DMNet: Delaunay Meshing Network for 3D Shape Representation —Supplementary Material—

Chen Zhang[1] Ganzhangqin Yuan[1,2] Wenbing Tao[1,*]

[1]National Key Laboratory of Science and Technology on Multi-spectral Information Processing, School of Artifical Intelligence and Automation, Huazhong University of Science and Technology, China
[2]TuKe Research

{zhangchen_, gzq_yuan, wenbingtao}@hust.edu.cn

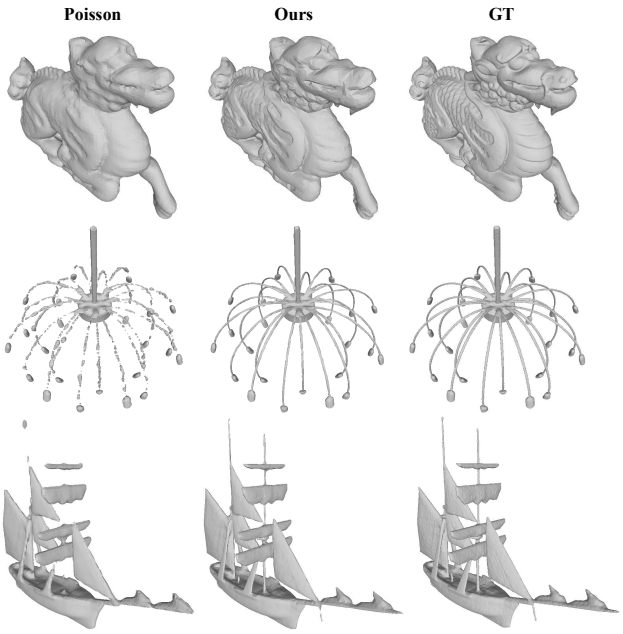

Figure 1: Visual comparison of our method with Poisson.

1. Implementation Details

1.1. Data Preparation

We use the Computational Geometry Algorithms Library (CGAL) [2] to construct the Delaunay triangulation for input points. The point indices and adjacencies of each tetrahedron and triangle can be obtained through the Delaunay triangulation. To generate ground truth labels, we randomly sample N_{ref} reference locations inside each tetrahedron and compute the inside/outside labels with respect to the ground truth surface. For an infinite tetrahedron with an infinite vertex, we are unable to determine its internal reference locations. As a solution, we replace it with a new finite tetrahedron depicted in Figure 2. Afterwards, we sample reference locations inside the new finite tetrahedron and use them as the reference locations for the corresponding infinite tetrahedron. This strategy is particularly useful for open scenes. As for closed models, we can simply assign the labels of all infinite tetrahedrons as outside.

1.2. Special Processing in Scaling Strategy

For an infinite tetrahedron in the Delaunay triangulation, we put it in the same leaf as its neighbor finite tetrahedron in our octree-based partitioning. For tetrahedrons at the leaf boundaries, we adopt a padding strategy to treat a few

*Corresponding author.

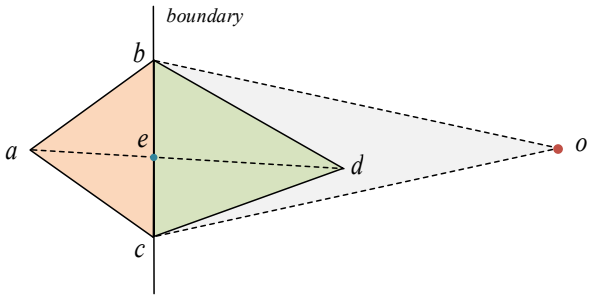

Figure 2: 2D example of constructing a new tetrahedron instead of an infinite one. bco is an infinite tetrahedron with an infinite vertex o. It is adjacent to a finite tetrahedron abc and shares a triangular facet bc on the boundary of the convex hull. We find the geometric center e of bc and extend line segment ae to a new point d, so that ed is K times length of ae. Then we connect d to the triangular facet bc to form a new tetrahedron bcd. Although the new tetrahedron is contained within the original infinite tetrahedron bco, it is still a useful alternative.

missing neighbor tetrahedrons as infinite tetrahedrons. It is worth noting that the number of tetrahedrons at the boundaries is very small relative to all tetrahedrons inside the leaf. With the multiple Local Graph Iteration layers, these tetrahedrons are still able to exchange information with other neighbors inside the leaf.

1.3. Surface Extraction

We index each triangle in the Delaunay triangulation, and extract it as part of the final surface if its two neighboring tetrahedrons have labels from different classes.

2. Experiments

2.1. Metrics

In this section, we provide a detailed description of the metrics used in our experiments. On ShapeNet, we employ six metrics to conduct a comprehensive assessment, including Normal Consistency (NC), Chamfer-L_1 Distance (C-L_1), F-Score (F), IoU, Percentage of Watertight Edges (WE), and Average Number of Triangles (N.T). The evaluation procedure provided by ConvONet [18] is utilized for the first four metrics.

