Explicit Neural Surfaces: Learning Continuous Geometry with Deformation Fields

Thomas Walker T.M.Walker@sms.ed.ac.uk
Octave Mariotti omariott@ed.ac.uk
Amir Vaxman avaxman@ed.ac.uk
Hakan Bilen h.bilen@ed.ac.uk
University Of Edinburgh

Editors: Sophia Sanborn, Christian Shewmake, Simone Azeglio, Nina Miolane

Abstract

We introduce Explicit Neural Surfaces (ENS), an efficient smooth surface representation that directly encodes topology with a deformation field from a known base domain. We apply this representation to reconstruct explicit surfaces from multiple views, where we use a series of neural deformation fields to progressively transform the base domain into a target shape. By using meshes as discrete surface proxies, we train the deformation fields through efficient differentiable rasterization. Using a fixed base domain allows us to have Laplace-Beltrami eigenfunctions as an intrinsic positional encoding alongside standard extrinsic Fourier features, with which our approach can capture fine surface details. Compared to implicit surfaces, ENS trains faster and has several orders of magnitude faster inference times. The explicit nature of our approach also allows higher-quality mesh extraction whilst maintaining competitive surface reconstruction performance and real-time capabilities.

Keywords: 3D vision, neural surfaces, explicit, intrinsic geometry, reconstruction, real-time, inverse rendering

1. Introduction

Reconstructing 3D objects from multiple-view images is a fundamental problem in computer vision and graphics. Classical 3D reconstruction methods (Hartley and Zisserman, 2003; Furukawa and Ponce, 2009; Agarwal et al., 2011) follow a multi-stage pipeline, match either handcrafted or learned features across images, and then recover their 3D coordinates through mesh generation. Their success heavily relies on correctly identifying matching features from different images, where accumulated matching errors cannot be easily fixed in the later stage. A recent promising paradigm is based on analysis-by-synthesis, learning neural representations for 3D such that their rendering matches the image collection.

Successful methods in this group use continuous volumetric representations (Mildenhall et al., 2020; Wang et al., 2021; Yariv et al., 2021, 2020) and render them in a differentiable manner. Neural radiance fields (NeRFs) (Mildenhall et al., 2020) and its variants represent geometry using density fields obtained from volume rendering. However, density-based geometries have undefined surface boundaries and therefore lead to noisy extracted surfaces. To mitigate this problem, recent techniques (Wang et al., 2021; Yariv et al., 2021) use an implicit signed distance function within this volume rendering framework to recover exact geometries as level sets. Despite their good performance in surface reconstruction, these methods have at least two important shortcomings. First, they are slow to train, render views,

and extract meshes from, since using an implicit function necessitates expensive sampling strategies to locate the surface.

Second, in order to extract meshes, implicit surfaces require a post-processing step such as marching cubes (Lorensen and Cline, 1987). However, this exposes such methods to reconstruction artifacts, generates an excessive number of elements, and produces low-quality meshing with grid aliasing. Whilst not being directly reflected in surface reconstruction performance, these drawbacks prevent neural implicit surfaces from being integrated into real-time geometric pipelines, and can be problematic for finite-element-methods (FEM) which are very sensitive to element quality (Shewchuk, 2002). A possible mitigation is to explicitly learn a mesh through differentiable rasterization (Thies et al., 2019; Worchel et al., 2022). Such methods achieve promising speed/performance trade-offs and can be integrated into real-time mesh-based graphics engines which are widely used. However, the flexibility and reconstruction quality are limited by a fixed discrete mesh and heavily rely on the presence of strong regularization techniques (Nicolet et al., 2021).

Motivated by these shortcomings, we introduce an *explicit* neural surface (ENS) representation based on a neural deformation field that maps an initial continuous domain (e.g., the unit sphere) into a target surface in a progressive coarse-to-fine strategy. The representation is trained through sampling discrete meshes, that are efficiently rendered by a jointly learned neural deferred shader. Our approach comprises three unique components. First, our deformation field is smooth, and is decoupled from domain mesh resolution and connectivity. Secondly, we benefit from our explicit geometry by extracting shape-aware *intrinsic* positional encodings to improve rendering and reconstruction performance. Specifically, we use a hybrid positional encoding mixing intrinsic Laplace-Beltrami eigenfunctions with extrinsic Fourier features. Finally, we sequentially combine neural deformation fields with progressively higher-frequency positional encodings, enabling gradual surface refinement through both coarse and then fine deformation fields. The resulting method attains an *explicitly* defined neural surface which demonstrates superior reconstruction performance to prior explicit methods while being a fully continuous representation. Compared to the implicit neural surfaces, our method is significantly faster at inference time while being competitive in reconstruction performance. In addition, our explicit surface definition allows for direct surface sampling, enabling higher-quality intrinsic remeshing across resolutions and with arbitrary connectivity, without requiring post-processing techniques or strong regularization.

2. Related Work

Neural Surface Representations Neural networks in 3D reconstruction have recently gained significant popularity for their ability to model continuous geometry. Numerous methods successfully represent surfaces as level sets of signed distance functions (SDF) (Wang et al., 2021; Yariv et al., 2020, 2021; Oechsle et al., 2021). An *implicit* surface is appealing as it can model arbitrary topological changes. NeuS and other implicit surface-based variants (Wang et al., 2021; Yariv et al., 2021; Oechsle et al., 2021) have been successfully integrated into volume rendering pipelines. However, implicit surfaces cannot be directly sampled, and therefore necessitate expensive volumetric procedures to train, render, and mesh. Concurrently, PermutoSDF (Rosu and Behnke, 2023) used learned spatial hash-encodings

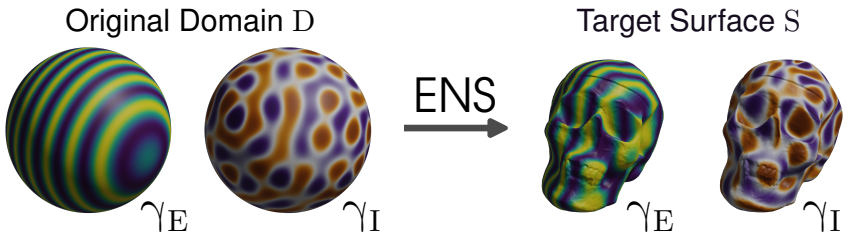


Figure 1: ENS uses both an extrinsic positional encoding γ_E and an intrinsic positional encoding γ_I of the original domain. By combining ambient as well as shape-aware surface embeddings we can attain fine surface details and textures.

to facilitate shallower MLPs and greatly accelerate training times. However, expensive spatial queries are still necessary to render images and extract meshes. A separate line of literature proposed learning an atlas of neural parametric surfaces (Groueix et al., 2018; Williams et al., 2018; Bednarik et al., 2019) for 3D reconstruction. This surface definition can facilitate sampling surfaces directly by virtue of a parameter space, however, using multiple disjoint mappings requires stitching together surface patches (softly), leading to poor-quality reconstructions. In contrast, our method enforces an initial topology as a prior and thus guarantees watertight surfaces, explicitly represented by a deformation field.

Neural Deferred Shading Deferred shading is a real-time rendering approach based on meshes, in which the shading component is performed entirely in screen-space (Deering et al., 1988). The scene is rasterized to create a screen-space map containing geometric information, and then processed by a shader to predict pixel-wise RGB values. In a recent work (Munkberg et al., 2022), shade meshes were extracted from an implicit SDF with Deep Marching Tetrahedra (DMTet) (Shen et al., 2021). While the deferred shading is faster relative to volume rendering, the need to use DMTet as a mesh-extraction step still limits the efficiency of their approach. Furthermore, the use of shaders that cannot handle arbitrary lighting limits their application to settings with fixed global illumination. Another approach (Thies et al., 2019) proposed fully parameterizing a deferred shader using a neural network, and hence synthesizing novel views with arbitrary illumination and materials. Worchel et al. (2022) (NDS) subsequently proposed to jointly optimize the mesh with the neural shader, and attain an efficient multi-view surface reconstruction pipeline. In our work, we extend this further to attain the benefits of neural surface representations.

3. Method

Our objective is to learn a continuous surface representation S of a specified object in a scene. As input, we consider N images, along with their corresponding cameras, $\mathcal{I} = \{I_i, C_i\}_{i=1}^N$ and binary masks $M = \{m_i\}_{i=1}^N$ segmenting the object of interest. Our architecture and pipeline are depicted in Figure 2. We first introduce our surface representation, and then our training procedure.

