NeuroGF: A Neural Representation for Fast Geodesic Distance and Path Queries
(*Supplementary Material*)

Anonymous Author(s)
Affiliation
Address
`email`

In this supplementary material, we provided more visualization examples and visual comparisons of different approaches for geodesic representation. Moreover, we presented detailed explorations on the data efficiency of learning NeuroGFs, and analyzed the memory footprints of different approaches. In the end, we further performed qualitative and quantitative evaluations on the surface geometry information encoded in NeuroGFs. Note that our source code and the checkpoints of the trained neural models have also been uploaded.

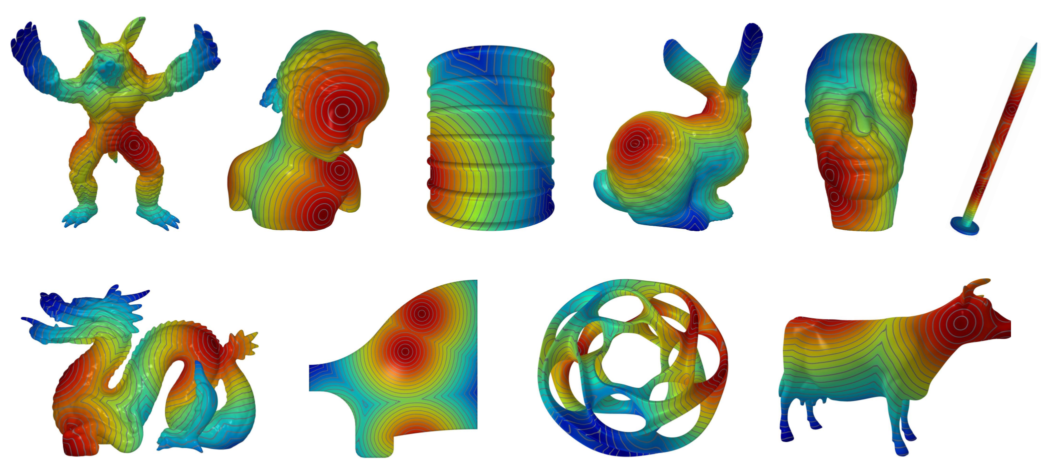

Figure S1: Visualization of MSAD geodesic distance fields computed by our approach. Zoom in to see details.

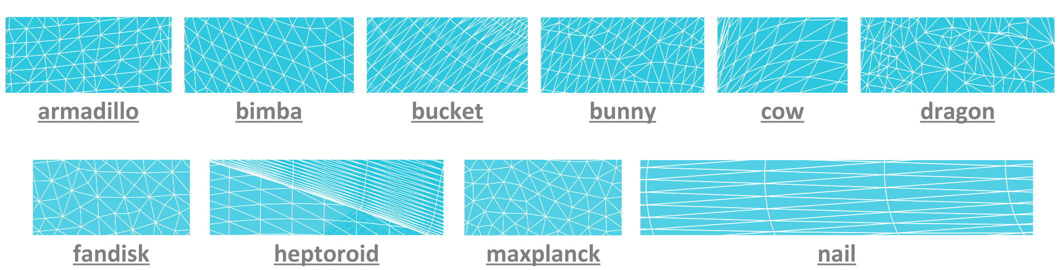

Figure S2: Visualization of mesh triangulations.

