GAN-BASED NERF NOISE SIMULATION IN MESH DENOISING TASK

Anonymous authors
Paper under double-blind review

A APPENDIX

A.1 CALCULATE DISTANCE BETWEEN POINT AND MESH

In this section we present an algorithm for quick calculation of distance between point and triangle mesh. The three-dimensional space around a mesh is described as a Voronoi diagram constructed for different classes of geometric primitives that mesh consists of: facets, edges, and vertices.

Consider the point Q and calculate the distance from Q to the mesh. The algorithm consists of the following steps:

- Find a vertex of the mesh A closest to the point Q. This can be done, for example, using a kd-tree calculated previously for all vertices of the mesh. We denote by $V_1, \ldots, V_n$ the vertices that are adjacent to vertex A. We also denote by $C_1, \ldots, C_n$ the centroids of facets adjacent to the vertex A.
- Denote vector $\overline{AQ}$ by $\bar{a}$. Next vectors $\overline{AV}_1, \ldots, \overline{AV}_n$ we denote by $\bar{v}_1, \ldots, \bar{v}_n$. Finally we denote vectors $\overline{AC}_1, \ldots, \overline{AC}_n$ by $\bar{c}_1, \ldots, \bar{c}_n$.
- First we should check if point Q is in the reference cone of A.

The article clearly describes that the three-dimensional space above/below a mesh can be described as a Voronoi diagram constructed for different classes of geometric primitives. The classical Voronoi diagram is a partition of space into regions, where each region of it forms a set of points closer to one of the elements of a certain set than to any other element of the set. A mesh consists of three types of geometric primitives: facet, edge, and vertex.

The space in which the mesh is represented is transformed into a Voronoi diagram for the facets, edges, and vertices of the mesh. Drawing from the article:

In the figure, red indicates the areas where the points are closest to one of the facets than to any other facet or any of the edges or vertices. Similarly, blue indicates the areas where the points are closest to some edge, and yellow indicates some vertex.

If you want to find the shortest distance from an arbitrarily taken point to the mesh surface, then you need to take into account this feature of dividing the space around the mesh, since the distance from a point to a flat triangle in 3D is not calculated in the same way as the distance from a point to a segment or from a point to a point. It is important to understand which of the geometric primitives is closest to the point before calculating the distance.

The algorithm for finding the shortest distance can be implemented without constructing a Voronoi diagram, but with the assumption that the surface to which the distance needs to be calculated is sufficiently convex.

Suppose you want to calculate the distance from the point Q to the mesh. The algorithm consists of the following steps:

1. Search for the vertex of the mesh A closest to the point Q. This can be done, for example, using a kd-tree calculated previously for all vertices of the mesh. Denote by $V_1, \ldots, V_n$ the vertices that are adjacent to vertex A. We also denote by $C_1, \ldots, C_n$ the centroids of facets adjacent to the vertex A;
2. Check whether the point Q lies in the reference cone of this vertex (in the figure these cones are indicated in yellow). To do this, take the vector connecting vertex A and point Q, that is, vector AQ. Next, you need to calculate the scalar products of the vector AQ with the vectors $AC_1, \ldots, AS_p$. If

all these scalar products are strictly less than zero, then the point Q belongs to the support cone. In this case, the desired distance is the length of the vector AQ. If at least one of the scalar products is greater than or equal to zero, then the distance is calculated according to the algorithm in paragraph 3; 3. For each facet k adjacent to vertex A, calculate the vectors L_1C_k, L_2C_k, L_3C_k, where L_1, L_2, L_3 are the midpoints of the facet edges. We also calculate the vectors L_1Q, L_2Q, L_3Q, then calculate the scalar products $(L_iC_k, L_iQ), i = 1, 2, 3$. If all three scalar products are greater than or equal to zero, then the minimum distance from the point Q to the mesh is equal to the distance to the facet k. If otherwise, the distance is calculated according to the algorithm in paragraph 4; 4. Calculate the scalar product of the vector $(AQ, AV_k), k = 1, \ldots, n$. Important: each of the vectors V_k must be normalized before calculating the scalar products. Let's define k for which the scalar product (AQ, AV_k) is maximal. An edge with index k is the nearest edge to the point Q. In this case, the minimum distance from the point Q to the mesh is equal to the distance to the edge k.

A.2 OBJAVERSE-XL SHAPES HASHES

Table 1: Each objaverse-XL shape has a unique hash that identifies it in this dataset.