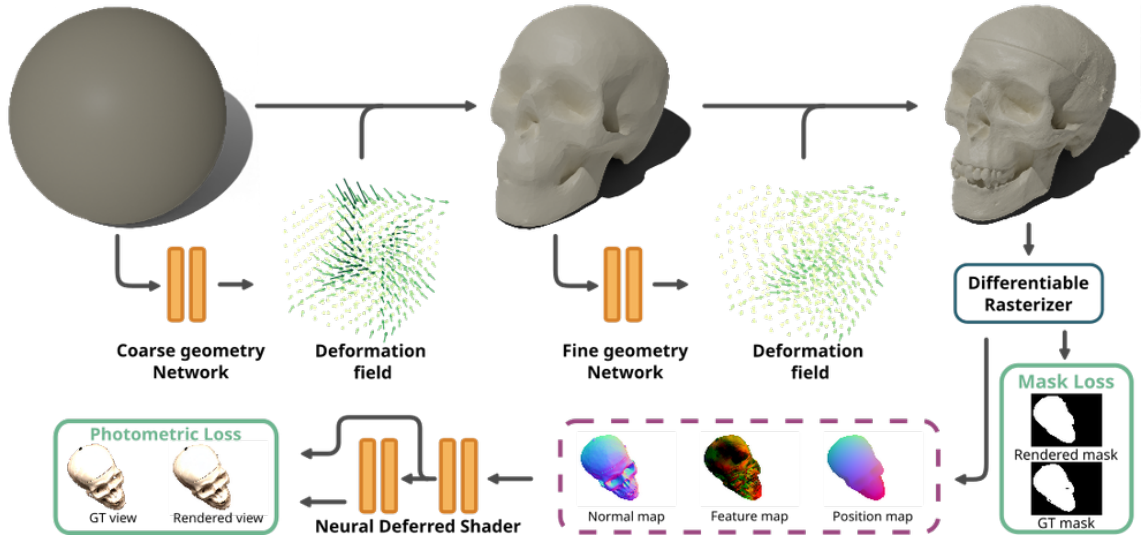


Figure 2: The fixed base domain D (here the unit sphere) is transformed by a low-frequency (top middle) and then a high-frequency neural deformation field, f_{coarse} and f_{fine} respectively, to create a surface (top-right skull). The final mesh is obtained by pullback to a meshing of the unit sphere. During training, such a mesh is rasterized and processed through a neural shader to produce a candidate image $\tilde{I}_i$ that is compared against a GT image I_i.

3.1. Neural Deformation Field

We compute an ambient mapping $f : \mathbb{R}^3 \to \mathbb{R}^3$. This is used to *explicitly* define a continuous surface S when applied to a known and fixed continuous input domain $D \subset \mathbb{R}^3$, *i.e.* $S = \{f(\mathbf{x}) | \mathbf{x} \in D\}$. We parameterize f by a deformation neural field $f_\theta(\mathbf{x})$, with parameters θ. We use meshes $\mathcal{M}_D = \{V_D, E, F\}$ defined on D, that are deformed into compatible meshes $\mathcal{M}_S = \{V_S, E, F\}$ on S. They are then used for training f, where we employ fast rasterization and deferred shading, in order to produce predicted images $\tilde{I}_i$ (Section 3.4). This representation effectively decouples the geometry of S, determined only by the network parameters θ, from any specific discrete mesh $\mathcal{M}$. Thus, it allows us to freely construct (or refine) any mesh $\mathcal{M}_D$ for the purpose of training f. Extracting a mesh during training and inference is extremely fast, using only a single forward pass of the deformation network, while the continuity of the mapping allows for meshes of any chosen size or connectivity (Figure 4). Targeting S with spherical topology, we set D to be the unit sphere, from which sampling can be done in simple closed form. Throughout training we sample meshes $\mathcal{M}_D$ (and consequently $\mathcal{M}_S$) of increased vertex density; please see appendix for details.

3.2. Intrinsic/Extrinsic Surface Embeddings

Due to the low-frequency spectral bias of neural networks on low-dimensional inputs (Tancik et al., 2020; Mildenhall et al., 2020), we use positional encodings of the input domain D to

learn fine surface deformations. Standard implicit representations (Mildenhall et al., 2020; Wang et al., 2021; Yariv et al., 2021) encode positional coordinates with respect to a spectral basis defined on the ambient Euclidean space. Since D is a known base domain in our case, we can augment the extrinsic encoding with an *intrinsic* spectral basis (as the Laplacian eigenfunctions Lévy (2006); Koestler et al. (2022)) of the input domain D. We approximate the continuous eigenfunctions of D by a specific fine mesh M_D with piecewise linear functions. For this mesh, we construct the cotan Laplacian W, and the Voronoi mass matrix A, and extract the eigenbasis $\{\phi_i\}$ that solves $W\phi_i = \lambda_i A\phi_i$ for eigenvalues λ_i. Following Koestler et al. (2022); Rustamov (2007), we define an intrinsic embedding $\gamma_I : D \to \mathbb{R}^d$ as

$$\gamma_I(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_d(\mathbf{x})], \tag{1}$$

where $\phi_1, \ldots, \phi_d$ are the lowest d eigenfunctions on D. As we demonstrate in Figure 5 and Figure 6, the deformation network benefits from having an intrinsic encoding to learn fine surface details and produce higher-quality renders. However, using solely intrinsic information only captures the metric information of the surface, and would prevent the deformation network from having awareness of location of D in the scene. For this reason we use a hybrid embedding $\gamma_H = [\gamma_I, \gamma_E]$, combining both intrinsic γ_I and extrinsic positional embeddings γ_E based on random Fourier feature (RFF) encoding $\gamma_E : \mathbb{R}^3 \to \mathbb{R}^d$ defined as

$$\gamma_E(\mathbf{x}) = [\cos(\mathbf{b}_1^\top \mathbf{x}), \sin(\mathbf{b}_1^\top \mathbf{x}), \ldots, \cos(\mathbf{b}_{d/2}^\top \mathbf{x}), \sin(\mathbf{b}_{d/2}^\top \mathbf{x})], \tag{2}$$

where coefficients $\mathbf{b}_i \in \mathbb{R}^3$ are sampled randomly from a multivariate Gaussian distribution. By controlling the standard deviation σ, one can control the distribution of basis function frequencies to target low or high-frequency deformations.

3.3. Coarse-To-Fine Optimization

Surface fitting can fall into bad minima in the early stages of training (Nicolet et al., 2021). This makes it problematic to bias deformation fields towards fine details, as it introduces high-entropy noise into the optimization process. To address this, we construct a coarse-to-fine optimization approach by decomposing the deformation field into a composition of two mappings. The first, f_{coarse}, comprises a low-frequency extrinsic RFF encoding γ_E to learn a correctly centered general shape. The second, f_{fine}, uses a high-frequency hybrid encoding $\gamma_H = [\gamma_E, \gamma_I]$ to learn fine details:

$$f(\mathbf{x}) = f_{\text{fine}} \circ \gamma_H \circ f_{\text{coarse}} \circ \gamma_E(\mathbf{x}). \tag{3}$$

By removing the onus of learning low frequencies from f_{fine}, we obtain a stable coarse-to-fine optimization process. While we could potentially use more functions in the composition, we found the two-stage deformation is sufficient to capture surface detail in our scenes.

