Supplementary Material for 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

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

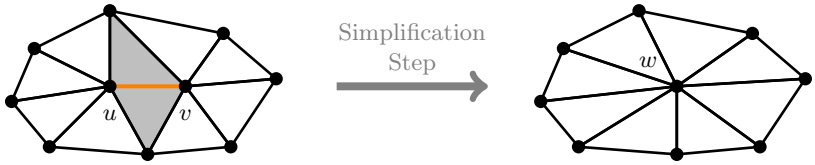

Fig. 1: Visualization of a single simplification step. The orange edge has been chosen by the algorithm to be contracted. The two adjacent vertices u and v are collapsed into the vertex w. The two gray triangles are removed in this process. Figure adapted from [2].

As described in the main text, we use the mesh simplification algorithm by Garland and Heckbert [2] 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. 1 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

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

* Equal contribution

(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. 1 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. (1): 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 [4, 7], which includes the mesh and multiple views of the cat and human object. We use the same views as done in [4] 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. 3.

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 [5]. 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 [6]. Hence, the loss formulation reads

$$\mathcal{L}_{\text{data}} = \frac{1}{N} \sum_{i=1}^{N} |\gamma(I_o(x, v)) - \gamma(I_{GT}(x, v)|, \tag{2}$$

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]:

$$g(c_{\text{lin}}) = \begin{cases} \frac{323}{25}\ c_{\text{lin}} & \text{if } c_{\text{lin}} \leq 0.0031308 \\ \frac{211}{200}\ c_{\text{lin}}^{\frac{5}{12}} - \frac{11}{200} & \text{else} \end{cases} \tag{3}$$

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. 2. 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.

C.2 Additional Results for the Texture Reconstruction

We show additional qualitative comparisons between the methods in Fig. 4. 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. 5, we present additional views for our method, showing that we consistently achieve a good reconstruction on all mesh parts.

C.3 Further analysis on the choice of hyperparameters

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. 1, we analyze the effect of reducing the MLP size even further. However, reducing the hidden dimension or the number of layers yields diminishing returns.

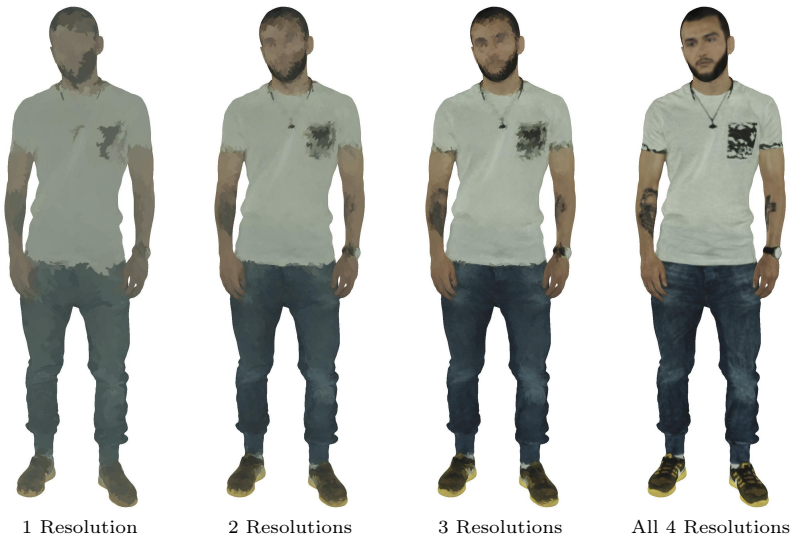

Fig. 2: Renderings with progressive unlocking of finer resolutions. The model was trained as detailed in the main text with 4 resolution levels $r^{(i)} \in \{1, 0.1, 0.05, 0.01\}$. From left to right we successively include finer resolutions in our feature gathering (Eq. (2) in the main text), starting 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.

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. 2 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 [2, 3] to obtain different mesh resolutions, which are then used for the multi-resolution approach. It is apparent through Tab. 3 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. 4.