Name	Train or test	Objaverse ID
bottle	Train	00b2c8c60d2f45a893ee73fd1f107e27
bird	Train	02c81d18c4f04b9b9275fde41d0e715b
sphere	Train	f8c97f11180440ccae5bc156ef087014
key	Train	4bdab6b1e3194045ab6362e4c6cda222
doll	Test	0e30fca3637e4083863e1240d6d1f1bf
spiral	Test	1d6ad3e20daa4873a3b1a0ab6c0ea8d1

A.3 FULL RESULTS

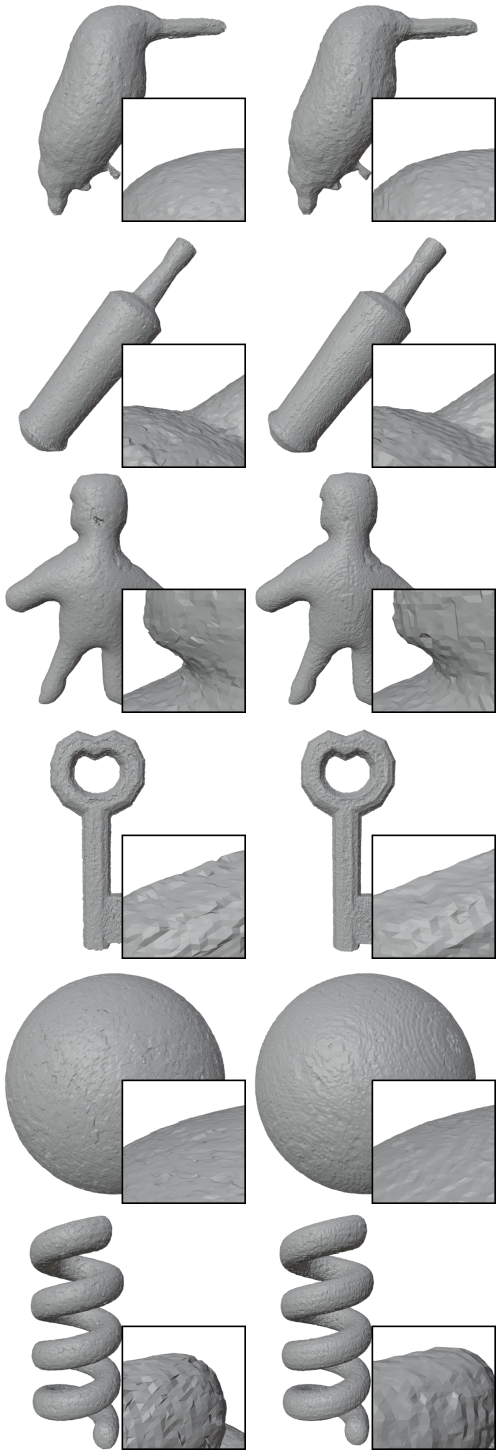
Table 2: Basic models results: DPSR + MC, KNN, MLP, U-Net. Experiments show the best results in KL div. for a specific test shape and all shapes highlighted in green and dark green, respectively. Our GAN-based approach performs significantly better for the rest of the metrics which are shown in Table 5.

	Test shape	Metrics				
		KL div. ↓	Cosine ↓	Linear ↓	Manh. ↓	Nuclear ↓
DPSR + MC	Bird	0.44385	0.02214	0.29057	8.13646	0.79620
	Bottle	0.49276	0.05177	0.57202	9.93406	1.60127
	Key	**0.06330**	0.06551	0.63861	13.24510	2.26182
	Sphere	**0.35360**	0.13064	0.69201	20.95043	1.80963
	Doll	**0.52423**	0.09685	0.62921	19.19287	1.61193
	Spiral	0.49276	0.07408	0.52683	15.96581	1.38656
KNN-regressor	Bird	**0.44349**	0.01495	0.24551	6.37309	0.59441
	Bottle	0.48912	0.03381	0.44590	9.42089	1.32781
	Key	0.06630	0.04790	0.48711	11.63991	1.53463
	Sphere	0.35994	0.08495	0.56878	16.45870	1.43616
	Doll	0.52934	0.03354	0.38112	10.68838	1.02367
	Spiral	**0.48815**	0.00552	0.14572	3.93982	0.44374
MLP	Bird	0.45572	0.05788	0.59004	14.28690	2.03749
	Bottle	0.50096	0.16061	1.43471	30.53953	5.69239
	Key	0.06677	0.04867	0.61003	12.47303	2.51558
	Sphere	0.36640	0.11959	1.04522	25.71050	3.72137
	Doll	0.54231	0.16903	0.80894	23.71446	3.58923
	Spiral	0.50105	0.06661	0.61870	14.49724	2.14522
U-Net	Bird	0.45791	0.12729	0.94130	18.56488	3.52717
	Bottle	**0.48909**	0.05028	0.56939	10.86997	1.68377
	Key	0.07039	0.27131	2.28507	40.40780	11.84921
	Sphere	0.36845	0.08761	0.73039	14.87120	2.56812
	Doll	0.54249	0.13434	1.06966	22.07343	4.16051
	Spiral	0.50067	0.13567	1.06968	24.73850	4.13640