3.4. Neural Deferred Shader

Similar to NDS (Worchel et al., 2022), we render images using the real-time graphics approach known as *deferred shading*. For a mesh $\mathcal{M}_S$ predicted by the deformation field, given a camera C_i, we rasterize and shade it into an image $\tilde{I}_i$ that is compared against the ground-truth

image I_i. We diverge from NDS, which considers strictly mesh data for shading, to benefit from our neural deformation field $f_\theta(\mathbf{x})$. Namely, we extending it to produce a feature vector $z_\theta(\mathbf{x})$ alongside deformed vertex locations, such that we output $(f(\mathbf{x}), z(\mathbf{x}))$. This is in direct analogy to the use of learned feature vectors in implicit SDF methods (Wang et al., 2021; Yariv et al., 2020). As argued by Yariv et al. (2020), this feature vector could learn to encode global effects such as shadows and secondary light reflections during training. We define the differentiable rasterizer,

$$r(V_S, z(V_D), C_i) = (x_i, z_i, n_i, \tilde{m}_i), \tag{4}$$

to generate a 2D image with per-pixel linearly-interpolated surface positions x_i, normals n_i, predicted masks $\tilde{m}_i$, and features z_i. Finally, this is fed to the *neural shader* $h_\sigma(x_i, n_i, z_i, C_i)$—a small learned MLP with parameters σ—that predicts per-pixel RGB values to render a final predicted image $\tilde{I}_i$. The use of a learned feature z facilitates fast and accurate colour learning. This is likely in-part due to its increased representation ability, but also because the deformation network benefits from the hybrid positional encoding $\gamma_H(\mathbf{x})$, allowing it learn high-frequency feature vectors z from intrinsic information. However, as discussed in a concurrent work by Wu et al. (2022), the use of feature vectors $z(\mathbf{x})$ to aid rendering can disturb normal-color dependency, overshadowing the learning of the geometry. For this reason we compute photometric losses with two shaders, a feature-based shader h_z and a geometry-based shader h_g which doesn't receive z. We refer the reader to Appendix A for implementation details.

3.5. Loss function

As our objective function we use a combination of a photometric L1 loss L_c, a mask-based loss L_m, and a normal regularization loss L_n computed on the mesh $\mathcal{M}_S$,

$$\min_{\theta,\sigma} \lambda_c L_c(\mathcal{I}; \theta, \sigma) + \lambda_m L_m(M_S; \theta, \sigma) + \lambda_n L_n(S; \theta), \tag{5}$$

with hyperparameters $\lambda_c, \lambda_m, \lambda_n \in \mathbb{R}$.

Mesh Regularization The low-frequency bias of neural networks has been successfully used as a geometric smoothness prior (Williams et al., 2018). However, the positional encoding, promoting higher-frequencies, can counter these advantages and result in noisy surfaces. For this reason, we use a normal-smoothness loss on $M_S = (V_S, E, F)$ to induce an underlying smooth f. Following Worchel et al. (2022), we place a soft constraint on the consistency of face normals $\mathbf{n} \in S^2$ using a cosine similarity,

$$L_n = \frac{\lambda_n}{|E|} \sum_{(i,j)\in E} (1 - \mathbf{n}_i \cdot \mathbf{n}_j)^2, \tag{6}$$

where (i, j) are the indices of the left and right faces of each edge in E. Note that, unlike the fixed-mesh method NDS, we do not use a Laplacian loss on the mesh (to promote triangle regularity), since eventually, M_S is just a proxy to learning f, where we, in fact, want to allow for triangles to deform sufficiently to fit narrow regions of the target geometry.

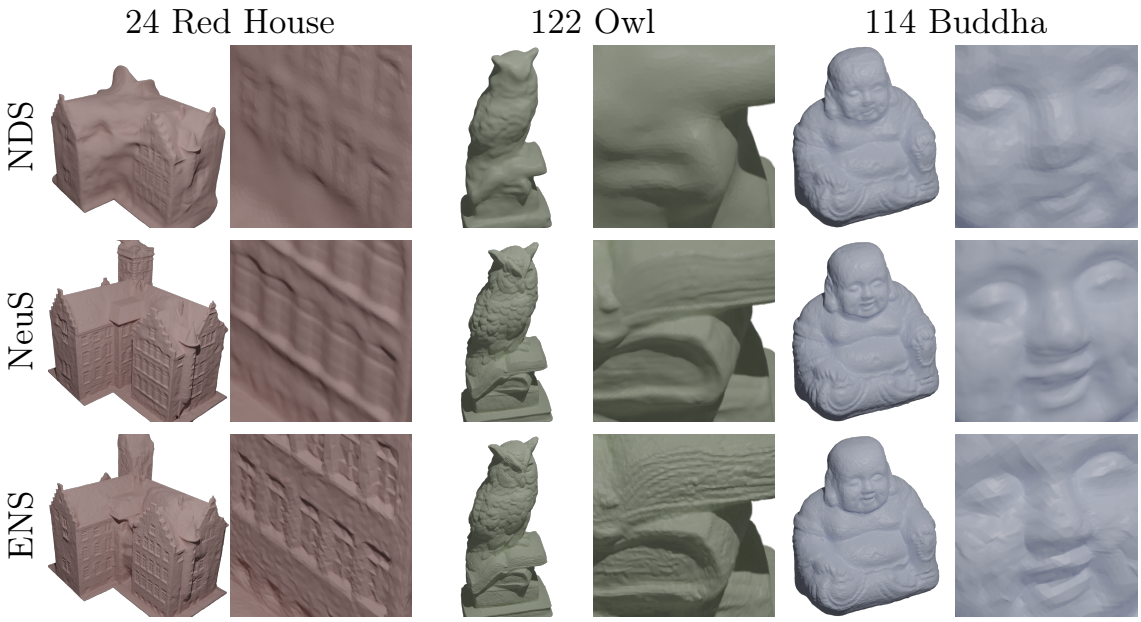


Figure 3: Qualitative evaluation for multi-view surface reconstruction. We compare our method ENS to the state-of-the-art implicit and explicit methods in three diverse scenes of the DTU dataset.

Appearance Loss In our deferred shading pipeline, predicted masks $\tilde{m}_i$ are produced by the rasterizer r before any shading. Our neural shader further produces two image predictions, $\tilde{I}_i^z$ and $\tilde{I}_i$ from the shader modules h_z and h_g respectively. With these quantities, we compute a combined photometric loss L_c and a mask loss as follows:

$$L_c = \frac{1}{|\mathcal{I}|}\sum_{i=1}^{|\mathcal{I}|} \|I_i - \tilde{I}_i^z\| + \lambda_g \|I_i - \tilde{I}_i\|, \quad L_m = \frac{1}{|\mathcal{I}|}\sum_{i=1}^{|\mathcal{I}|} \|m_i - \tilde{m}_i\|, \tag{7}$$

where λ_g controls the contribution of h_g and consequently its impact on the learned geometry. Details about L_n are in the supplementary material.

4. Experiments

For shading, we use the code provided by Worchel et al. (2022) in PyTorch (Paszke et al., 2019). For differentiable rasterization, we use the high-performance modular primitives provided by Laine et al. (2020).

4.1. Surface Reconstruction

We evaluate our approach on the task of surface reconstruction from multi-view raster images. We compare our method to the explicit NDS (Worchel et al., 2022), and two implicit surface approaches, IDR (Yariv et al., 2020), and NeuS (Wang et al., 2021). While both are

Scan	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Avg	TT	MT	RT
IDR	1.58	2.06	0.75	0.43	**1.06**	0.68	0.68	**1.38**	1.17	0.88	0.63	**0.99**	0.37	0.50	0.52	0.92	5hrs	5s	30s*
NeuS	**0.83**	**0.98**	**0.56**	**0.37**	1.13	**0.59**	**0.60**	1.45	**0.95**	**0.78**	**0.52**	1.43	**0.36**	**0.45**	**0.45**	**0.77**	5hrs	5s	30s*
NDS	4.24	5.25	1.30	0.53	1.72	1.22	1.35	1.59	2.77	1.15	1.02	3.18	0.62	1.65	0.91	1.95	**3min**	**N/A**	**5ms**
ENS	1.66	3.30	*0.88*	0.40	1.59	1.04	0.84	1.50	1.22	0.91	*0.75*	2.67	**0.36**	0.59	0.53	1.22	**5min**	**2ms**	**5ms**

Table 1: Chamfer scores (in millimeters) and average single scene times for multi-view surface reconstruction on the DTU dataset. TT: training time, MT: mesh extraction time, RT: rendering time. Results in italics use a different original topology (Appendix B.1). Results followed by an $*$ are estimations based on the author's comments.

