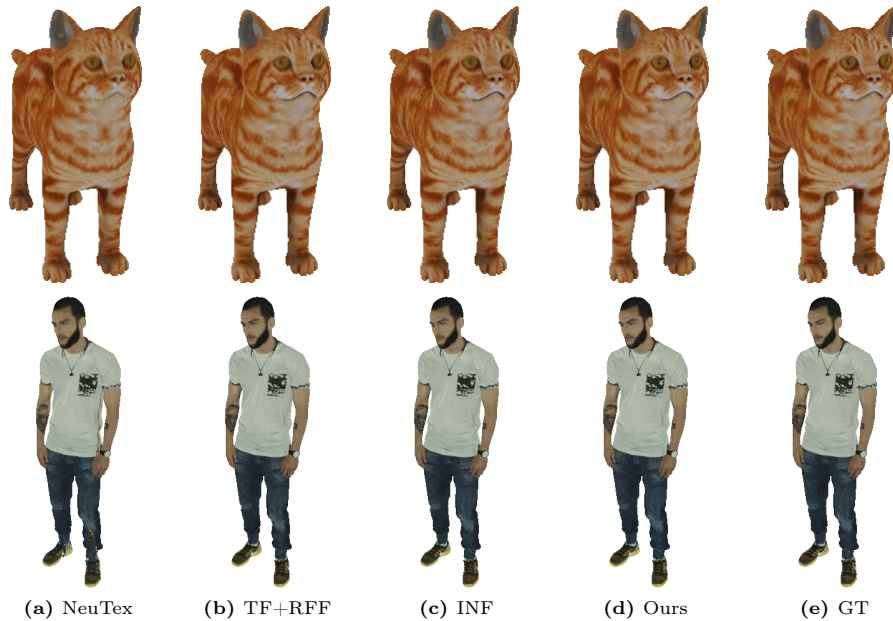
| (a) NeuTex | (b) TF+RFF | (c) INF | (d) Ours | (e) GT |

Fig. 11: Further qualitative comparisons on unseen views to the baseline methods [19,32,48]. We produce results that are on par with the state-of-the-art while providing our notable speedup. Note that we use $d = 10$ for the latent codes of the cat object in this figure.

Encoding dimensions The dimension d of our features is the main factor for the model size, and therefore a small dimension is desirable. However, the model size needs to be balanced with the reconstruction quality. Tab. 4 shows the reconstruction quality over a large range of encoding dimensions. We observe only a slight influence of the encoding dimension on the reconstruction quality, with a slight indication of overfitting for increasing the dimension. For very low dimensions, the results show a significant drop, indicating that the model is unable to reconstruct the texture faithfully, given too few features to store the information.

Resolution configurations Our method leverages a mesh simplification algorithm [8, 9] to obtain different mesh resolutions, which are then used for the multi-resolution approach. It is apparent through Tab. 5 that the method is robust to different resolution configurations as long as the finest resolution is the original resolution, *i.e.* $r^{(1)} = 1$.