Table 3: GAN training results. Five train datasets: $bird$, $bottle$, key, $sphere$, all. Two test datasets: $doll$, $spiral$. The best results for a specific test shape are highlighted in green. The best metrics for all shapes are highlighted with dark green. The GAN results for the KL div. are slightly lower, however they are comparable to the rest of the approaches. The GAN results for other metrics are significantly better than others.

	Train shape	Test shape	Metrics				
			KL div. ↓	Cosine ↓	Linear ↓	Manh. ↓	Nuclear ↓
In domain	Bird	Bird	0.45106	0.00597	0.18472	5.28831	0.60968
	Bottle	Bottle	0.49516	0.01402	0.26539	5.96482	0.79997
	Key	Key	0.06589	**0.01937**	**0.32343**	**5.68141**	**1.17867**
	Sphere	Sphere	0.35953	0.00557	0.15620	3.67948	0.47831
Out of domain	Bottle	Bird	0.45183	0.00671	0.17695	4.63796	0.55543
	Key	Bird	0.45828	0.02109	0.32092	7.61674	1.07289
	Sphere	Bird	0.45269	**0.00251**	**0.10487**	**2.84228**	**0.32697**
	All	Bird	0.45327	0.00601	0.16738	4.50107	0.53778
	Bird	Bottle	0.49573	0.01846	0.32097	6.18145	1.00624
	Key	Bottle	0.50158	0.03387	0.43412	8.31266	1.36748
	Sphere	Bottle	0.49697	0.01151	0.24423	5.57860	0.79836
	All	Bottle	0.49770	**0.00951**	**0.21888**	**5.25660**	**0.68357**
	Bird	Key	0.06479	0.02679	0.37076	6.79448	1.27395
	Bottle	Key	0.06527	0.06527	0.34457	6.19686	1.22980
	Sphere	Key	0.06415	0.03881	0.44968	8.66565	1.49396
	All	Key	0.06473	0.03096	0.40012	7.69379	1.39265
	Bird	Sphere	0.35944	**0.00183**	**0.09029**	**2.37248**	**0.32119**
	Bottle	Sphere	0.36089	0.00613	0.17313	4.32524	0.51328
	Key	Sphere	0.36573	0.01888	0.32437	8.69921	0.97167
	All	Sphere	0.36108	0.00282	0.10764	2.65000	0.36230
Test domain	Bird	Doll	0.53548	0.01027	0.22929	5.39447	0.67479
	Bottle	Doll	0.53560	0.01730	0.28424	7.04564	0.79888
	Key	Doll	0.54289	0.02074	0.32903	7.94347	1.14373
	Sphere	Doll	0.53638	**0.00625**	**0.16130**	**4.68518**	**0.46974**
	All	Doll	0.53805	0.00936	0.19859	5.59246	0.55628
	Bird	Spiral	0.49832	0.00642	0.18521	5.28779	0.51220
	Bottle	Spiral	0.49747	0.01647	0.29413	7.96902	0.83973
	Key	Spiral	0.50192	0.03417	0.44514	11.57165	1.42135
	Sphere	Spiral	0.49742	**0.00368**	**0.12526**	**3.60572**	**0.31796**
	All	Spiral	0.49609	0.00976	0.21889	6.00135	0.60933