- NC. It is defined as the average absolute dot product of the normals in one mesh and the corresponding nearest neighbors in the other mesh.
- C-L_1 Distance. It is calculated as the mean of an accuracy metric and a completeness metric. Points are randomly sampled on both the output mesh and the ground truth mesh. The accuracy metric is defined as the distance to the nearest neighbor on the ground truth mesh for points on the output mesh, while the completeness metric is defined as the distance to the nearest neighbor on the output mesh for points on the ground truth mesh.
- F-Score. It calculates the harmonic mean between precision and recall, similar to Chamfer-L_1 Distance, but the difference is that F-Score evaluates the percentage of points that fall within a certain distance threshold.
- IoU. It is defined as the ratio of the volume of the intersection between the output mesh and the ground truth mesh to the volume of their union. It requires both the output mesh and the ground truth mesh to be watertight.
- Percentage of Watertight Edges. We take the percentage of watertight edges in the output mesh as the value of WE.

On the FAMOUSTHINGI dataset, we follow DSE [19] to use Chamfer Distance (CD), Normal Reconstruction Error (NR), and WE as evaluation metrics. Additionally, for Edge Chamfer Distance (ECD) and Edge F-Score (EFS), we use the settings of NMC[5].

- CD. It finds the nearest point in the other point cloud, and averages the square of distances up.
- NR. It measures the angle difference between the ground truth normal and the normal obtained from our reconstructed mesh at each vertex.
- ECD. It measures Chamfer Distance for points on edges. We first sample points $\mathbf{S} = \{\mathbf{s}_i\}$ uniformly on the mesh, then use the same strategy as NMC [5] to retain points on the edges. We define "sharpness" as: $\sigma(\mathbf{s}_i) = \min_{j \in \mathcal{N}_\varepsilon(\mathbf{s}_i)} |\mathbf{n}_i \cdot \mathbf{n}_j|$, where $\mathcal{N}_\varepsilon(\mathbf{s})$ extracts the indices of the samples in $\mathbf{S}$ within distance ε from $\mathbf{s}$, and $\mathbf{n}$ is the surface normal of a sample. We set $\mathbf{s} = 0.004$, and generate the edge sampling of the surface by retaining points for which $\sigma(\mathbf{s}_i) < 0.2$. Given two shapes, the ECD between them is the Chamfer Distance between the corresponding edge samplings.
- EFS. We generate the edge sampling in the way of ECD, and compute the F-Score between the corresponding edge samplings of the given two shapes as EFS. The threshold value of F-Score is set to 0.5%.

On the DTU dataset, we follow SSRNet [16] to use CD and the official evaluation method [9], which evaluates DTU Accuracy and DTU Completeness.

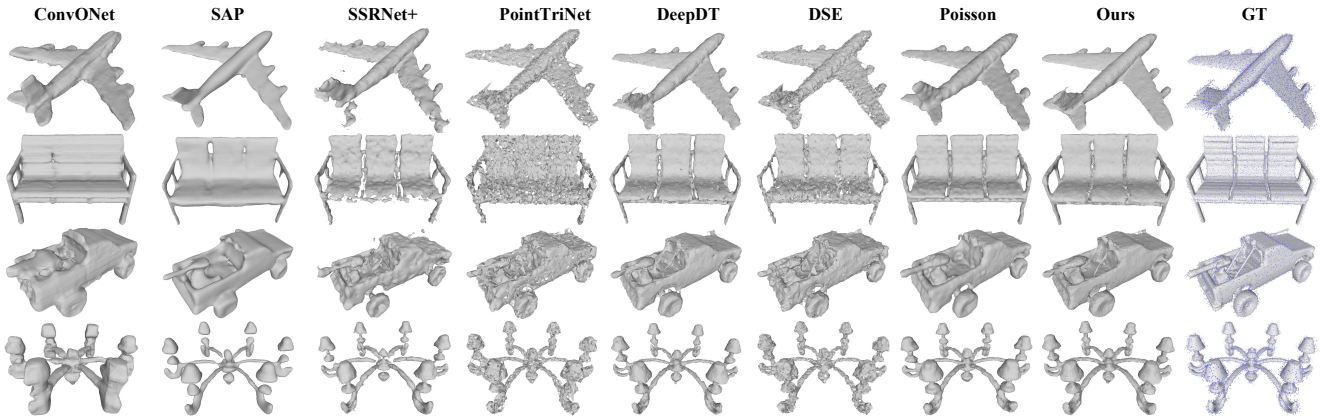

Figure 3: Comparison of reconstruction results on noisy data.