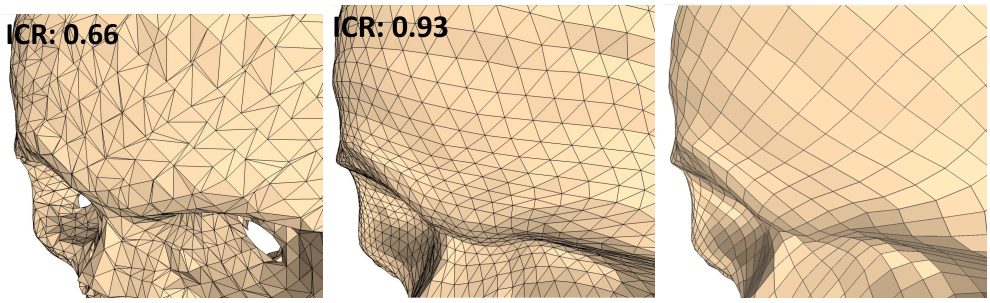


Figure 4: Implicit (left, from NeuS), vs. explicit (ours) extracted meshes of comparable resolutions. The marching cubes artifacts are evident visually in unwanted holes and aliasing, as well as in the quality measures (ICR, where 1 is ideal). Our method can generate any mesh by pullback from the domain, here exemplified for both triangle (middle) and quad (right), and achieve higher quality elements.

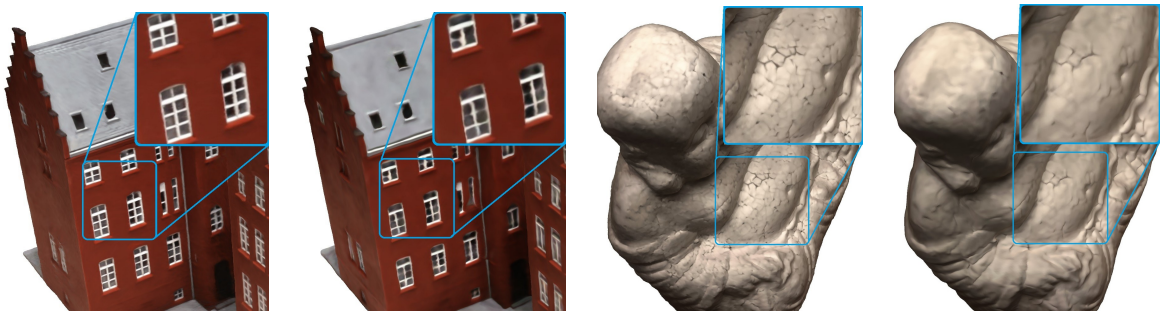
SDF-based, IDR is more closely related to ours in that it uses surface rather than volume rendering.

Dataset We use 15 scenes from the DTU dataset (Aanæs et al., 2016) for comparison. Each scene contains 1600 x1200 resolution images, each paired with camera parameters and binary masks, from 49 or 64 poses. Collectively, the scenes are challenging for reconstruction algorithms as they contain non-Lambertian materials, high-frequency geometry, inconsistent lighting, and geometry/texture ambiguities. We follow the official DTU evaluation in surface mode to generate Chamfer-L1 scores in comparison to ground truth point clouds (Table 1).

ENS significantly outperforms NDS, the benchmark explicit approach, across all scenes, while maintaining the extremely efficient training, rendering and mesh extraction. On sphere topology, we approach state-of-the-art reconstruction quality in a fraction of the time, and attain a preferable *explicit* neural representation from which meshes can be directly extracted and rendered in real-time. In Figure 3, we present reconstructions from our approach compared to NDS and NeuS. Compared to NeuS, our coarse-to-fine hybrid encoding enables us to capture high-frequency details that are not present in the compared methods. For scan 24, one of the more challenging tasks, where NDS fails, our method successfully recovers a surface and successfully learns sharp, high-frequency features.

		Scene 65			Scene 118			Scene 114		
		NeuS	ENS	ENS*	NeuS	ENS	ENS*	NeuS	ENS	ENS*
ICR	Average (↑)	0.66	0.84	**0.95**	0.65	0.72	**0.92**	0.65	0.79	**0.94**
ICR	% < 0.10 (↓)	5.80	0.27	**0.05**	6.03	2.28	**0.15**	5.95	0.95	**0.11**
ICR	% < 0.25 (↓)	13.74	1.15	**0.16**	14.40	7.79	**0.89**	14.41	5.10	**0.66**
ICR	% < 0.90 (↓)	83.20	49.39	**14.49**	82.99	65.29	**24.84**	83.01	53.88	**18.32**
CD	Average (↓)	**0.59**	1.04	1.06	**0.45**	0.59	0.63	**0.36**	**0.36**	**0.36**

Table 2: Normalized Inradius-to-circumradius ratio (ICR) quality and Chamfer Distance (CD) statistics for NeuS vs ENS (ours). For normalized ICR, 1 is ideal and 0 is degenerate. We include both ENS and a regularized ENS*, for which we optimize ICR. With this model we can extract very high quality meshes in real-time, with minimal impact on Chamfer distance.

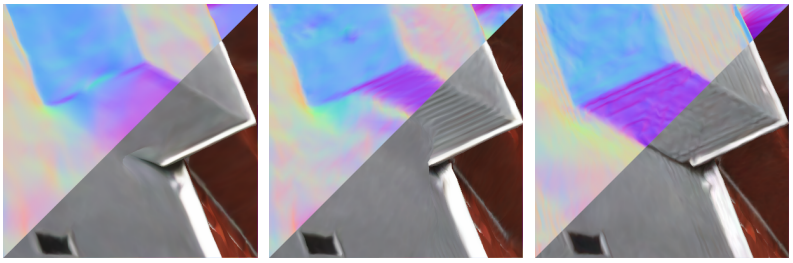


(a) with γ_I (b) without γ_I (c) with γ_I (d) without γ_I

Figure 5: Rendering ENS with and without intrinsic positional encoding γ_I. We observe improved rendering details in the models which use intrinsic positional encoding.

Mesh Quality In addition to efficiency, a major advantage of ENS is that we can attain higher-quality meshes for downstream tasks than implicit approaches, as the surface is modelled explicitly. In Figure 4, we compare meshes of $\approx$ 10K vertices extracted from NeuS and ENS, where we generate both triangle and quadrilateral meshes from f and D with just inference, as they are decoupled from any fixed mesh M. It is evident that the marching-cubes mesh exhibits grid artifacts (Sawdayee et al., 2022). See Table 2 for quality analysis. While mesh-only methods like NDS require strong regularization to converge to a desirable surface (Nicolet et al., 2021), we can inherit the regularity of our neural network as a prior to attain good solutions (Williams et al., 2018). However, unlike implicit approaches, ENS can efficiently compute mesh-based losses and back-propagate them to the continuous representation. For example, in Table 2 we present mesh quality statistics on a subset of DTU where we directly optimize the average inradius-to-circumradius (Shewchuk, 2002) of the output mesh during training, and hence attain higher quality meshing for downstream tasks. For further mesh quality analysis as well as implementation details please see Appendix C.

Ablation	CD (↓)
w/o γ_E	3.39
w/o γ_I	1.31
w/o f_{coarse}	DNC
w/o h_g	1.28
Full model	**1.22**



(a) CD (b) w/o γ_I (c) w/o h_g (d) ENS

Figure 6: Quantitative and qualitative ablation study. (a): Average Chamfer distance (CD) on the DTU dataset for different ablated models. DNC: did not converge. (b-d): Comparison of renders and normal maps on the house scene for converging ablated models.

Ablation In Figure 6 we ablate various model components by individually removing them from the full model. Without intrinsic encoding ("w/o γ_I"), the results in missing fine details such as the tiles on the roof of scan 24. In Figure 5 we further demonstrate that intrinsic encoding produces notably improved rendering quality, capturing fine structures such as window frames. Using only intrinsic encoding, ("w/o γ_E") struggles to capture coarse shape in the initial stages of training and results in significantly depreciated surface accuracy. See Appendix B for further examples. Removing the geometry-based shader ("w/o h_g") results in a model which can capture high-frequencies. However, areas with strong colour-normal dependency, such as the roofs of the house, are prone to extraneous growths with painted on textures (see Section 4.1). See Appendix B.1 for further details.

5. Limitations and Future work

The dominant limitation of our method is the same as surrounding explicit methods, in that we have to predetermine the topology of the base domain D to fit the target object in question. However, the topology is only fixed at the level of choosing D, where we can use an explicit representation of any geometry that is assumingly topologically-equivalent to the solution. An exciting future direction could be integrating explicit neural surfaces into FEA-based shape optimization pipelines, for which the high mesh-quality produces of our representation is particularly attractive for accurately solving PDEs.