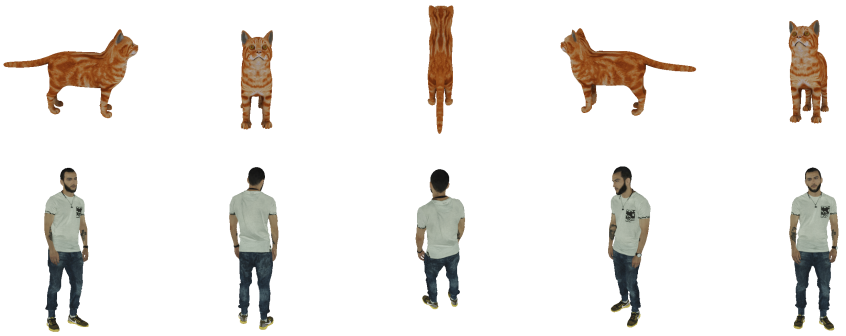

Fig. 3: Training views for the cat and human dataset used for the texture reconstruction experiments. Note that we use the same views as done in the training by [4].

Table 1: 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

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. 6 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. 5. 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. 7. We see that for all objects in the dataset, the results of our method are hardly distinguishable from those of the other methods.

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

Fig. 4: Further qualitative comparisons on unseen views to the baseline methods [4, 7, 8]. 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.

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. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: SIGGRAPH (1997). https://doi.org/10.1145/258734.258849
3. 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
4. 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
5. 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
6. 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
7. 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

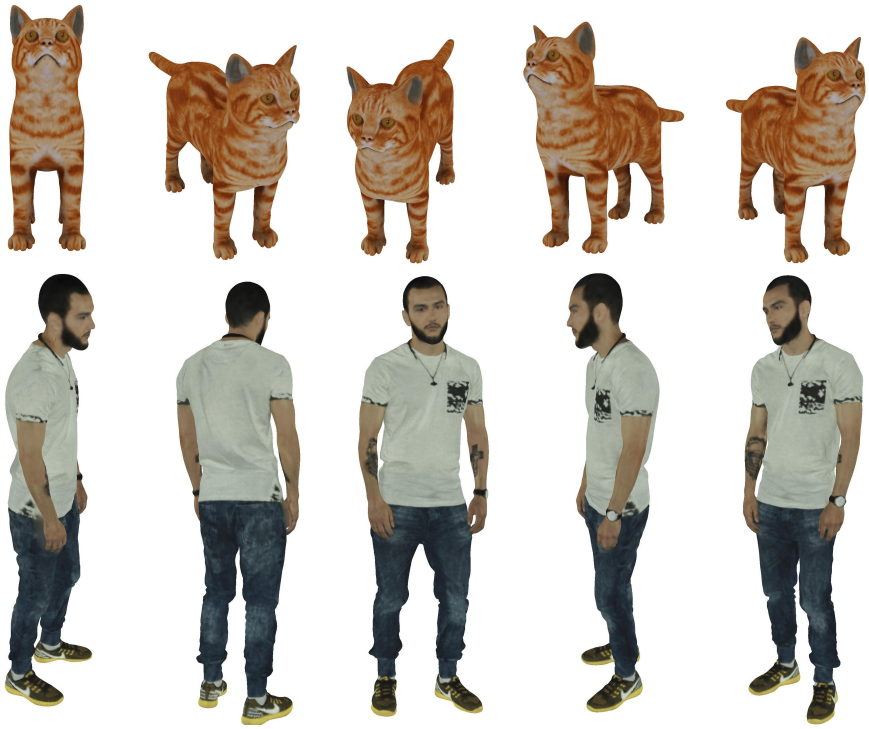

Fig. 5: Further renderings for unseen views for the cat and human object using MeshFeat. We see, that our method enables good reconstruction on all parts of the mesh.

8. 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

Table 2: 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 3: 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 4: 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

Fig. 6: 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 5: 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. 7: 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.