Noise	Method	C-L_1 ↓	F(0.3%) ↑	NC ↑	WE ↑	IoU ↑
0.5%	Alpha shapes [6]	0.063	0.451	0.899	**1.0**	0.557
	Ball pivoting [1]	0.040	0.585	0.928	0.802	-
	ConvONet [18]	0.066	0.367	0.915	**1.0**	0.818
	PointTriNet [21]	0.029	0.642	0.940	0.786	-
	IER Meshing [12]	0.065	0.385	0.898	0.615	-
	SAP [17]	0.033	0.629	0.947	**1.0**	0.918
	SSRNet+ [24]	0.035	0.488	0.945	0.990	-
	DSE [19]	0.028	0.659	0.941	0.947	-
	DeepDT [13]	0.031	0.700	0.947	**1.0**	0.947
	Poisson [10]	**0.025**	**0.724**	**0.959**	**1.0**	0.953
	Ours	**0.025**	0.723	0.957	**1.0**	**0.954**
1%	Alpha shapes [6]	0.100	0.241	0.890	**1.0**	0.552
	Ball pivoting [1]	0.054	0.447	0.853	0.725	-
	ConvONet [18]	0.094	0.287	0.869	**1.0**	0.730
	PointTriNet [21]	0.039	0.501	0.860	0.729	-
	IER Meshing [12]	0.094	0.199	0.883	0.489	-
	SAP [17]	0.036	0.565	0.941	**1.0**	0.902
	SSRNet+ [24]	0.050	0.272	0.928	0.989	-
	DSE [19]	0.036	0.530	0.879	0.859	-
	DeepDT [13]	0.033	0.639	0.934	**1.0**	0.926
	Poisson [10]	**0.029**	0.639	**0.949**	**1.0**	**0.931**
	Ours	**0.029**	**0.651**	0.942	**1.0**	**0.931**

Table 1: Quantitative comparison for test data with noise on ShapeNet. '-' means that the data is not evaluated due to the limitation of watertightness.

- DTU Accuracy and DTU Completeness. Both of these metrics are used to measure the distance between two point clouds, similar to the Chamfer-L_1 Distance.

2.2. Noise Resistance

Explicit methods are more sensitive to noisy data compared to implicit methods, since they derive the vertices of meshes from the input noisy points. Therefore, noisy data is a great challenge for explicit methods. To evaluate the noise resistance of various explicit methods, we apply a perturbation to ShapeNet data with $10K$ points according to a Gaussian function with 1%-bounding-box deviation as well as 0.5%-bounding-box deviation. We also apply the same perturbation to the normals of the points. To provide a comprehensive comparison between explicit methods and implicit methods, we add Poisson, SSRNet+, SAP, and ConvONet as additional benchmarks. Since explicit methods take noise points as mesh vertices, inevitably producing uneven surfaces, we add a slight Laplacian smoothing [23] to all results in this experiment. It is worth noting that the smoothing does not change the watertightness and integrity of the meshes, but only improves some visual effects. Even for Poisson, it still works well.

Table 1 presents the quantitative evaluation results, revealing that the methods for generating connected triangles experience significant performance degradation when confronted with highly noisy data, particularly in terms of Chamfer-L_1 Distance and the Percentage of Watertight Edges. Although maintaining both precision and quality of meshes poses a significant challenge for explicit methods, our DMNet effectively reduces the impact of noise while maintaining high performance on distance metrics and IoU.

Method	C-L_1 ↓	F(1%) ↑	F(0.3%) ↑	NC ↑	WE ↑	IoU ↑	N.T(k)
Alpha shapes [6]	0.057	0.839	0.598	0.879	0.989	-	15.43
Ball pivoting [1]	0.035	0.910	0.681	0.943	0.803	-	14.16
PointTriNet [21]	0.0227	0.994	0.763	0.960	0.852	-	18.85
IER Meshing [12]	0.031	0.962	0.706	0.948	0.791	-	16.95
NDC [4]	0.048	0.945	0.683	0.953	0.991	-	65.11
SSRNet+ [24]	0.031	0.981	0.569	0.956	0.984	-	59.37
DSE [19]	0.0226	0.9938	0.768	0.956	0.974	-	19.87
DeepDT [13]	0.026	0.978	0.751	0.948	0.986	-	19.25
Poisson [10]	0.024	0.9934	0.762	0.964	**1.0**	0.965	39.50
Ours	**0.0211**	**0.9974**	**0.795**	**0.969**	**1.0**	**0.981**	20.13

Table 2: Quantitative comparison for non-uniform data on ShapeNet. '-' means that the data is not evaluated due to the limitation of watertight meshes.

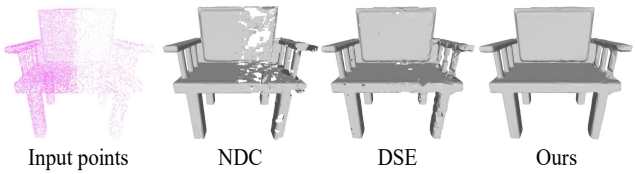

Figure 4: Visual results for data with density variation.

In the comparison, our method exhibits similar performance to Poisson, which outperforms other methods significantly, including state-of-the-art implicit methods. This suggests that methods learning the labeling of tetrahedrons from Delaunay triangulation have the potential to resist noise to a certain extent. Figure 3 presents the qualitative results for noisy data with 1%-bounding-box deviation. It can be seen that our method offers superior detail preservation.