6. Conclusion

We proposed an explicit neural surface (ENS) for multi-view 3D reconstruction that combines the advantages of both explicit mesh and neural representations. Our approach produces high-quality meshing, and is extremely fast to train, render and mesh while being competitive with state-of-the-art implicit methods. This representation then has the potential to be used for downstream geometry-processing tasks such as physically-based optimization (Umetani

and Bickel, 2018), computational geometric design (Chong Bao and Bangbang Yang et al., 2022) or exploring topologically consistent shape spaces (Kilian et al., 2007).

References

Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, pages 1–16, 2016.

Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10): 105–112, 2011.

Jan Bednarik, Shaifali Parashar, Erhan Gundogdu, Mathieu Salzmann, and Pascal Fua. Shape reconstruction by learning differentiable surface representations, 2019. URL https://arxiv.org/abs/1911.11227.

Jan Brandts, Sergey Korotov, and Michal Křížek. On the equivalence of regularity criteria for triangular and tetrahedral finite element partitions. *Computers & Mathematics with Applications*, 55(10):2227–2233, 2008.

Chong Bao and Bangbang Yang, Zeng Junyi, Bao Hujun, Zhang Yinda, Cui Zhaopeng, and Zhang Guofeng. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision (ECCV)*, 2022.

Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. *Acm siggraph computer graphics*, 22(4):21–30, 1988.

Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009.

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation, 2018. URL https://arxiv.org/abs/1802.05384.

Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

Martin Kilian, Niloy J Mitra, and Helmut Pottmann. Geometric modeling in shape space. In *ACM SIGGRAPH 2007 papers*, pages 64–es, 2007.

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

Lukas Koestler, Daniel Grittner, Michael Moeller, Daniel Cremers, and Zorah Lähner. Intrinsic neural fields: Learning functions on manifolds. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, pages 622–639. Springer, 2022.

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020.

Bruno Lévy. Laplace-beltrami eigenfunctions towards an algorithm that" understands" geometry. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pages 13–13. IEEE, 2006.

William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. URL https://arxiv.org/abs/2003.08934.

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8280–8290, June 2022.

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. Large steps in inverse rendering of geometry. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 40(6), December 2021. doi: 10.1145/3478513.3480501.

Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *International Conference on Computer Vision (ICCV)*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Radu Alexandru Rosu and Sven Behnke. Permutosdf: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Raif M. Rustamov. Laplace-Beltrami Eigenfunctions for Deformation Invariant Shape Representation. In Alexander Belyaev and Michael Garland, editors, *Geometry Processing*. The Eurographics Association, 2007. ISBN 978-3-905673-46-3. doi: 10.2312/SGP/SGP07/225-233.

Haim Sawdayee, Amir Vaxman, and Amit H Bermano. Orex: Object reconstruction from planner cross-sections using neural fields. *arXiv preprint arXiv:2211.12886*, 2022.

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Jonathan Richard Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *IMR*, pages 115–126, 2002.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020. URL https://arxiv.org/abs/2006.10739.

Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *Acm Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

Nobuyuki Umetani and Bernd Bickel. Learning three-dimensional flow for interactive aerodynamic design. *ACM Transactions on Graphics (TOG)*, 37(4):1–10, 2018.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction, 2021. URL https://arxiv.org/abs/2106.10689.

Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction, 2018. URL https://arxiv.org/abs/1811.10943.

Markus Worchel, Rodrigo Diaz, Weiwen Hu, Oliver Schreer, Ingo Feldmann, and Peter Eisert. Multi-view mesh reconstruction with neural deferred shading. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6187–6197, 2022.

Tong Wu, Jiaqi Wang, Xingang Pan, Xudong Xu, Christian Theobalt, Ziwei Liu, and Dahua Lin. Voxurf: Voxel-based efficient and accurate neural surface reconstruction. *arXiv preprint arXiv:2208.12697*, 2022.

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020.

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021.

7. Appendix

We provide further implementation details in Appendix A, extended ablation studies in Appendix B, and quantitative/qualitative analysis of mesh quality in Appendix C.

Appendix A. Implementation Details

Network Details For both deformation field networks $f_{\text{coarse}}, f_{\text{fine}}$ we use a single MLP with one hidden layer of 400 softplus units, which outputs a deformed vertex position $f(\mathbf{x})$, and a 128-dimensional feature vector $z(\mathbf{x})$. We use a residual connection on the vertex locations such that we have:

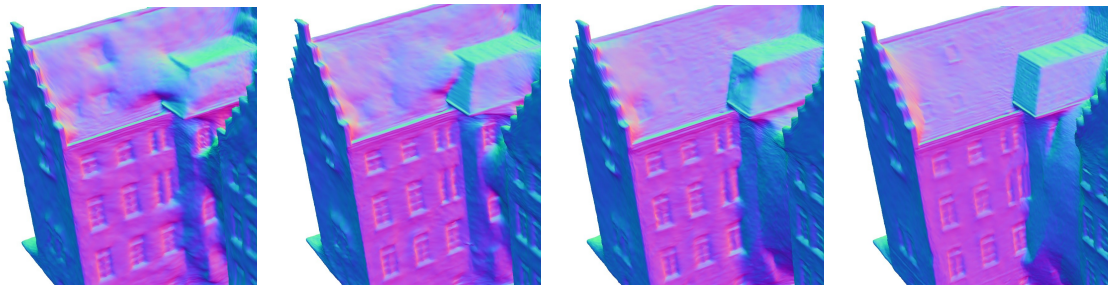
$$f(\mathbf{x}) = \mathbf{x} + \delta \text{MLP}(\mathbf{x}), \tag{8}$$

where is δ is a scaling parameter. When f_{fine} is introduced at iteration 500, this parameter is scheduled to linearly increase from 0 to 0.1 over 100 iterations (this was found to improve stability). For each neural shader h_z, h_g we use a single MLP with 3 hidden layers of 256 ReLU units.

In our experiments (see Table 2) we demonstrate that accessing a mesh during training allows us to learn a deformation field that optimizes for mesh quality alongside reconstruction accuracy. Specifically, we add an additional inradius-to-circumradius regularization term into our loss. The inradius-to-circumradius ratio is a commonly used mesh quality measure in computational geometry and finite element analysis (Shewchuk, 2002). The inradius (r) is defined as the radius of the unique incircle, while the circumradius (R) is the radius of the unique circumcircle. The normalized ratio $2r/R$ being closer to 1 indicates that the triangle is closer to an equilateral shape, which leads to more accurate numerical simulations and reduced interpolation errors. We compute a loss over the mesh by penalizing the deviation of the average triangle normalized inradius-to-circumradius from 1,

$$L_{ICR} = \frac{\lambda_{ICR}}{|F|} \sum_{i \in F} (1 - \frac{2r_i}{R_i}), \tag{9}$$

where $\lambda_{ICR} = 5 \times 10^{-3}$



(a) $\lambda_g = 0$ (b) $\lambda_g = 0.1$ (c) $\lambda_g = 0.3$ (d) $\lambda_g = 1$

Figure 7: Qualitative comparison for multi-view surface reconstruction with varying geometry shader loss coefficient λ_g.

Geometry-Based Shader While it is beneficial to learn colour variations that are uncorrelated to surface normals, diffuse materials with a strong correlation can instead induce color variations from the feature vector $z(\mathbf{x})$ rather than pushing the geometry of $f(x)$ to deform. To mitigate this, we decouple h into two components: a feature-based shader $h_z(x_i, n_i, z_i, C_i)$ predicts a base color image $\tilde{I}_i^z$, which we subsequently feed into a geometry-based shader $h_g(x_i, n_i, \tilde{I}_i^z, C_i) = \tilde{I}_i$. The key important property is that the geometry-shader *detaches* $\tilde{I}_i^z$; that is, we directly prescribe $\partial \tilde{I}_i / \partial \tilde{I}_i^z = 0$ when back-propagating gradients through the architecture. Thus, the gradients of h_g propagate through the geometry directly, and balance the independent training of z and f as illustrated in Section 4.1 and Figure 7.

Training Details For each neural network we use an ADAM (Kingma and Ba, 2014) optimizer. For the neural shaders h_z, h_g we use a learning rate of $1 \cdot 10^{-3}$, and for each deformation network we use a learning rate of $2 \cdot 10^{-3}$, which is reduced by a factor of 0.75 at the mesh refinement step. At every iteration we shade 5% of the pixels in the intersection of ground truth and predicted masks, from 6 different randomly sampled views. We set the geometry-based shader h_g loss coefficient $\lambda_g = 0.1$ for all experiments (see Appendix B). On DTU we train for 500 iterations with the coarse deformation field f_{coarse} and coarse mesh, before training for a further 1500 iterations with the full resolution mesh and both deformation fields $f_{\text{coarse}}, f_{\text{fine}}$. Note that in this second stage, f_{coarse} is "frozen" and not optimized for; in practice, we observed that optimizing both networks had no improvements to surface reconstruction, albeit increasing training times.

