Neural Geometry Fields For Meshes – Supplementary Material

Venkataram Sivaram
ves223@ucsd.edu
University of California San Diego
USA

Ravi Ramamoorthi
ravir@cs.ucsd.edu
University of California San Diego
USA

Tzu-Mao Li
tzli@ucsd.edu
University of California San Diego
USA

ACM Reference Format:
Venkataram Sivaram, Ravi Ramamoorthi, and Tzu-Mao Li. 2024. Neural Geometry Fields For Meshes – Supplementary Material. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings (SIGGRAPH '24 Conference Proceedings), ??, 2024, ??.* ACM, New York, NY, USA, 8 pages. https://doi.org/??

1 RASTERIZATION

In this section, we detail how our neural geometry fields representation is well suited to modern rasterization pipelines.

Meshlet rendering. Modern rendering pipelines (e.g. Vulkan, DirectX, Metal) recently introduced mesh shader pipelines for rasterization as a general means of producing graphics primitives on the fly. Whereas traditional rasterization systems would group mesh data together in order to reduce the number of draw calls, most mesh shaders partition the mesh into clusters of triangles, *meshlets*, which can be retrieved and sent out from mesh shaders directly into rasterization processors. The advantage of this clustering process is the improvement in memory access patterns by processing meshlets that fit into smaller caches.

For procedural geometry, mesh shaders are a way to generate meshes without any unnecessary inputs to the render pipeline (as is the case for geometry and tessellation shaders). This is the approach we take to efficiently rasterizing our representation. We unpack each patch in a neural geometry field independently as its own meshlet.

We implement in this pipeline in Vulkan[1] and the `GLSL` programming language. For each patch in a neural geometry field (NGF) instance, we create a task group, which determines the number of mesh groups to launch according to the current tessellation resolution. Each mesh group generates a subregion of the vertices and triangles in a patch, and this is where the neural network is evaluated. Normal vectors can be computed by taking cross products of the vertex position with respect to the frame buffer, which lets us perform basic shading operations. Figure 2 demonstrates our implementation of this system, which runs at realtime for reasonable tessellation resolutions.

[1]Using the `VK_EXT_mesh_shader` extension.

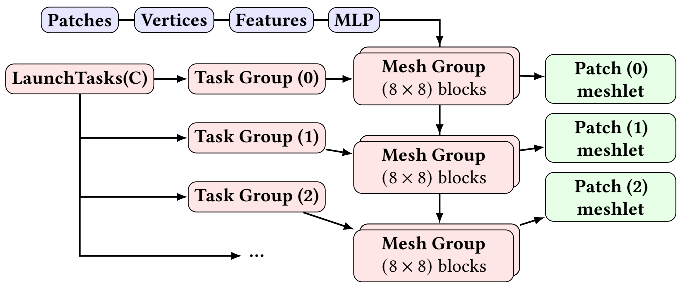


Figure 1: Meshlet rendering pipeline for neural geometry fields. We instantiate one task group for each patch in a neural geometry field, which in turn instantiates as many mesh shader work groups as needed to cover all the triangles necessary for the requested tessellation resolution. Each mesh shader group is responsible for generating a portion of the patch meshlet by computing the interpolated patch attributes, encoding and the neural network.

Discussion. Although our rudimentary implementation of NGF rasterization is already quite performant, we speculate that further optimizations can result in near sub-millisecond render times for individual instances. In particular, taking advantage of GPU shared memory and AI cores (e.g. NVIDIA's Tensor cores) would likely bring significant performance boosts to neural network evaluation times. Furthermore, the layout of the surface as a collection of disjoint patches enables culling techniques that can be run in parallel.

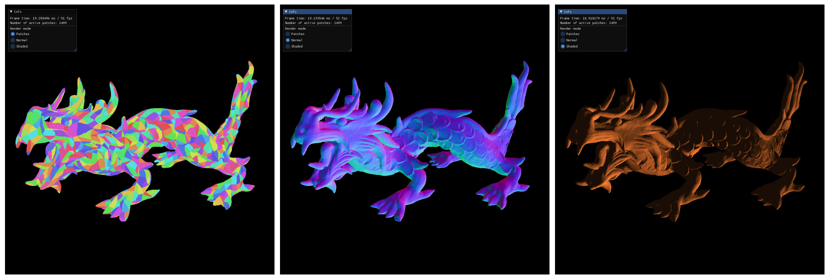
2 TEXTURE MAPPING

The lack of texture mapping capabilities for signed-distance fields is a limitation that prevents them from being widely used in production settings. Although our NGF reconstruction pipeline does not explicitly optimize for or derive a surface parametrization, it turns out that this is simple to do with the information already stored with an NGF instance. Recall that our representation contains the configuration of the patches as a quadrilateral mesh Q.

