GeoUDF: Surface Reconstruction from 3D Point Clouds via Geometry-guided Distance Representation (Supplementary Materials)

Siyu Ren\(^1,2\) Junhui Hou\(^1\*\) Xiaodong Chen\(^2\) Ying He\(^3\) Wenping Wang\(^4\)
\(^1\)City University of Hong Kong \(^2\)Tianjin University \(^3\)Nanyang Technological University \(^4\)Texas A&M University
siyuren2-c@my.cityu.edu.hk, jh.hou@cityu.edu.hk, xdchen@tju.edu.cn, yhe@ntu.edu.sg, wenping@cs.hku.hk

In this supplementary material, we will provide more details of our framework GeoUDF in Section 1, and more details of experiment settings and more results in Section 2. We also refer the readers to the project page.

1. More Details of GeoUDF

1.1. Local Geometry Representation

**Feature Extraction.** As shown in Fig. 14a, we employ 3-layer EdgeConv \(^{[11]}\) to embed \(\mathcal{P}\) into a high-dimensional feature space, producing hierarchical features, \(\{c^{(l)}_i \in \mathbb{R}^{d_l}\}_{i=1}^{N}, l = 1, 2, 3\), which are concatenated as the point-wise features of \(\mathcal{P}\).

**Poisson Disk Sampling over \(\mathcal{D}\).** As shown in Fig. 14b, we apply the farthest point sampling (FPS) on the uniform 2D grids to approximate Poisson Disk Sampling on the 2D area \(\mathcal{D}\) for obtaining 2D coordinates.

![Figure 14: (a) Feature Extraction. (b) Poisson Disk Sample in \(\mathcal{D}\).](image)

1.2. Geometry-guided UDF Estimation

**Discussion on the differences from IMLS-based methods DeepIMLS \(^{[6]}\) and DOG \(^{[10]}\).** In DeepIMLS \(^{[6]}\) and DOG \(^{[10]}\), the SDF of a query point is formulated as the weighted averaging of the distances of the query point to the tangent planes of a set of MLS (moving least-squares) points, i.e., Eq. (2) of DeepIMLS \(^{[6]}\) and Eq. (3) of DOG \(^{[10]}\). In the GUE module of our method (i.e., Eq. (4)), we formulate the UDF of a query point as the weighted averaging of the distances of the query point to the tangent planes of a set of **neighboring points on the surface**. Although the IMLS-based methods DeepIMLS \(^{[6]}\) and DOG \(^{[10]}\) and our method adopt similar formulas, they are **significantly** different.

- In DeepIMLS \(^{[6]}\) and DOG \(^{[10]}\), the MLS points are **NOT** distributed on surfaces, and the MLS points and their weights are **simultaneously** learned i.e., the MLP points are learned, which are then used to calculate their weights via a **pre-defined** function with a learnable factor directly (see Eq. (2) of \(^{[6]}\) or Eq. (3) of \(^{[10]}\)), resulting in that they may suffer from the **coupling issue**. **Differently**, in the GUE module of our method (i.e., Eq. (4)), **only** the weights are learned. The points used to calculate the distances (i.e., \(\{p_k\}\)) are distributed **on the surface** and obtained by an
upsampling module (i.e., LGR) pre-trained (see Lines 491 - 497 of our manuscript) under the supervision of dense point clouds. In other words, our method decouples the learning of the weights and the points.

• In DeepIMLS [6], the gradient estimation of the SDF (i.e., Eq. (7) of [6]) is part of the real derivative of the SDF (i.e., Eq. (2) of [6]), thus coupled with SDF estimation and suffering from the inaccuracy of SDF estimation. DOG [10] estimates the SDF gradients through the auto-differentiation of SDF (i.e., Eq. (3) of [10]), which is time-consuming. Differently, the GUE module of our method separates the estimation of UDFs and gradients. Specifically, based on the continuity of a surface, GUE learns another set of weights to average the aligned normals of $K$-NNs to estimate UDF gradients, i.e., Eq. (5) of our manuscript. Thus, our UDF gradient can achieve bias-free. Note that the learned weights in Eqs. (4) and (5) of our manuscript are independent.

• More importantly, we demonstrate the superiority of our method over DOG [10] both quantitatively and qualitatively in Table 1 and Fig. 7 of the manuscript. Note that according to the results reported in the original paper of DOG [10], DOG [10] achieves better performance than DeepIMLS [6].

1.3. Edge-based Marching Cube

Edge Intersection Detection. Fig. 15 shows all kinds of the position relationship between the surface and edge.

Figure 15: The position relationship between the surface and edge.

2. Experiments

2.1. Implementation Details

During training, we set the upsampling factor $M = 16$, and the boundary of $\mathcal{D} \delta = 0.1$. The ground-truth dense point clouds were obtained by sampling the corresponding mesh models through Poisson Disk Sampling [2]. For each shape, we randomly sampled $N' = 2048$ points on the surface, then added Gaussian noise with the standard deviation of 0.05, 0.02, or 0.03 to move these points away from the mesh to generate query points in each training iteration. The feature dimensions were $d_1 = 384$ and $d_2 = 128$. We used the Adam [5] optimizer to train our network and set the batch size to 16.

During inference, the threshold for edge intersection detection in Sec. 3.3 of the manuscript was set to $\tau = 5 \times 10^{-4}$, and the resolution of E-MC was 128 for the shapes and garments in the ShapeNet and MGN datasets, and 192 for the shapes or scenes in the ShapeNet car and ScanNet datasets.