In total, our approach takes only ≈ 5 minutes to train on a single NVIDIA A100 40GB GPU. Rendering and mesh extraction times were measured using the maximum resolution mesh (163,842 vertices), requiring only a forward pass of shallow MLPs (taking 2–5ms).

Training Meshes We use an icosahedral mesh on the unit sphere that is subdivided to the initial coarse $\mathcal{M}_D$ with 2,562 vertices. This mesh is eventually subdivided (with new vertices being normalized to lie on the sphere) to 163,842 vertices in the second stage (alongside the introduction of f_{fine}). We found that an additional level of division (to 655,362 vertices) offered no clear improvement in Chamfer distance whilst increasing training times and memory requirements.

Positional Encodings For the the low-frequency deformation field f_{coarse} we use a random Fourier feature scale of $\sigma = 0.5$. For the high-frequency network f_{fine} we use a random Fourier features scale of $\sigma = 4$, and 2320 eigenfunctions which are pre-computed on the highest resolution input mesh (icosahedron with 163,842 vertices). Specifically, we compute the first 10,000 eigenfunctions, and use a subset of 2320 eigenfunctions. This subset comprises the eigenfunctions selected empirically in a related work (Koestler et al., 2022), as well as the last 1500 high-frequency eigenfunctions (8500-10000). For the neural shaders h_g, h_z, we use Fourier feature positional encodings γ_ω, γ_n of 4 and 3 octaves for the view directions ω and normals n respectively.

Appendix B. Extended Ablations

Intrinsic Encodings In Figure 8 we compare the renders and normal maps of the full ENS model, using a hybrid encoding γ_H of (extrinsic) random Fourier features and (intrinsic) eigenfunctions, and a model using only extrinsic positional encoding γ_E. We observe sharper

renders with γ_H, and more accurate surface details. For example, the bunny and redhouse have high-frequency textures and surface details which the extrinsic-only model γ_E struggles to resolve (see windows). In our experiments we observed that this can result in inaccurate surface deformations and lead to poor local minima.

Dual Shader Ablation In the appearance-based loss methodology (Section 3.5) of the main paper), we therefore define the photometric loss of our model as a combination of losses from the feature-based shader h_z and the geometry-based shader h_g, the latter of which is scaled by coefficient λ_g. In Figure 7 we demonstrate the effect of varying the loss coefficient λ_g on the learned geometry for scan 24 (redhouse). When using no geometry-based shader ($\lambda_g = 0$), we observe unwanted extraneous growths on the roofs where the normal-colour correlation is strong. By increasing the coefficient we clearly see the effect of the geometry shader enabling these regions to be reconstructed correctly. However, we also see this comes at the cost of less pronounced concavities. In our experiments, we found setting $\lambda_g = 0.1$ across all scenes strikes a good balance between these behaviours. nevertheless, we note this can act as a useful hyperparameter to control a prior on normal-geometry dependence.

B.1. Non-Spherical Topology

Although we use a spherical input domain for most scenes, any domain D could potentially be used. For scan 40 we use an initial torus mesh, and for scans 106 and 65 we follow Worchel et al. (2022), and generate a topologically-aligned input domain D by extracting a visual hull from the ground-truth masks $\{m_i\}$, which is then linearly subdivided throughout training. The hull is extracted by projecting a $(32 \times 32 \times 32)$ volumetric grid into image space and removing any points not contained in the masks. We then run marching cubes to extract an input domain with the correct topology, see Figure 9. We refer the interested reader to NDS Worchel et al. (2022) for more details. The initial hull is then linearly subdivided to $\approx 70K$ vertices. For these cases, we compute eigenfunctions on the subdivided visual hull.

Divergent Models In our ablation, we consider a model without the coarse-to-fine strategy and instead attempt to learn detailed models directly with a high-frequency deformation field. We remove f_{coarse} and train only f_{fine} with the highest resolution mesh. As illustrated in Figure 14, this model struggles to capture the coarse shape and quickly falls into bad minima as a result of its spectral bias.

Appendix C. Extended Remeshing and Mesh Quality Results

We next provide examples of the meshes extracted from ENS as compared to the neural implicit model NeuS (Wang et al. (2021)). We demonstrate the superior mesh quality our approach, especially at low resolutions. To illustrate the continuity of the underlying representation, we also show quad meshes extracted from ENS. Note that to get eigenfunction values on the quad mesh vertices, we interpolate them barycentrically from the original icosahedral triangle mesh.

In Figure 11 we present meshes of $\approx$ 2.5K vertices extracted from NeuS and ENS on multiple scenes. Across all of the meshes extracted from NeuS we observe aliasing and reconstruction errors such as unwanted holes and floating artifacts. At this resolution, marching cubes is prone to miss sharp structures, for example the on roofs of the redhouse

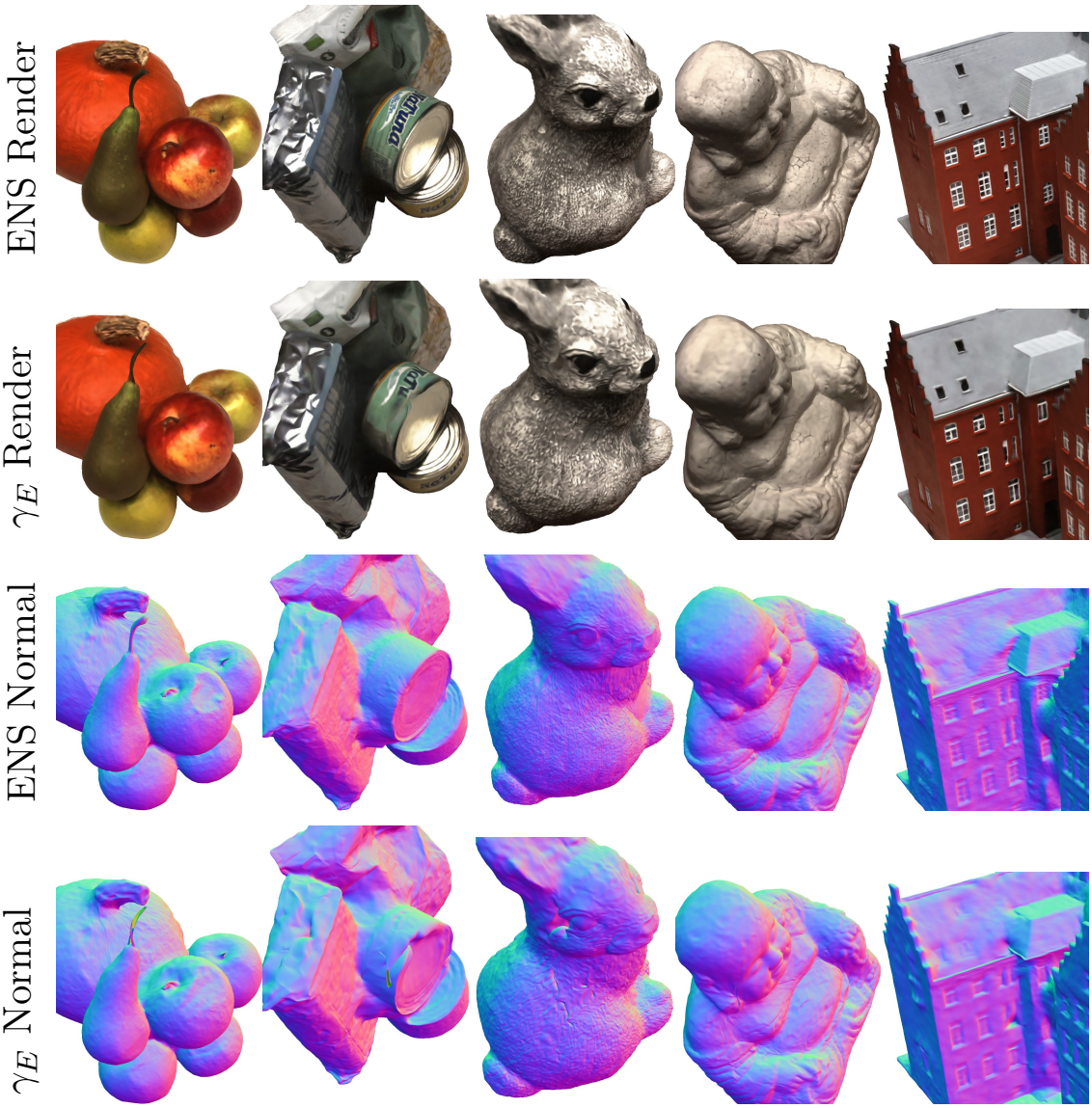


Figure 8: Qualitative comparison for multi-view surface reconstruction renders (top) and normal maps (bottom) between using the full model (ENS) with hybrid encoding γ_H and an extrinsic-only encoding γ_E, best viewed in high resolution.