We first obtain a parametrization f on Q using standard parametrization algorithms. For each patch σ in the NGF, we can find the texture coordinate g of any sample $\boldsymbol{u}$ using bilinear interpolation,

$$\begin{aligned} g(\boldsymbol{u}) = (1-\boldsymbol{u}_u)((1-\boldsymbol{u}_x)\cdot f^{-1}(v_{00}) + \boldsymbol{u}_x \cdot f^{-1}(v_{10})) \\ + \boldsymbol{u}_y((1-\boldsymbol{u}_x)\cdot f^{-1}(v_{01}) + \boldsymbol{u}_x \cdot f^{-1}(v_{11})) \end{aligned} \tag{1}$$

where $v_{00}, v_{01}, v_{10}, v_{11}$ are the corner vertices of σ. Figure 3 demonstrates process on the Planck mesh.



	100	250	1000	2500
8×8	0.5 ms	1.5 ms	2.5 ms	6 ms
15×15	1.5 ms	3 ms	8.5 ms	19 ms
22×22	2.8 ms	5 ms	15 ms	40 ms
29×29	4.5 ms	8.5 ms	32 ms	70 ms

Timings

Figure 2: Realtime rasterization of neural geometry fields. We implemented a realtime rasterizer for neural geometry fields following the pipeline in Figure 1. The renderer is capable of retrieving normal vectors from the frame buffer without minimal memory footprint, and this enables basic surface shading. The Timings table displays the frame times for the entire pipeline for various patch counts (top) and tessellation resolutions (left). At a typical tessellation resolution of $k = 15$, our rendering framerates remain well above 30 frames-per-second. The mesh above is xyz ©Stanford 3D Scanning Repository.

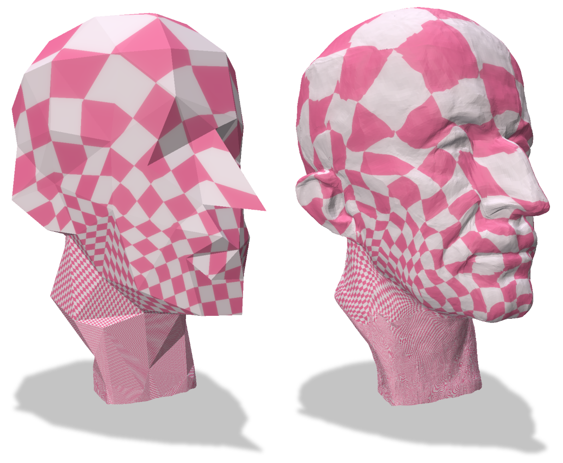


Figure 3: Texture mapping. Neural geometry fields are amenable to texture mapping by inhering an arbitrary uv parametrization on its base quadrilateral mesh. We demonstrate this process above on the Planck mesh. A parametrization for the base (left) is found using LSCM. Then, using Equation 1, we transfer the mapping onto points of the neural geometry field. Meshes adapted from Planck ©MPI.

3 ABLATIONS

Positional encoding. The configuration of frequencies used for positional encoding leads to notable effects in the resulting representation. Our method does well even without positional encoding, but we find that using some rather than no Fourier features typically leads to improved performance. Interestingly, we also found that too many frequencies lead to the opposite effect, often indicated by unstable losses throughout optimization. Ultimately, we found that $L = 8$ was the optimal number of frequencies, see Figure 4.

Loss function. We briefly mentioned that we chose an appearance-based loss as it was more stable than distance-based queries such as the Chamfer distance. To analyze this, we perform coarse-to-fine optimization on our representation using our loss function and the Chamfer distance loss. In Figure 5, we show the Chamfer loss for both approaches and display the resulting representations. Our loss function requires fewer iterations to converge in comparison to the Chamfer loss function, even in terms of the Chamfer metric itself. Furthermore, the representation optimized with the Chamfer loss contains many visual artifacts, primarily in the form of a jagged surface. Our loss function, in comparison, is both more visually and geometrically consistent with the reference.

Smoothing term. We include a smoothing term in the loss in order to distribute the vertices on our representation across its surface. In Figure 6 we demonstrate the effect this has on the vertex density of the final (uniformly sampled) surface, with and without this term. We measure vertex density as the minimum distance from a vertex to any of its one-ring neighbors. Adding the smoothing term indeed relieves the surface of kinks with high concentrations of vertices. For the particular model chosen in Figure 6, the summed vertex density is 716.82 for the representation without smoothing and 756.31 for the representation with smoothing.

4 NEURAL METHODS

We briefly compare our method to other neural surface representations.

Instant NGP. Multi-resolution hash encoding performs well in representing various neural graphics primitives, including signed-distance fields. In comparing neural geometry fields to these neural signed distances fields, we rely on marching cubes to reliably extract a concrete surface. We provide an amble budget of triangles for this extraction process, more than double that of the mesh extracted from our corresponding representation.

We depict this comparison in Figure 7, using the Indonesian model. Against Instant NGP models of varying table sizes, and corresponding feature sizes to equalize storage size, our method recovers a more stable surface with significantly fewer visual artifacts.