1) Visualization of MSAD Geodesic Distance Fields. We randomly selected 5 source points on the shape surface to produce the resulting MSAD geodesic distance field, as illustrated in Figure S1. Besides, we also visualized the triangulation of the testing meshes in Figure S2.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

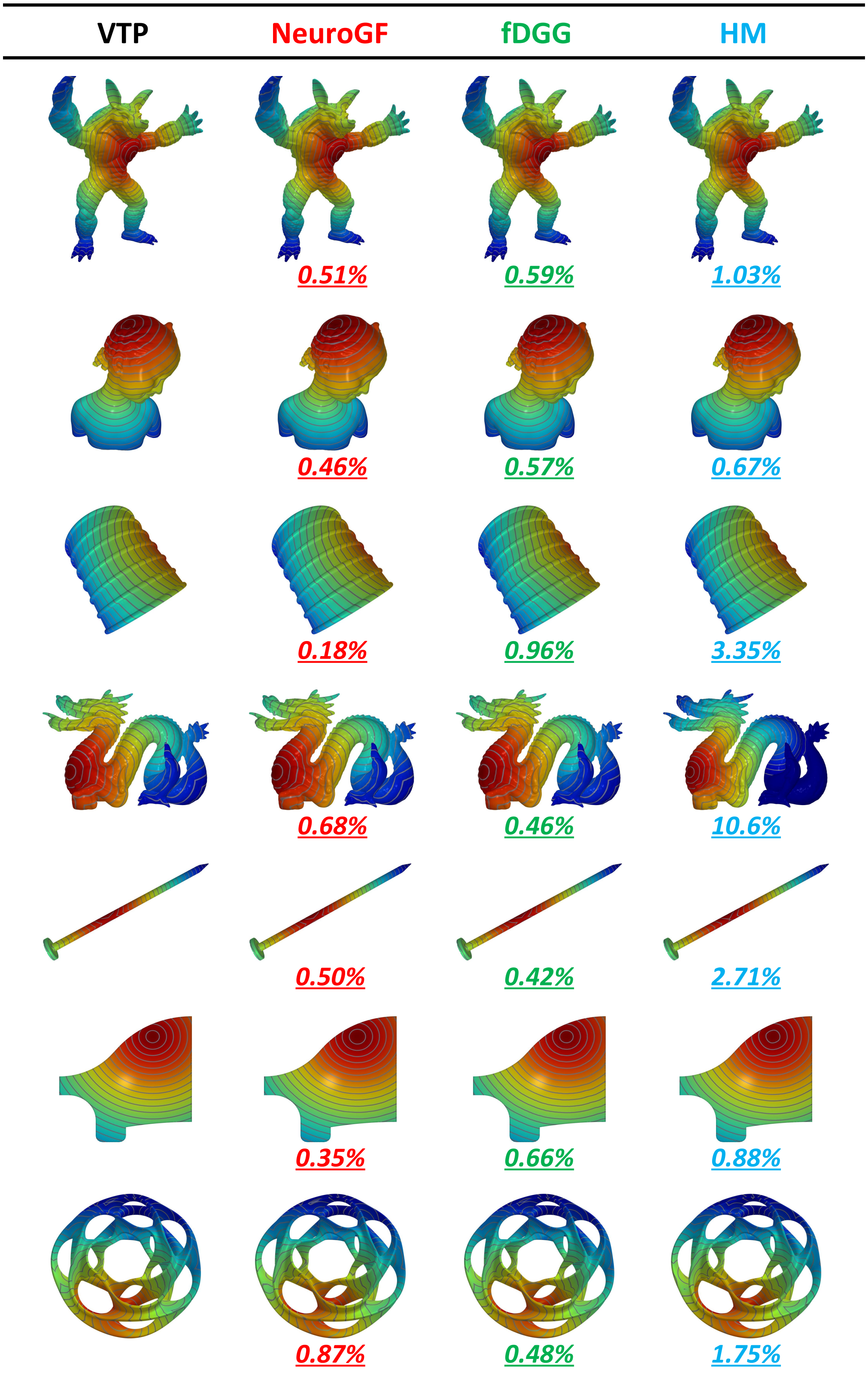

Figure S3: Visual comparison of SSAD geodesic distances. Below each example, we also marked the corresponding mean relative error (%). Zoom in to see details.