2.3. Random Sampling

An outstanding explicit method should not only be effective for uniformly sampled data (the distances between the nearest points are approximate), but also for non-uniformly sampled data. Therefore, we randomly sample $10K$ points for each shape in ShapeNet to further test the adaptability of different explicit methods to point density variations. Additionally, we add SSRNet+, NDC, and Poisson as benchmarks to compare the performance of explicit and implicit methods. For all learning-based methods, we apply their well-trained models under uniformly sampled data. However, for DeepDT, we retrain the model due to its poor adaptability to data with varying distributions. Based on Table 2, it can be observed that all methods experience varying degrees of performance degradation when dealing with non-uniformly sampled data. While our DMNet and implicit methods exhibit smaller fluctuations compared to other explicit methods. Moreover, our method still maintains a significant advantage in terms of evaluation results, indicating the robust adaptability of our method to point density variations. Furthermore, to demonstrate this capability of our method more clearly, an example of a significant variation in the density of different parts of a point cloud is shown in Figure 4. In contrast, our method maintains a better integrity.

2.4. Comparison with Optimization-based Methods

We compare our method with some state-of-the-art optimization-based methods, such as IGR [8], Neural Pull [14], and CAP-UDF [25], to further demonstrate our outstanding detail reconstruction capability, especially for thin structures. Different from the learning-based approaches, these optimization-based methods sacrifice time for accuracy. We select ten data with complex thin structures from ShapeNet for comparison and use the default settings of these optimization-based methods. Each model is optimized tens of thousands of times, and the resolution is set to 512. For IGR, we use its version with normal supervision. Table 3 presents the quantitative comparison and our method achieves the best results on each metric. Surprisingly, our method produces a significantly lower number of triangles compared to other optimization-based methods, achieving a balance between accuracy and resolution. The visual results are presented in Figure 5. For these complex thin structures, our method still reconstructs more complete models and smooth lines without any smoothing post-processing. It is noteworthy that our method produces significantly fewer triangles, only one-ninth or even one-eightieth of the number produced by other methods. The fine-grained details are well preserved by a small number of triangles, further demonstrating the excellent reconstruction capability of our method.

Method	C-L_1 ↓	F(0.3%) ↑	NC ↑	WE ↑	IoU ↑	N.T(k)
IGR [8]	0.037	0.753	0.926	**1.0**	0.911	186.40
Neural Pull [14]	0.079	0.551	0.889	**1.0**	0.749	1358.46
CAP-UDF [25]	0.026	0.694	0.912	0.938	-	1623.38
Ours	**0.022**	**0.784**	**0.930**	**1.0**	**0.954**	20.24

Table 3: Quantitative comparison with optimization-based methods on ShapeNet. Each input point cloud has 10K points. '-' means that the data is not evaluated due to the limitation of watertight meshes.

Method	Chamfer-L_1 ↓	Chamfer-L_2 ↓	F1 (%) ↑	3D IoU (%) ↑
DefTet	0.97	0.33	96.81	87.69
DMTet	0.77	0.26	98.76	91.05
Ours	**0.27**	**0.16**	**99.38**	**93.82**

Table 4: Quantitative comparison with DMTet and DefTet.

2.5. Comparison with Methods Learning Tetrahedral Deformation

We further compare some methods combining explicit and implicit representations, such as DMTet [22] and DefTet [7]. The two methods deform the vertices of regular tetrahedrons based on the learned implicit field functions and generate an explicit mesh that aligns with the implicit representation. In contrast, our approach constructs the explicit mesh directly from the input data. Considering that the DMTet code is not released, we adopt the experimental setup outlined in the DMTet paper to ensure a fair comparison. We sample 5k points and add Gaussian noise with a standard deviation of 0.5%. The comparison results are shown in Table 4, where the results of DMTet and DefTet are obtained from the DMTet paper. Our method exhibits better reconstruction accuracy and completeness.