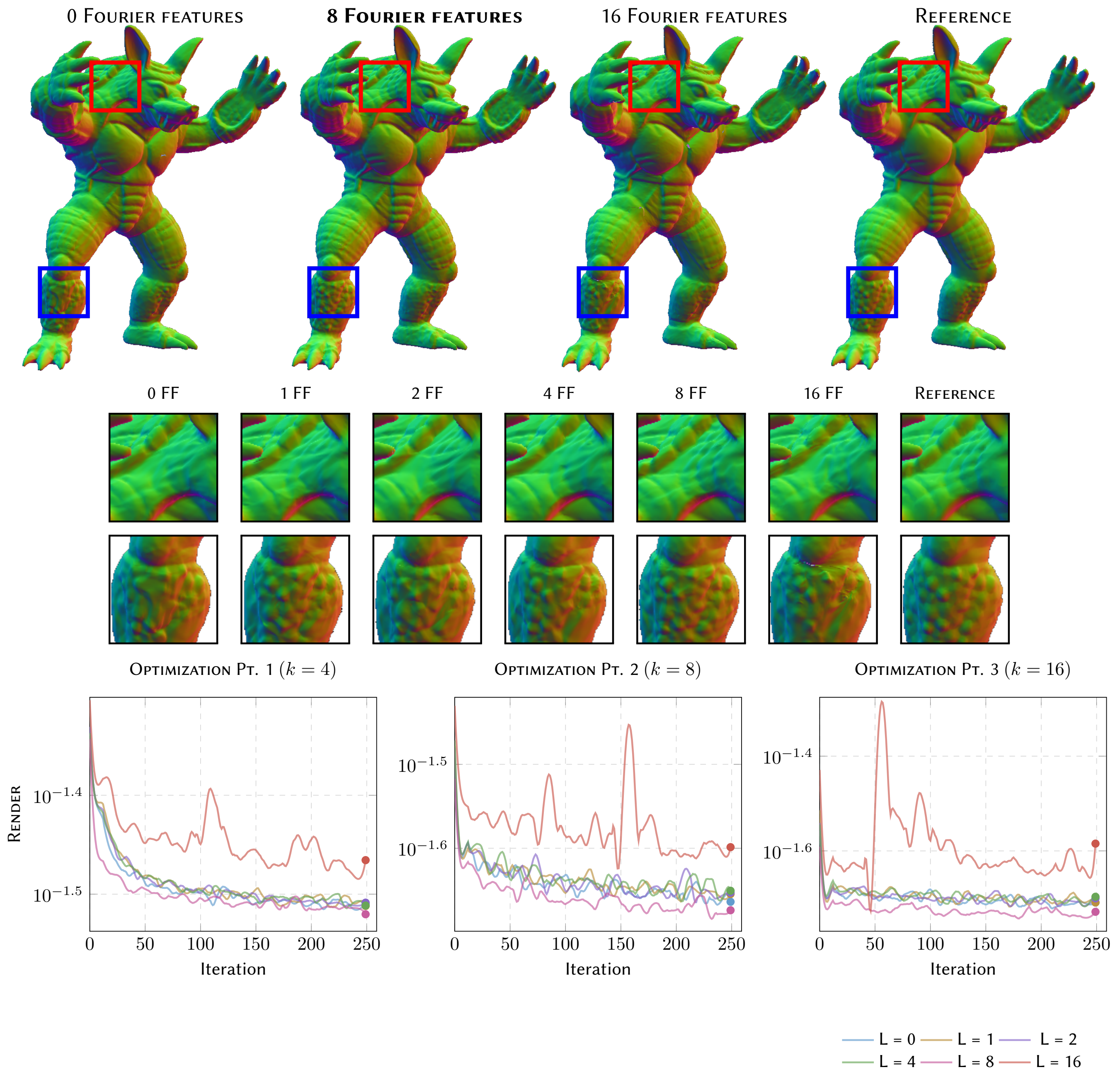


Figure 4: Positional encoding. Varying the number of frequencies L used in positional encoding affects the fidelity of the resulting representation. As shown above (top), smaller L can still perform well, but still lacks details that additional frequencies can provide at $L = 8$. Too many frequencies, on the hand, can result in the opposite problem. We plot the rendering loss throughout optimization, for each phase of our pipeline (i.e. for each fixed tessellation resolution). One thus observes that $L = 8$ is an optimal configuration for positional encoding. Model credited to the Stanford 3D Scanning Repository.

It can be presumed that the hash table size at such compression levels is too small, leading to several collisions, which manifest as various artifacts.