2) Visualization of SSAD Geodesic Distance Fields. We provided more visual comparisons of the SSAD geodesic distance fields exported from different approaches, as presented in Figure S3. To facilitate comparisons, we also marked the quantitative metrics of geodesic representation accuracy below each example computed from different approaches. In particular, Figure S4 presents close-up views for regions on the anisotropic *nail* model.

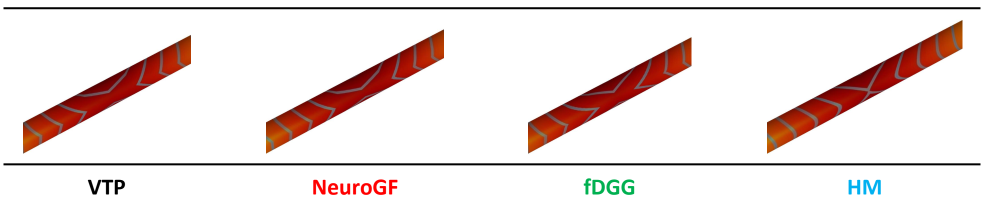

Figure S4: Close-up views for geodesic distance fields deduced from different approaches on the anisotropic *nail* model.

Table S1: Effects of offline training with different amounts of paired ground-truth geodesics on the *dragon* model. In the right two columns, we respectively record the preprocessing (Prep.) time cost for generating ground-truth geodesic distance (G.D.) and shortest path (S.P.) data.

#Sources	*MRE (%)*	*Chamfer-L_1 ($\times 10^{-2}$)*	*Prep. G.D. (minutes)*	*Prep. S.P. (hours)*
10000	0.66	1.297	119.1	21.6
1024	0.68	1.319	12.2	2.2
512	0.75	1.397	6.1	1.1
256	0.83	1.580	3.0	0.6
128	1.09	2.044	1.5	0.3
64	1.63	2.926	0.8	0.14
32	2.21	3.725	0.4	0.07

3) Offline Training with Limited Ground-Truth Geodesics. For the preprocessing stage of preparing the required ground-truth geodesics for offline training of NeuroGFs, during which 4 data generation scripts run in parallel, we explored the effects of using different numbers of source points, i.e., 10000, 1024, 512, 256, 128, and 64. For each source point, we only preserved its geodesic distances between 4096 target points and its shortest paths between 2048 target points. As compared in Table S1, our approach can still achieve relatively satisfactory geodesic representation performances when only a limited amount of source points are exploited to produce ground-truth training data. Besides, we can also observe that using a much larger amount of training samples (i.e., with 10000 source points) only leads to insignificant performance gains. Furthermore, note that here our preprocessing stage is not well-optimized, since we need to do lots of disk writing operations. We can directly integrate the approaches used for data preparations (fDGG [2] and DGG-VTP [1]) to Python to avoid costly disk writing time in the future. Comparatively, the recent work SEP [4] requires around 30 minutes for preprocessing a mesh with 20K vertices. Its time complexity of preprocessing is $O(f(n)+m^3n)\sqrt{n})$, where n is the number of mesh vertices and $f(n)$ is the time complexity of the used SSAD algorithm, which is at least $O(n)$ time for fDGG [2]. We can see that this time complexity increases more than linearly in the number of mesh vertices. Thus, we can estimate that for the *dragon* mesh model with more than 400K vertices, SEP would require more than 10 hours to finish preprocessing. This is much longer than our preprocessing stage using 1024 source points, which achieves geodesic representation accuracy comparable to SEP.

4) Memory Footprint of Online Point-to-Point Geodesic Query. We compared the memory footprints of different approaches for online answering the geodesic distance between an input pair of source and target points. The quantitative results are reported in Table S2, where we can observe that our approach consistently maintains satisfactory memory efficiency. Thanks to our implicit querying paradigm, the resulting memory footprint is independent of the complexity of the given shape. Instead, for both the two competing approaches of HM [3] and fDGG [2], their memory footprints increase when dealing with larger meshes.