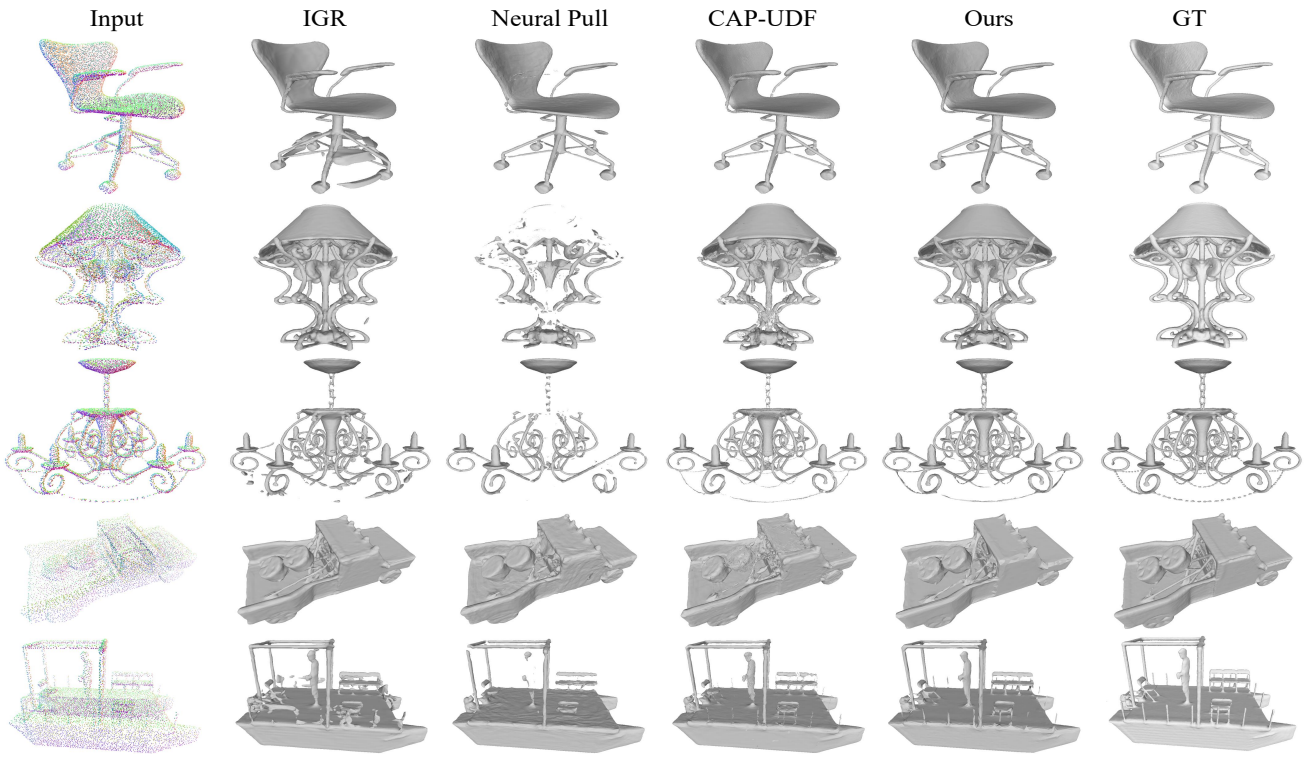

Figure 5: Qualitative comparison with optimization-based methods. Our method exhibits better model integrity and fewer artifacts than other methods.

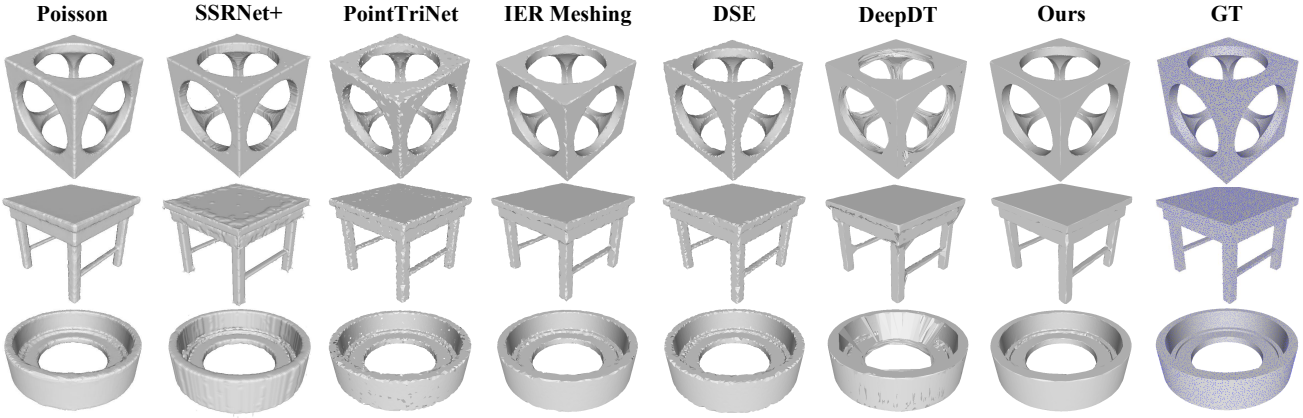

Figure 6: Visual comparison of different algorithms on the FAMOUSTHINGI dataset. Our method well preserves the sharp features.

Explicit Method	Time (s)	Memory (MB)	Implicit Method	Time (s)	Memory (MB)	Optimization-based	Time (s)	Memory (MB)
Alpha shapes [6]	0.99	-	Poisson [10]	4.56	-	IGR [8]	12250	8853
Ball pivoting [1]	0.32	-	ONet [15]	0.70	1897	Neural Pull [14]	480	1273
PointTriNet [21]	25.57	4446	ConvONet [18]	0.88	960	CAP-UDF [25]	690	1324
IER Meshing [12]	29.92	2510	SAP [17]	0.30	2245			
DSE [19]	34.14	6962	NDC [4]	6.11	3875			
DeepDT [13]	3.22	2867	SSRNet+ [24]	12.21	2957			
Ours	3.64	2935						

Table 5: Average running time and video memory occupation of different methods.