Neural Subdivision. Unlike most signed-distance field surface representations, neural subdivision directly generates a discrete triangle mesh. Similar to us, the met, the method requires a coarse base mesh in order to upscale it. The method attempts to generalize its knowledge of surfaces, thereby being applicable well outside its

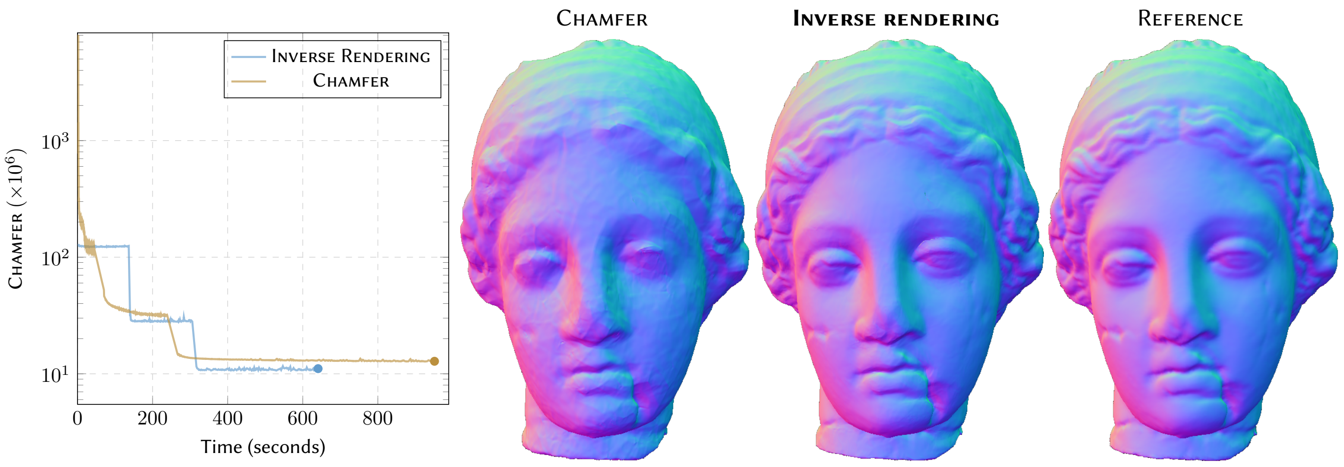


Figure 5: Loss function. We compare the optimization of our representation under our proposed loss function and the Chamfer distance function. On the left, we plot the Chamfer loss along training duration for each loss. Note that our appearance-based approach (second from right) requires less time to converge to a visually indistinguishable result from the reference model (right). Using the Chamfer distance loss, on the other hand, not only leads to a worse Chamfer loss in the end, but the result is also visually significantly worse than our method (second from left). The mesh is due to Cyberware.

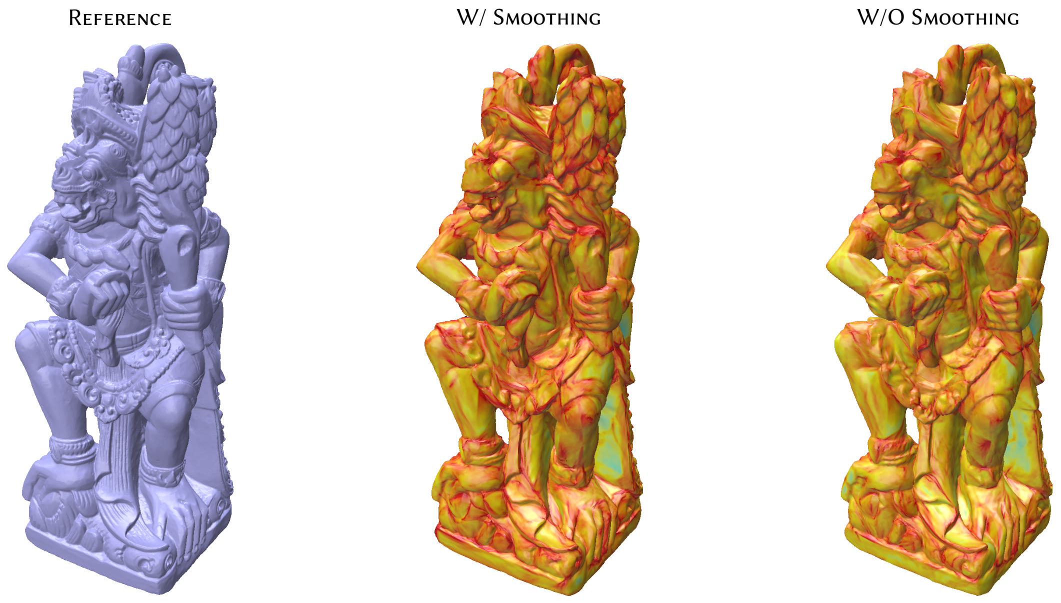


Figure 6: Smoothing loss term. Above, we demonstrate the impact of including the smoothing term in our objective function. Regions with more red indicate high vertex density, whereas those with more blue or green indicate lower density. Inclusion of the smoothing term relieves vertex congestion, especially on the waist and face of the statue. The mesh is taken from Indonesian Statue ©peel3d.

trained domain. However, this is also its major drawback, as it often fails to recover higher frequency details on large surfaces.