Table S2: Comparison of memory efficiency for online point-to-point geodesic query.

Mesh	***#V (K)***	***Memory Footprint (MB) of Point-to-Point Geodesic Query***		
		HM [3] (on CPUs)	fDGG [2] (on CPUs)	NeuroGF (on GPUs)
armadillo	173	272	60	10
bimba	75	138	26	10
bucket	35	45	15	10
bunny	35	56	13	10
cow	46	70	17	10
dragon	436	597	152	10
fandisk	20	29	7	10
heptoroid	287	690	108	10
maxplanck	49	85	17	10
nail	2.4	2.9	1.3	10

5) Geometry Representation Recovery. As pointed out in the paper, NeuroGFs jointly encode both 3D geometry and geodesics in a unified neural representation structure, although geodesic information is our major focus and geometric information just serves as a byproduct in this work.

Here, we provided necessary qualitative and quantitative evaluations for signed distance field information encoded in NeuroGFs. We deduced the predictions of signed distance values on a uniformly-distributed 3D grid with the resolution of 512^3, and then measured the L_1 differences between the predicted and ground-truth signed distances, as reported in Table S3. For visualization, we applied the Marching Cubes [5] algorithm for isosurface extraction. The resulting mesh reconstructions are displayed in Figure S5.

Table S3: Mean L_1 errors between our predicted and ground-truth signed distances.

Mesh	armadillo	bimba	bucket	bunny	cow	dragon	fandisk	heptoroid	maxplanck	nail
L_1 ($\times 10^{-3}$)	1.22	0.97	0.68	0.97	0.86	1.28	0.67	1.01	1.03	0.45

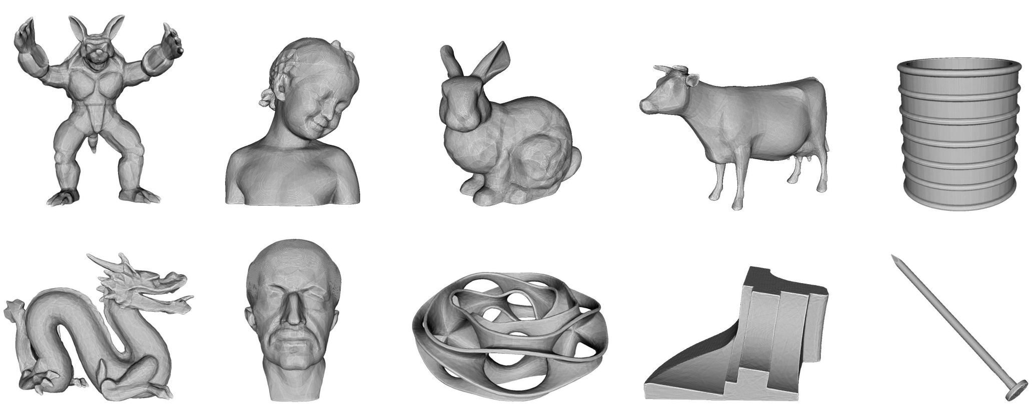

Figure S5: Visualization of mesh reconstruction deduced from NeuroGFs.

References

[1] Y. Y. Adikusuma, J. Du, Z. Fang, and Y. He. An accuracy controllable and memory efficient method for computing high-quality geodesic distances on triangle meshes. *CAD*, 150:103333, 2022.

[2] Y. Y. Adikusuma, Z. Fang, and Y. He. Fast construction of discrete geodesic graphs. *TOG*, 39(2):14:1–14:14, 2020.

[3] K. Crane, C. Weischedel, and M. Wardetzky. Geodesics in heat: A new approach to computing distance based on heat flow. *TOG*, 32(5):152, 2013.

[4] C. Gotsman and K. Hormann. Compressing geodesic information for fast point-to-point geodesic distance queries. In *SIGGRAPH Asia*, pages 1–9, 2022.

[5] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.