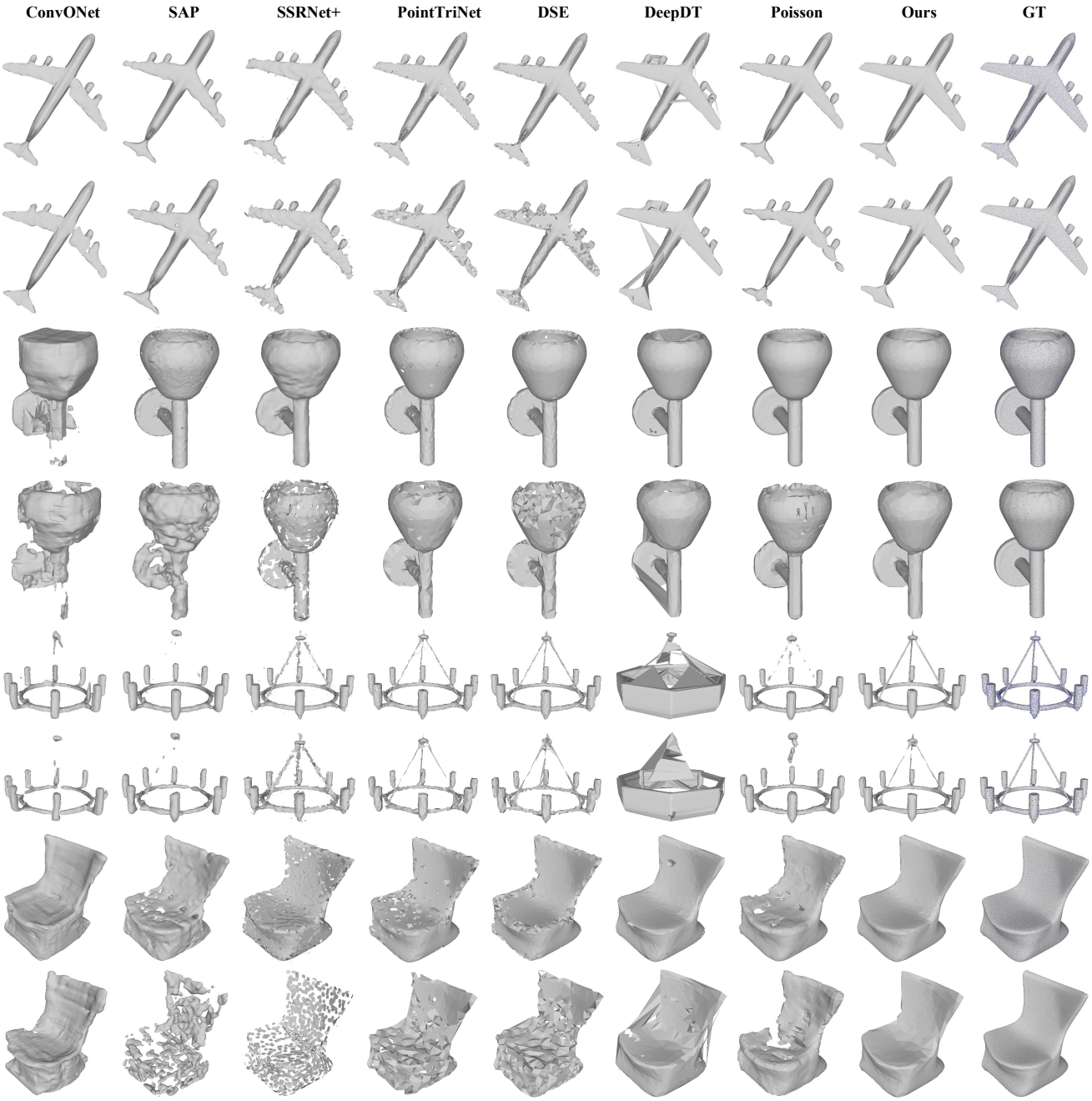

Figure 7: Comparison of reconstruction results for sparse points. For one shape, we show two rows of results, where the upper row is for $3K$ points and the lower row is for $1K$ points. Our method maintains good model integrity even as the point cloud density is significantly reduced.

2.6. Running Time and Video Memory Usage

We evaluate the efficiency and video memory usage of different competing methods on $10K$ points sampled on ShapeNet, using 1 GeForce RTX 2080 Ti GPU in an Intel Xeon(R) CPU system. The average time consumption and video memory occupation are reported in Table 5. To simplify testing, we use the CPU to evaluate the speed of geometric methods such as Poisson, Ball Pivoting, and Alpha Shapes. As for optimization-based methods, we only measure the average optimization time, excluding the time taken to extract the surface. It can be seen that learning-based explicit methods generally have high time consumption, but our method and DeepDT show strong competitiveness. This highlights the benefits of using Delaunay triangulation to balance resolution, memory usage, and time consumption. The acceptable time consumption and memory usage of our approach make it practical for daily reconstruction tasks.