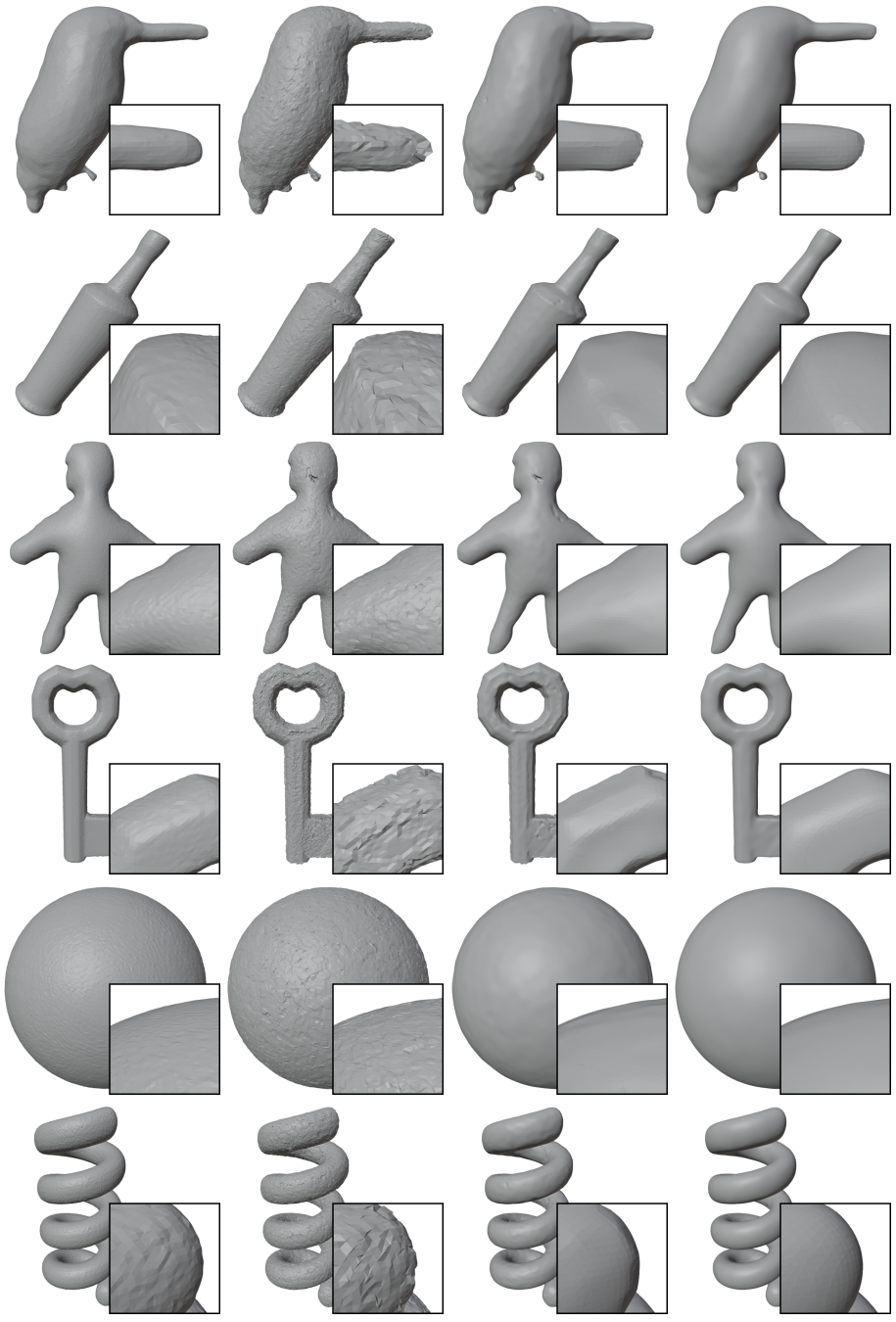
A.4 NeRF-like noise examples



(a) Artificial NeRF noise (b) Real NeRF noise

Figure 1: The real NeRF noise is is compared with artificial noise generated by our pipeline.

A.5 DENOISING ILLUSTRATION



(a) GT (b) GAN-based noisy (c) Trained on Synthetic (d) Trained on Ours

Figure 2: GT and noisy meshes are prepared for denoising tests as explained in Section 7.1. The denoising was performed with the Cascaded Regression model which was trained on the dataset produced by our GAN-based pipeline. We have trained Cascaded Regression on the dataset produced by KNN-based pipeline for comparison to our method. It can be seen that Cascaded Regression trained on GAN-based dataset performs better.