To perform the comparison, we train a neural subdivision model on a dataset consisting of only the primary model, shown in Figure 8. The primary model is decimated to 1000 faces to serve as the coarse base mesh that the model operates on. To match the storage budget of our representation, we increase the width of the network layers used to 48 neurons, bringing the total size to 138 kilobytes. We run our method with 250 patches with the usual configuration, with a total cost of 85 kilobytes.

Since neural subdivision performs local convolutions on the coarse base mesh, it is unable to recover sharp details on surfaces. In Figure 8, for example, it cannot infer the structure of the eyes and lips of the model. On the hand, since our method is trained thoroughly

Indonesian

Reference	Ours	Instant NGP		
Configuration	2.5K Patches	$T = 2^{11}$, $F = 8$	$T = 2^{12}$, $F = 4$	$T = 2^{13}$, $F = 2$
Render	$1.84 \cdot 10^{-4}$	$1.64 \cdot 10^{-2}$	$1.65 \cdot 10^{-2}$	$1.64 \cdot 10^{-2}$
Normal	$6.84 \cdot 10^{-3}$	$8.22 \cdot 10^{-1}$	$8.24 \cdot 10^{-1}$	$8.23 \cdot 10^{-1}$
Chamfer	$7.28 \cdot 10^{-6}$	$2.99 \cdot 10^{-5}$	$7.06 \cdot 10^{-6}$	$7.74 \cdot 10^{-6}$

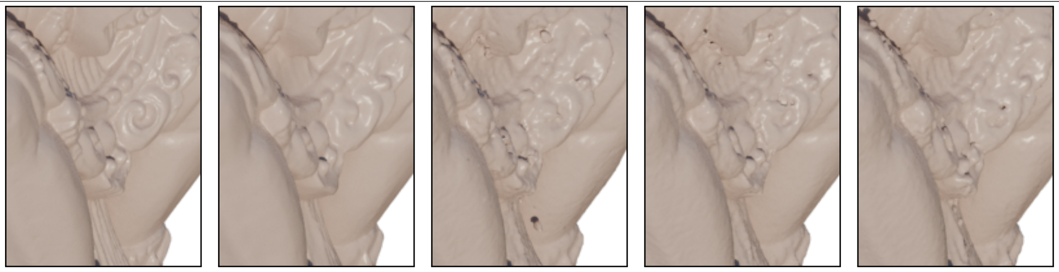


Figure 7: Instant NGP. Above, we compare neural geometry fields with meshes reconstructed from Instant NGP signed-distance fields. We evaluate this comparison at multiple configurations with similar sizes to our representation, and extract the surface using marching cubes. Although the distribution of vertices on the results of Instant NGP can envelop the target surface, they are often noisy in comparison to our method. Hence, whereas the Chamfer distances may be of similar magnitude, the visual metrics indicate that our method is more suitable for compact surface representation. This model is from Indonesian Statue ©peel3d.

to match the surface and its higher frequency features, it is able to reconstruct the model much better and at a fraction of the storage cost incurred by neural subdivision.

5 ADDITIONAL RESULTS

Figures 9-14 show additional results of reconstructed neural geometry fields, along with the results of baselines QSlim and nvdiffmodeling for comparison. At high compression rates, we observe that our method is significantly better at preserving sharp edges and high frequency surface details, see Figures 11 and 13. A limitation of our method, as demonstrated in Figure 12, is that for extreme compression rates (e.g. low patch counts), the surface partitioning step of our method may fail to procure a sufficient number of patches. This is a result of the simplification process, whereby a majority of the remaining triangles are non-manifold, and cannot easily be combined into quadrilaterals.