(a) Iteration 0 (b) Iteration 100 (c) Iteration 2000

Figure 9: We demonstrate that a visual hull (a) can be extracted from input masks and used as the input domain with correct topology.

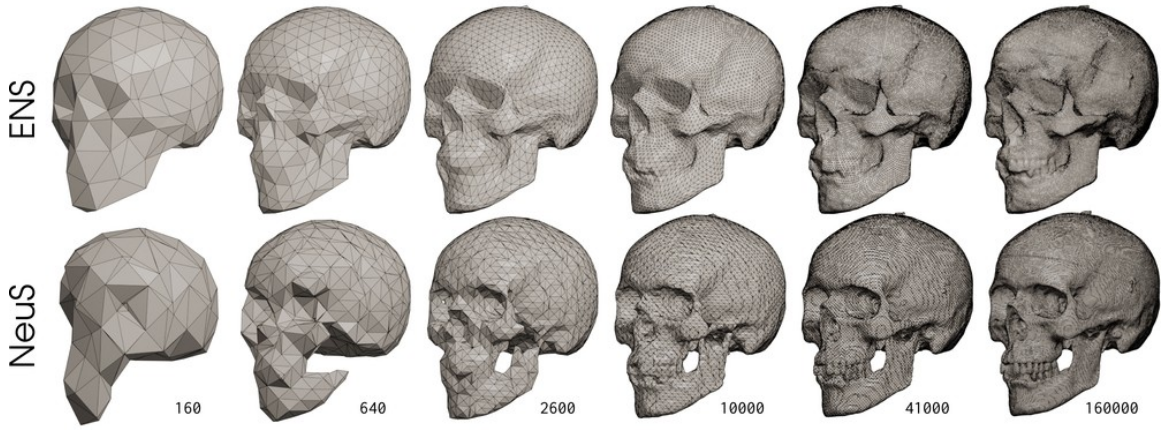


Figure 10: Meshes with varying (and top-bottom comparable) number of vertices extracted from a fully-trained model for our method (top) and NeuS Wang et al. (2021) (bottom). It is evident our approach produces topologically-consistent quality meshes without spurious holes and grid artifacts (see forehead) in any resolution.

and or the stalk the apple. In comparison, our approach produces high-quality meshes which are suitable for downstream tasks that require coarse discretization. Increasing the resolution, in Figure 12 we present meshes of $\approx$ 10K vertices extracted from each approach. At this resolution marching cubes still produces artifacts such as floaters and noticeable ridging artifacts (see redhouse) where straight edges are not aligned to the sampling grid. Our approach can maintain straight edges for any connectivity, as we can directly sample the explicit neural surface. Note that Chamfer distance can be insensitive to these types of artifacts from marching cubes, however they have large qualitative significance.

The artifacts of marching cubes (and the resulting poor face quality) is present even for high resolution meshes. In Figure 13 we measure the quality of high resolution meshes

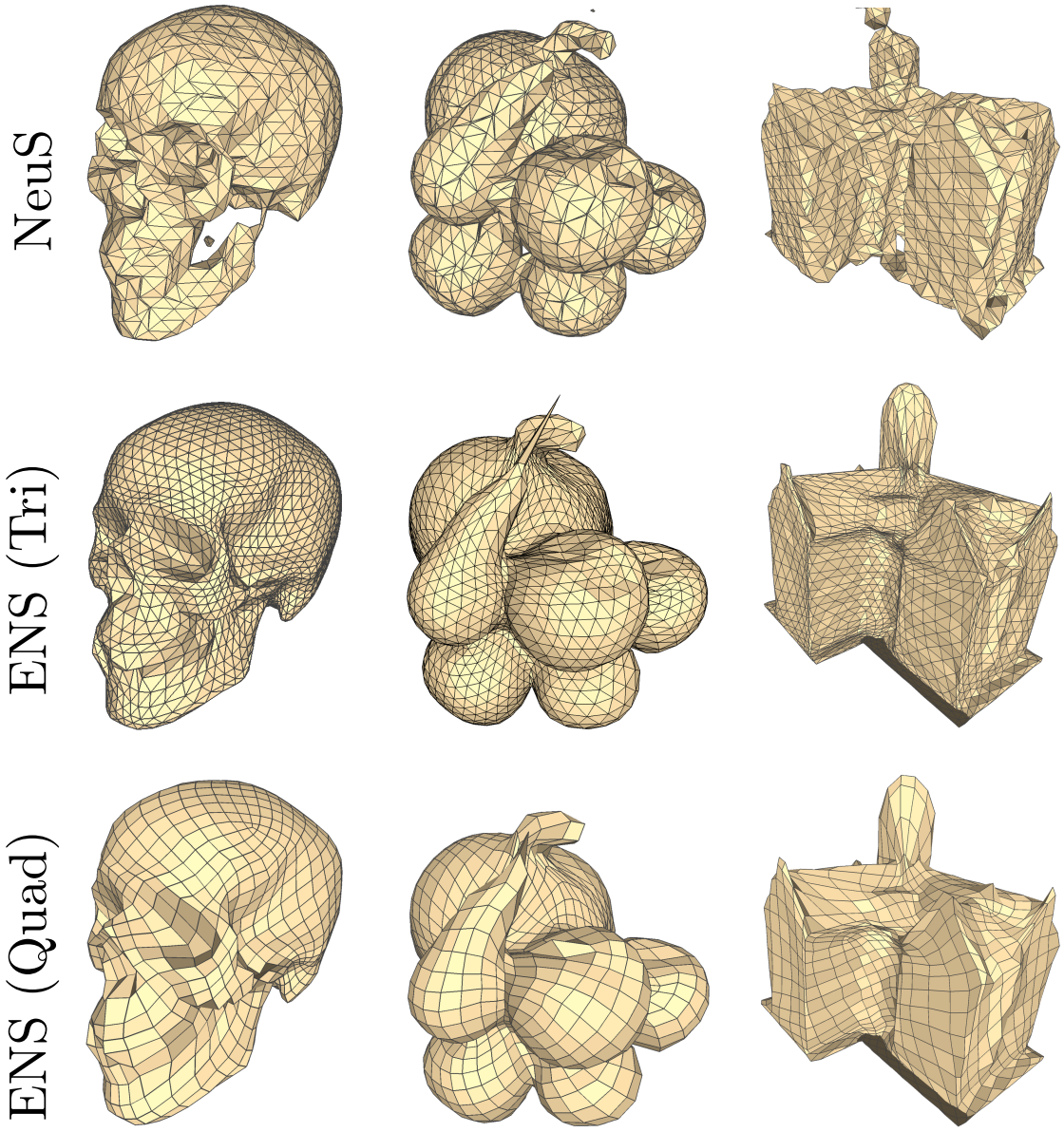


Figure 11: Meshes with 2.5K vertices extracted from NeuS compared to ENS. Note that we use a smaller quad mesh with 1.5K vertices.