C.4 Absolute inference speed

To gain more insight into the inference speedup, we provide absolute values for inference speed in Tab. 6.
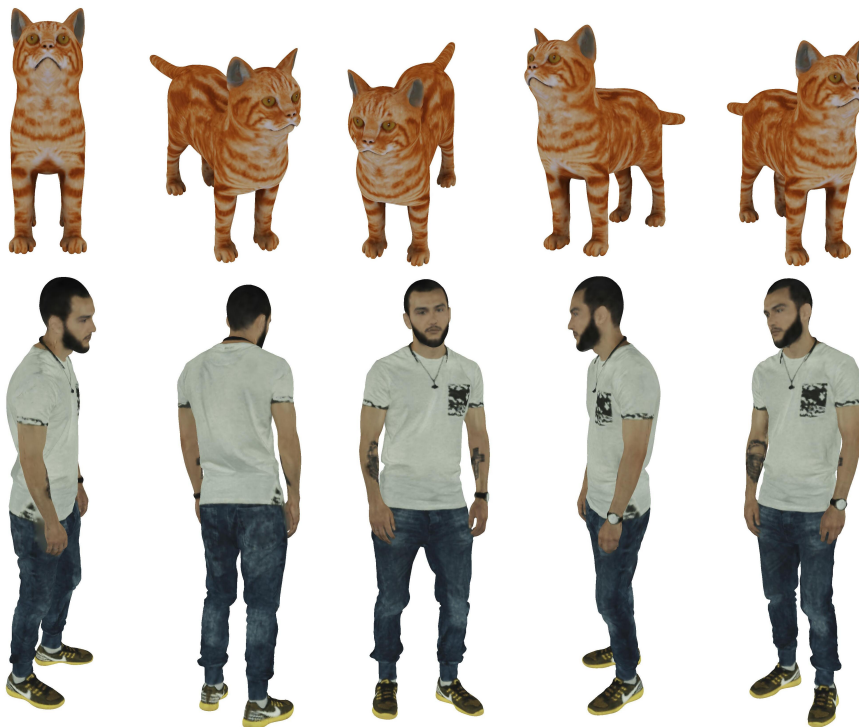
Fig. 12: Further renderings for unseen views for the cat and human object using Mesh-Feat. We see, that our method enables good reconstruction on all parts of the mesh.

C.5   Qualitative comparison to a single resolution approach

Sec. 4.2 in the main text demonstrates how a single-resolution setup struggles to achieve a good reconstruction. Fig. 13 shows that the results of the single-resolution show noisy areas despite using the regularizer. This might indicate that the multi-resolution enables the regularizer to act more efficiently since it increases the area of influence through the coarse resolutions.

C.6   Additional Results for the BRDF Estimation

For completeness, we show the quantitative results for all 5 objects of the DiLiGenT-MV dataset individually in Tab. 7. The results confirm, that we achieve results that are consistently of comparable reconstruction quality to the other methods while achieving a significant inference speed-up. Qualitative results in the form of a single view per object are presented in Fig. 14. We see that for all objects in the dataset, the results of our method are hardly distinguishable from those of the other methods.

Table 4: Influence of the encoding dimensions d for texture reconstruction on the human object. While the reconstruction quality is fairly similar over a large range of dimensions, we observe a slight decrease for increasing values of d, which might indicate the tendency of overfitting. Also, we see a significant drop at the low end of the table, which shows that a minimum number of encoding dimensions is required to achieve high-quality reconstructions.

Latent code dim (d)	PSNR ↑	DSSIM ↓	LPIPS ↓
1	29.30	0.3209	0.8058
2	32.28	0.2134	0.4182
3	32.21	0.2166	0.4095
4	32.51	0.2019	0.3962
5	32.42	**0.2001**	0.4053
6	**32.65**	0.2003	0.3984
7	32.43	0.2022	**0.3342**
8	32.46	0.2003	0.4051
9	32.51	0.2014	0.4045
10	32.43	0.2043	0.4035
11	32.43	0.2001	0.4113
12	32.48	0.2060	0.4015
13	32.48	0.2039	0.4191
14	32.43	0.2023	0.4050
15	32.47	0.2036	0.3991

Table 5: Influence of the choice of resolutions $r^{(i)}$ on the reconstruction quality. We observe a fairly stable reconstruction quality for different resolution combinations, as long as the finest resolution $r^{(1)} = 1$ is included.

Used Resolutions	PSNR ↑	DSSIM ↓	LPIPS ↓
{1, 0.1, 0.05, 0.01}	**32.51**	**0.2019**	0.3962
{1, 0.1, 0.01}	32.44	0.2024	0.4010
{1, 0.5, 0.25, 0.125}	32.35	0.2118	0.3899
{1, 0.25, 0.0625, 0.0625}	32.38	0.2061	**0.3874**
{0.75, 0.25, 0.075, 0.025}	31.51	0.2728	0.4811
{0.9, 0.12, 0.05, 0.01}	31.99	0.2301	0.4303

Table 6: Absolute inference speed in milliseconds for texture representation on the human object. Our time measurement includes a GPU warmup over 10 steps. The reported absolute inference time is the time taken for a forward pass for each method averaged over 300 repetitions.