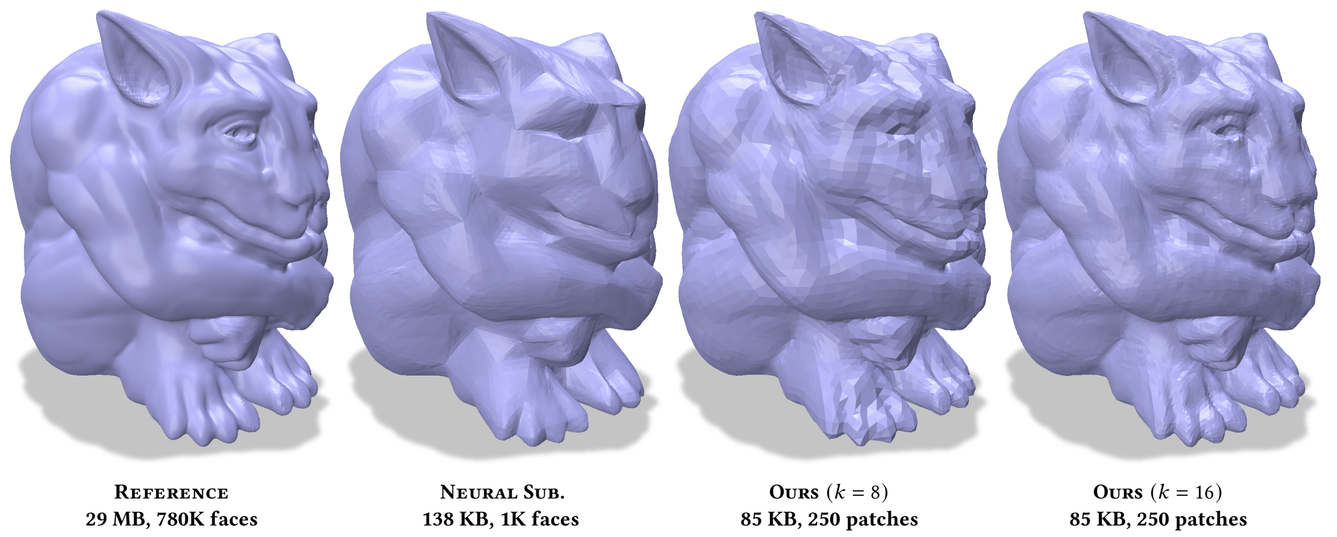


Figure 8: Neural Subdivision. We compare our method to Neural Subdivision with the model above (left). Despite operating on a smaller coarse mesh and incurring a smaller storage overhead, our method is capable of recovering sharper details. Even when overfit to a single mesh, neural subdivision struggles to learn the higher frequency details of the surface, such as the eyes and lips of the model. In the views above, we run neural subdivision thrice (for a combined subdivision factor 8) on a coarse subdivided mesh with 1000 faces, and run our method using only 250 patches at tessellation resolution $k = 8$ and 16. This mesh is taken from Thingi10K.

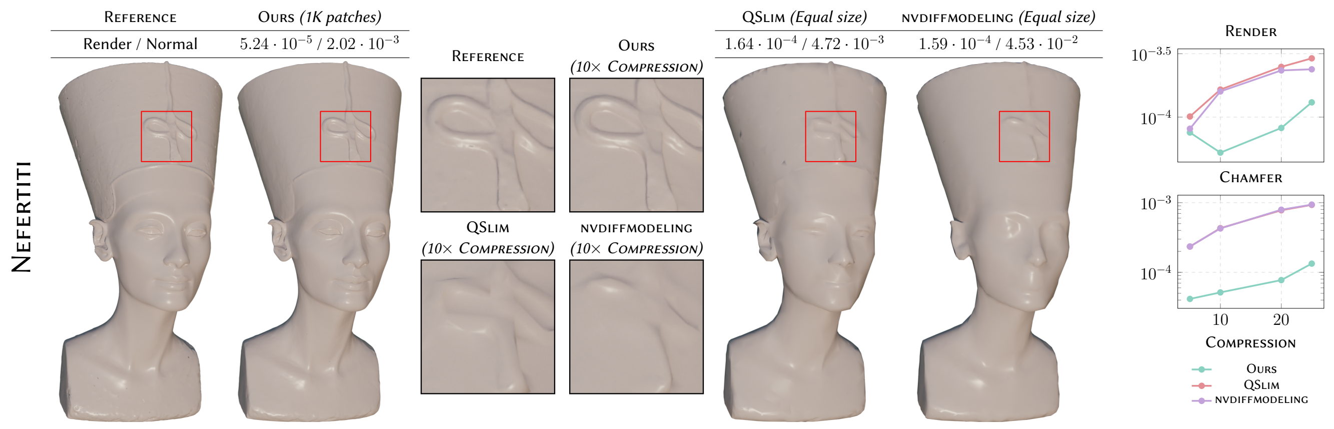


Figure 9: NEFERTITI. Our method obtains an accurate representation of the face details in these while maintaining a notable compression rate. The low vertex count limitation QSLIM and NVDIFFMODELING inhibits them from preserving key features like the eyes and, nose and crown indentations. Above model is NEFERTITI ©Berlin Egyptian Museum.