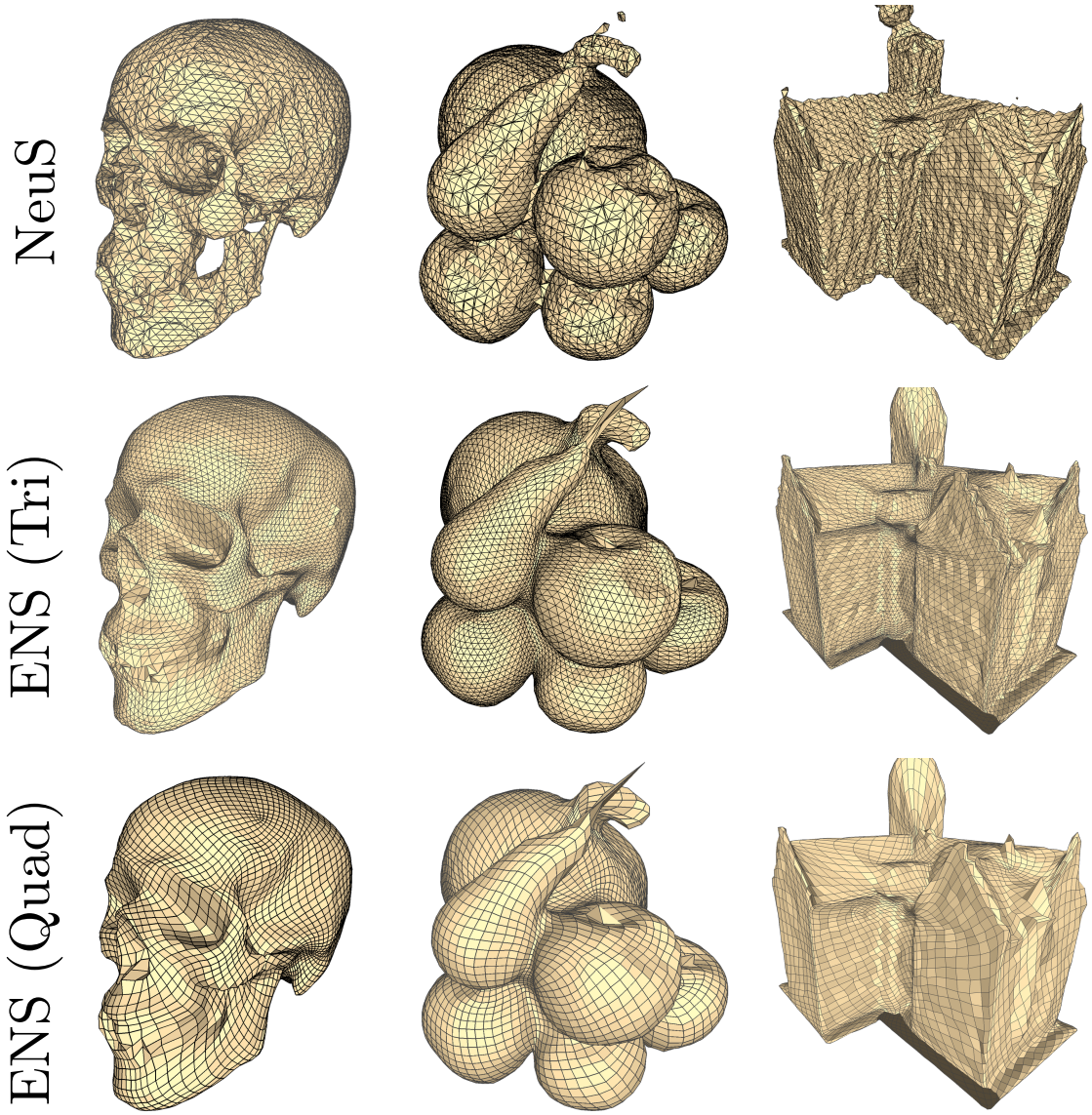


Figure 12: Meshes with 10K vertices extracted from NeuS compared to ENS. Note that we use a smaller quad mesh with 6K vertices.

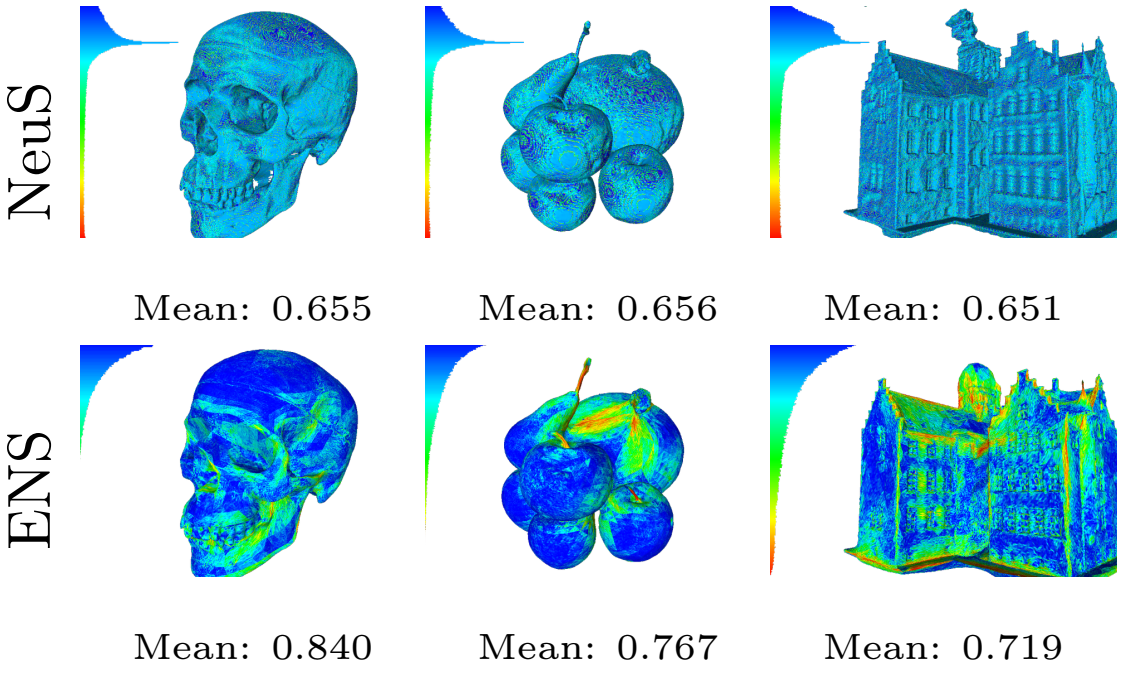
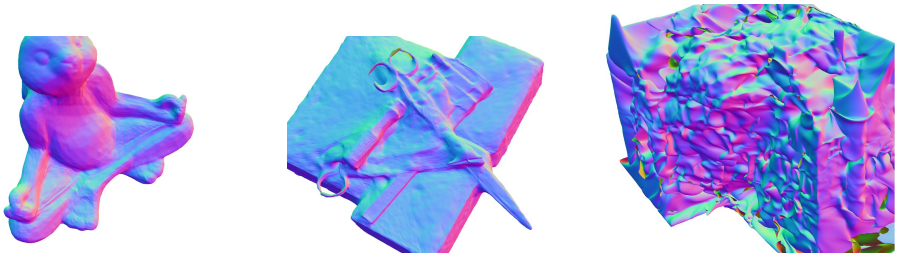


Figure 13: Mesh quality analysis of high-resolution meshes extracted from NeuS and ENS. Faces are coloured by normalized inradius to circumradius ratio, where ideal is 1 (blue) and degenerate is 0 (red).

(> 150K vertices) extracted from NeuS and ENS using the per-face inradius to circumradius ratio (Shewchuk, 2002). This metric is used to measure suitability for use in finite-element-method (FEM) applications (Brandts et al., 2008). We observe that meshes extracted from NeuS have consistently poorer face quality, as evident by the histograms and average face-quality statistics. For ENS, we note that poorer quality faces are often localized to areas of high curvature (e.g. stalks of the apples, edges of house) which require narrower fitting. For NeuS, marching cubes creates poor quality faces uniformly across the surface as a result of using a volumetric sampling grid and a fixed template of faces. When extracting meshes at low-resolution, marching cubes produces topologically inconsistent results and ridging artifacts, see Figure 10

Appendix D. Failure Cases

In Figure 14 we present failure cases of ENS, which contribute the largest source of Chamfer error in our quantitative performance. For scan 110 (Figure 14 (a)) we learn a well behaved surface, however the chest area is misplaced as a result of texture ambiguities. In Figure 14 (b) we present a failed reconstruction of scan 37. This scene has complex topology which our model struggles to approximate.



(a) 110 Rabbit (b) 37 Scissors (c) f_{fine} only

Figure 14: (a)-(b) Normal maps of failed reconstructions of scene 110 and 37 of the DTU dataset. In figure (c) we show that training divergence using only a high-frequency deformation field f_{fine}.