2.7. More Visual Results

The visual results of different algorithms on the FAMOUSTHINGI dataset are presented in Figure 6. In contrast to the smooth edges produced by implicit methods and the jagged edges produced by other explicit methods, our method has an outstanding ability to preserve edge sharpness. Compared with DeepDT, our method significantly reduces the generation of artifacts. In addition, Figure 1 shows the visual comparison of our method with Poisson, and Figure 7 presents visual results of various methods on sparse points. Moreover, Figure 8 and Figure 9 present more results of our method on ShapeNet with $10k$ points.

3. Use of Existing Assets

3.1. Datasets

ShapeNet Dataset [3] Following ONet [15] and ConvONet [18], we use a subset consisting of 13 categories for experiments. To obtain a watertight model for sampling, we use the Poisson algorithm [10] to reconstruct the one hundred thousand points provided by ConvONet.

FAMOUSTHINGI Dataset [19] This dataset comprises 91 shapes, primarily consisting of CAD models. Their shapes differ from those in ShapeNet, and all meshes are watertight. Models in this dataset have a large number of sharp edges. We use this dataset to perform generalization experiments and test the ability to reconstruct sharp structures.

DTU Dataset [9] In contrast to the above datasets, DTU is a real-world scanned dataset consisting of 124 open scenes with millions of points. As such, it presents a unique challenge for reconstruction. We use this dataset to evaluate the ability to reconstruct large-scale real-world data. Following SSRNet+ [24], we use the Poisson surfaces with an octree depth of 10 as the ground truth surfaces.

3.2. Baselines

In our benchmark, L-Method [11], Ball pivoting [1], and Alpha shapes [6] are the classical explicit methods, and Poisson [10] is a classical implicit method that is often regarded as an excellent competitor by various approaches. In learning-based methods, ONet [15], ConvONet [18], SAP [17], NDC [4], SSRNet [16], and SSRNet+ [24] are state-of-the-art methods for learning implicit representation in different manners. PointTriNet [21], IER Meshing [12], DSE [19], and DeepDT [13] are state-of-the-art learning-based explicit methods. Next, we describe in detail the specific parameter settings of the comparison methods in our experiments. Unless otherwise specified, we use the default settings.

- Ball pivoting [1] and Alpha shapes [6]. Following the parameter settings of DSE [19], the radius of Ball pivoting is set to the bounding box diagonal divided by the square root of the vertex count. For Alpha shapes, the radius is set to 5% of the bounding box diagonal, which is reported to have good results in [19].

- L-Method [11]. We complete the implementation of this method using the program in COLMAP [20]. Due to the need of visibility information, we only compare it on DTU dataset.

- Poisson [10]. We set its maximum octree depth to 7 in our experiments on the ShapeNet dataset. We find that even if set higher, it basically makes no difference to the results. In the experiments on DTU, we follow SSRNet+ [24] to set the maximum octree depth to 9 and trim the surfaces by the official Surface Trimmer tool with a trimming value of 9.5.

Figure 8: Visual results of our method on ShapeNet.