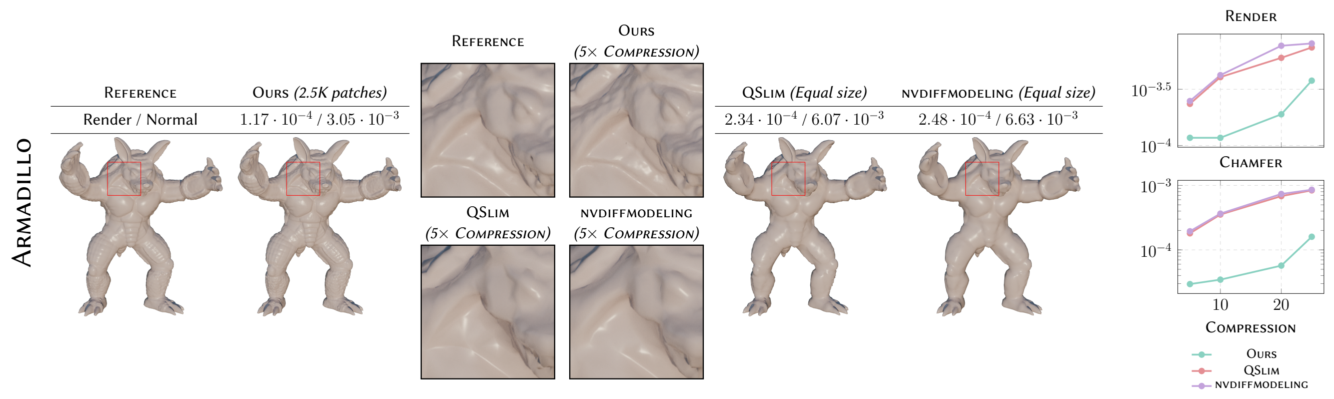


Figure 10: Armadillo. Even for relatively small models, our method outperforms the baseline methods and is able to reconstruct sharper features, such as the legs and collarbone of this armadillo. Above model is Armadillo ©Stanford 3D Scanning Repository.

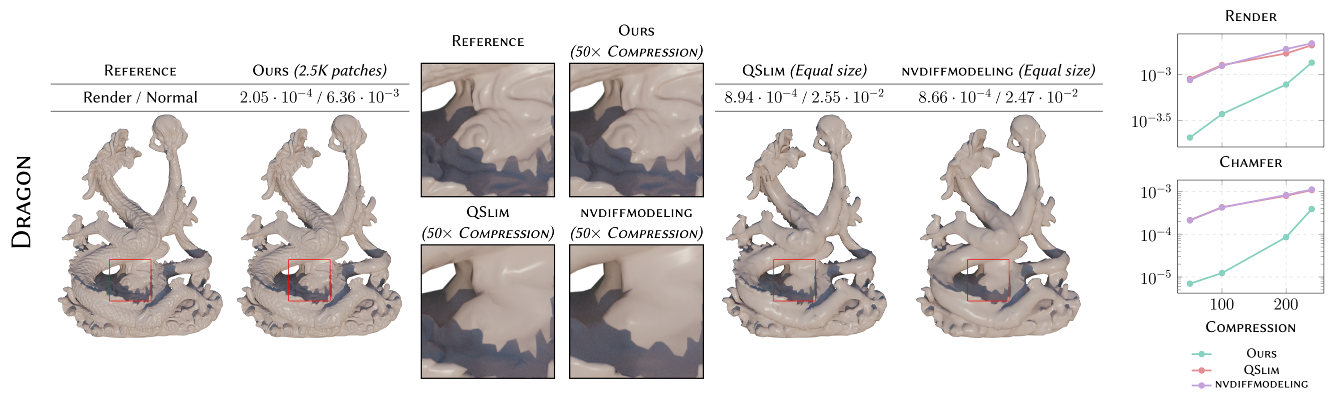


Figure 11: Dragon. With a sufficient amount of patches, neural geometry fields have the capacity to closely match the reference surface. In the case of this dragon mesh, the result of our method captures the finer folds of the surface, even those which are sometimes occluded. Due to their limited vertex budget, the baseline methods yield an over smoothed reconstruction. The model above is taken from Thingi10K.

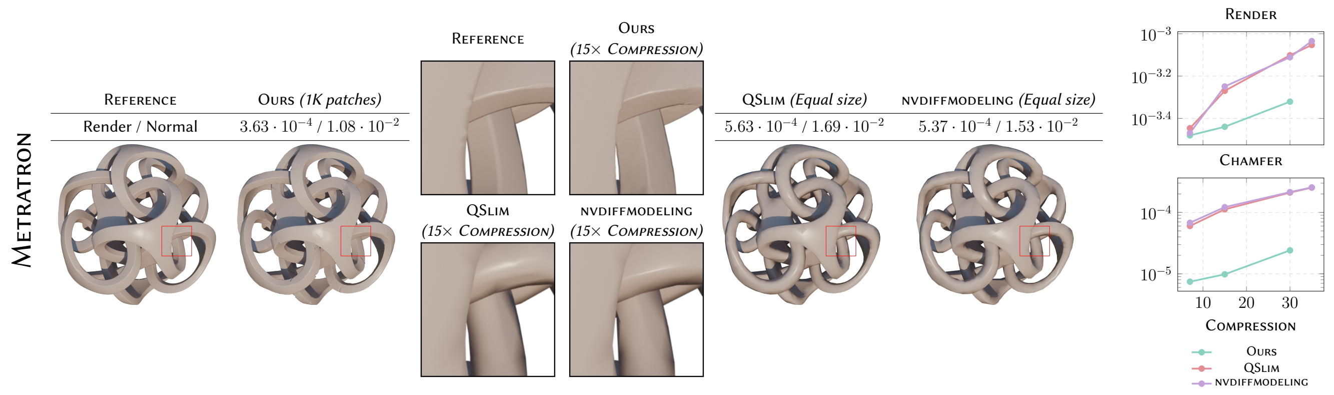


Figure 12: Metratron. At modest compression rates, the baselines fail to capture the sharp edges of this high genus mesh. On the other hand, our optimization pipeline results in a reconstruction that preserves such edges in a compact manner. However, at extreme compression rates, the simplification process of our surface partitioning procedure fails to yield a sufficient number of useful patches, in which case our pipeline fails to successfully converge. The mesh above is taken from Thingi10K.