Method	human	cat
Neutex	14.738ms	14.453ms
TF+RFF	7.120ms	7.090ms
INF	5.000ms	4.994ms
Ours d=4	1.144ms	1.021ms
Ours d=10	1.409ms	1.381ms

Multiresolution   Single resolution   Single resolution   Single resolution
$(d = 4)$ $(d = 4)$ $(d = 6)$ $(d = 10)$
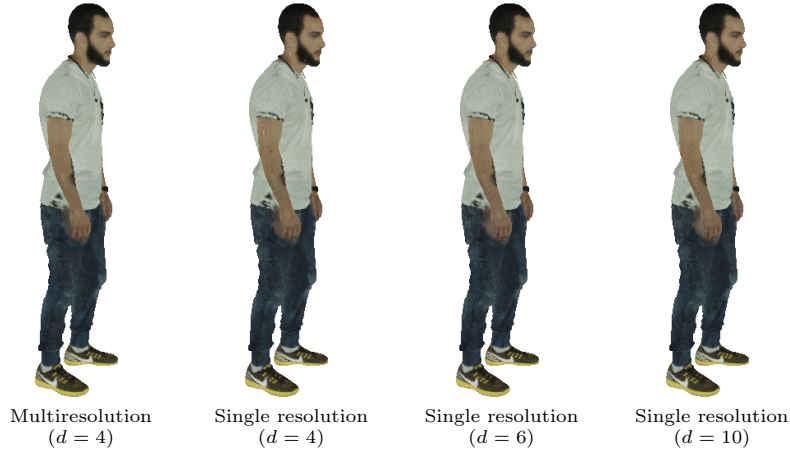
Fig. 13: Qualitative comparison between the multi-resolution approach and reconstructions using features over only a single resolution (the original mesh resolution). The single-resolution approach leads to more noisy reconstructions despite the increased dimension (d) of the feature vectors.

Table 7: Quantitative results of the BRDF reconstruction for all five objects of the DiLiGenT-MV dataset. Note that DSSIM and LPIPS are scaled by 100. The results show, that our method yields reconstruction quality that is on par with the other methods while achieving a significant speed-up for the inference.

Object	Method	PSNR ↑	DSSIM ↓	LPIPS ↓	# Params. ↓	Speedup ↑
bear	TF+RFF	43.68	0.4826	**0.9227** 🏆	**332k** 🏆	1.00x
	INF	**43.78** 🏆	**0.4772** 🏆	0.9730 🏆	204931k	1.08x 🏆
	Ours	43.73 🏆	0.4778 🏆	0.9913	930k 🏆	**7.78x** 🏆
buddha	TF+RFF	37.03 🏆	**1.0785** 🏆	**2.0095** 🏆	**332k** 🏆	1.00x
	INF	37.02	1.0904	2.0393 🏆	204929k	1.08x 🏆
	Ours	**37.04** 🏆	1.0862 🏆	2.0569	930k 🏆	**7.35x** 🏆
cow	TF+RFF	47.22	0.3318	**1.1384** 🏆	**332k** 🏆	1.00x
	INF	**47.31** 🏆	**0.3299** 🏆	1.2184 🏆	204931k	1.08x 🏆
	Ours	47.29 🏆	0.3300 🏆	1.4572	930k 🏆	**7.62x** 🏆
pot2	TF+RFF	46.69	0.4581	0.9326 🏆	**332k** 🏆	1.00x
	INF	**46.81** 🏆	**0.4485** 🏆	**0.9317** 🏆	204927k	1.08x 🏆
	Ours	46.80 🏆	0.4518 🏆	0.9802	930k 🏆	**7.72x** 🏆
reading	TF+RFF	36.02 🏆	1.0079	2.5034 🏆	**332k** 🏆	1.00x
	INF	**36.14** 🏆	**0.9832** 🏆	**2.4690** 🏆	204929k	1.08x 🏆
	Ours	36.02	1.0041 🏆	2.5352	930k 🏆	**7.44x** 🏆

Fig. 14: Qualitative Results for the BRDF reconstruction for all objects of the DiLiGenT-MV dataset. The results of our method are practically indistinguishable from the results of the other methods.