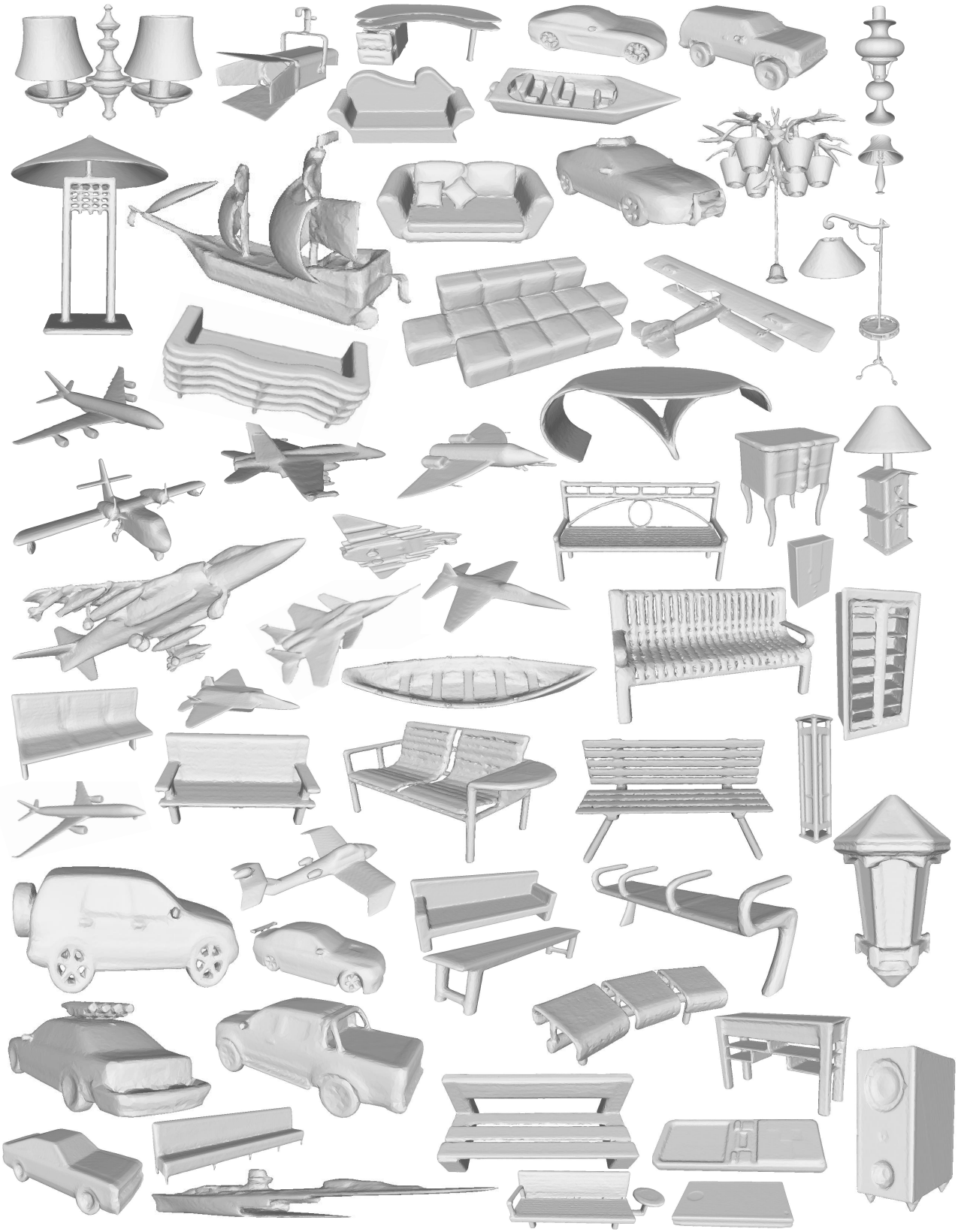

Figure 9: Visual results of our method on ShapeNet.

References

[1] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999.

[2] Jean-Daniel Boissonnat, Olivier Devillers, Monique Teillaud, and Mariette Yvinec. Triangulations in cgal. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 11–18, 2000.

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[4] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022.

[5] Zhiqin Chen and Hao Zhang. Neural marching cubes. *ACM Transactions on Graphics (TOG)*, 40(6):1–15, 2021.

[6] Herbert Edelsbrunner and Ernst P Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)*, 13(1):43–72, 1994.

[7] Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. *Advances In Neural Information Processing Systems*, 33:9936–9947, 2020.

[8] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, pages 3789–3799. PMLR, 2020.

[9] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014.

[10] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[11] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE, 2007.

[12] Minghua Liu, Xiaoshuai Zhang, and Hao Su. Meshing point clouds with predicted intrinsic-extrinsic ratio guidance. In *European Conference on Computer Vision*, pages 68–84. Springer, 2020.

[13] Yiming Luo, Zhenxing Mi, and Wenbing Tao. Deepdt: Learning geometry from delaunay triangulation for surface reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2277–2285, 2021.

[14] Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Neural-pull: Learning signed distance function from point clouds by learning to pull space onto surface. In *International Conference on Machine Learning*, pages 7246–7257. PMLR, 2021.

[15] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[16] Zhenxing Mi, Yiming Luo, and Wenbing Tao. Ssrnet: Scalable 3d surface reconstruction network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 970–979, 2020.

[17] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. *Advances in Neural Information Processing Systems*, 34:13032–13044, 2021.

[18] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020.

[19] Marie-Julie Rakotosaona, Paul Guerrero, Noam Aigerman, Niloy J Mitra, and Maks Ovsjanikov. Learning delaunay surface elements for mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22–31, 2021.

[20] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[21] Nicholas Sharp and Maks Ovsjanikov. Pointtrinet: Learned triangulation of 3d point sets. In *European Conference on Computer Vision*, pages 762–778. Springer, 2020.

[22] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021.

[23] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358, 1995.

[24] Ganzhangqin Yuan, Qiancheng Fu, Zhenxing Mi, Yiming Luo, and Wenbing Tao. Ssrnet: Scalable 3d surface reconstruction network. *IEEE Transactions on Visualization & Computer Graphics*, (01):1–14, 2022.

[25] Junsheng Zhou, Baorui Ma, Yu-Shen Liu, Yi Fang, and Zhizhong Han. Learning consistency-aware unsigned distance functions progressively from raw point clouds. In *Advances in Neural Information Processing Systems*, 2022.