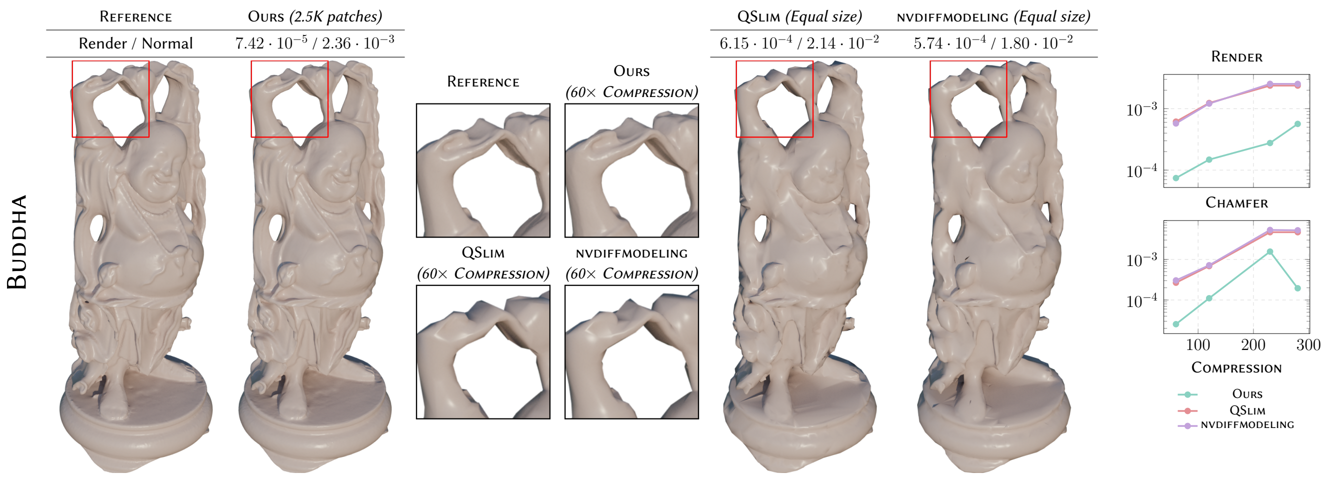


Figure 13: BUDDHA. Even at such a high compression rate, our method is still able to grasp the intricate folds on the statue's cloth. The baseline methods, on the other hand, struggle to grasp the sharp folds on the fabric. One notes, however, that our result at the maximal patch counts performs worse than a lower patch count result. We observed that this is due to large sections of overlapping geometry in the extracted mesh, which is not easily combatted by our current loss function. Above model is BUDDHA ©Stanford 3D Scanning Repository.

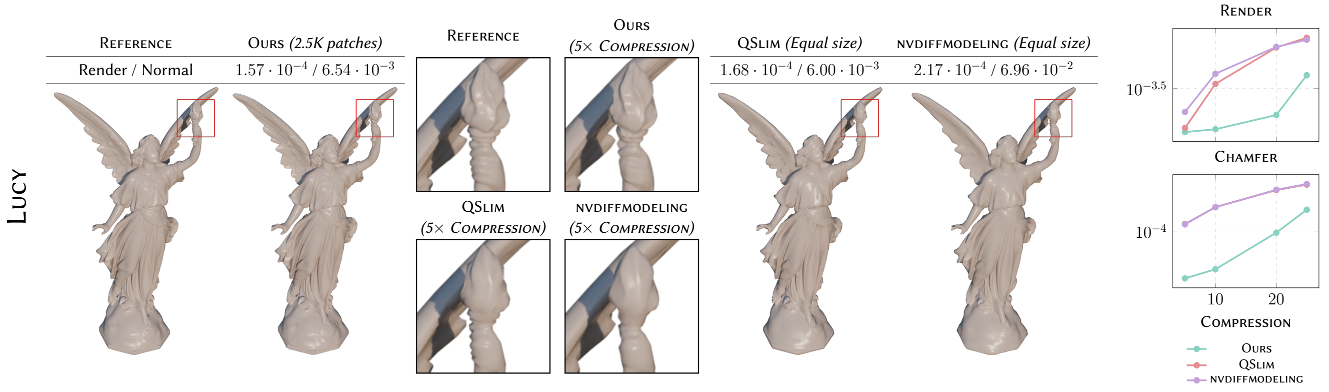


Figure 14: LUCY. Our method is able to recover the fine details of this statue, such as the fingers on the hand and the torch. The baselines, on the other hand, struggle with this, and generate overly smoothed results. Above model is LUCY ©Stanford 3D Scanning Repository.