2.2. Evaluation Metric

Following previous work [7, 9, 8, 1, 12], we used CD and F-Score to evaluate the reconstruction accuracy. We denote the reconstructed and ground-truth mesh by $S_{REC}$ and $S_{GT}$, on which we randomly sample a set of $10^5$ points, denoted as $P_{REC}$ and $P_{GT}$ and the CD of these two meshes is that of the two point sets, i.e.,

$$\text{CD}(S_{REC}, S_{GT}) = \text{CD}(P_{REC}, P_{GT}).$$

(15)
The F-Score is defined as the harmonic mean between the precision and the recall of points that lie within a certain distance threshold $\epsilon$ between $S_{\text{REC}}$ and $S_{\text{GT}}$.

$$F\text{-Score}(S_{\text{REC}}, S_{\text{GT}}, \epsilon) = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}},$$ (16)

where

$$\text{Recall}(S_{\text{REC}}, S_{\text{GT}}, \epsilon) = \left\{ p_1 \in P_{\text{REC}}, \text{s.t. } \min_{p_2 \in P_{\text{GT}}} \| p_1 - p_2 \| < \epsilon \right\},$$

$$\text{Precision}(S_{\text{REC}}, S_{\text{GT}}, \epsilon) = \left\{ p_2 \in P_{\text{GT}}, \text{s.t. } \min_{p_1 \in P_{\text{REC}}} \| p_1 - p_2 \| < \epsilon \right\}.$$

In the ablation study, we evaluate the accuracy of UDFs and their gradients. We first set uniform grids of $64^3$ in a unit cube, and then for each shape, we chose the grids near the surface, i.e., their UDFs are larger than $5 \times 10^{-4}$ and smaller than 0.02. We use the absolute error and angle error to measure the evaluated UDFs and their gradients of these query points.

### 2.3. Settings of the Experiments on Clean Data

In our manuscript, the methods under comparison, including ONet [7], CONet [9], SAP [8], POCO [1], and DOG [10], only conducted experiments on the noisy data in their original papers. And they did not release the pre-trained networks on clean data. Although POCO [1] released the pre-trained network on clean data, it requires ground-truth normals. Due to the different distributions of clean and noisy data, directly applying their pre-trained networks with noisy data to clean data would result in an obvious performance drop. Thus, for a fair comparison, we retrained their models on the clean data using their official codes. The released pre-trained networks of NDF [4] and GIFS [12] were only trained on the ShapeNet car dataset, rather than the whole 13 classes, and thus, we retrained them on both clean and noisy data for fair comparisons.

From the results in Table 1 of our manuscript, it can be seen that some methods, including ONet, CONet, SAP, and DOG, achieve better performance on noisy data than clean data. The possible reasons are: (1) with slight perturbation, some thin structures could be extracted through OCC or SDF; and (2) the noisy data enhance the robustness of the models during training.

### 2.4. More Experimental Results

#### Complex Shapes

Fig. 17 shows the reconstruction results of complex shapes in the ShapeNet dataset.

#### More Difficult Inputs

We also tested the performance of our GeoUDF when inputs are more difficult. Specifically, we tested the trained networks on point clouds with more severe noise (the standard deviation (STD) of Gaussian noise is set to 0.01 and 0.02), visibly non-uniform distribution, and fewer points (the number of point is 500). Fig. 18 shows our GeoUDF is much more robust than GIFS. Since our GeoUDF relies on the local geometry of input data, it would fail in those regions with extremely sparse points.

#### Error Map

For each reconstructed shape in Fig. 6 of the manuscript, we calculated the distance of its vertices to the ground-truth mesh, then used hot colormap to assign colors for the vertices. The visual results are shown in Fig. 19.

### 2.5. Ablation Study

#### Upsampling Process

As mentioned in our manuscript, the quadratic polynomial is sufficient to approximate a small surface area. To verify this, we trained a network by replacing Eq. (1) of the manuscript with the cubic polynomial as a comparison. The results are shown in the following Table 1, where it can be seen that the cubic polynomial decreases the upsampling accuracy because of overfitting.

#### Flowchart of the “Regress” Method

In Table 6 of the manuscript, we set a baseline named “Regress” to predict UDFs and gradients by using an MLP. Fig. 16 shows the detailed flowchart.
Figure 16: The network architecture of the baseline method “Regress” in Table 6 of the manuscript.

![Network Architecture Diagram](image)

Figure 17: Reconstruction results of complex shapes in the ShapeNet dataset.

![Reconstruction Results](image)

Figure 18: Visual comparisons on more difficult inputs. Each subfigure, from left to right: STD=0.01, STD=0.02, visibly non-uniform, and 500 points.

![Visual Comparisons](image)

Table 1: Comparisons of Quadratic and Cubic Polynomials.

<table>
<thead>
<tr>
<th>Method</th>
<th>CD (\times10^{-2})</th>
<th>P2F (\times10^{-3})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quadratic</td>
<td>0.245</td>
<td>0.512</td>
</tr>
<tr>
<td>Cubic</td>
<td>0.257</td>
<td>0.522</td>
</tr>
</tbody>
</table>
Figure 19: Error Maps of the shapes in the ShapeNet dataset [3].

Figure 20: More results of reconstructed shapes with multi-layer structures.
Figure 21: More results of reconstructed water-tight shapes.
Figure 22: More results of reconstructed open shapes.

Figure 23: More results of reconstructed scene-level shapes.
References


