Surface Reconstruction from Point Clouds: A Survey and a Benchmark

Zhangjin Huang*, Yuxin Wen*, Zihao Wang, Jinjuan Ren, and Kui Jia

Abstract—Reconstruction of a continuous surface of two-dimensional manifold from its raw, discrete point cloud observation is a long-standing problem in computer vision and graphics research. The problem is technically ill-posed, and becomes more difficult considering that various sensing imperfections would appear in the point clouds obtained by practical depth scanning. In literature, a rich set of methods has been proposed, and reviews of existing methods are also provided. However, existing reviews are short of thorough investigations on a common benchmark. The present paper aims to review and benchmark existing methods in the new era of deep learning surface reconstruction. To this end, we contribute a large-scale benchmarking dataset consisting of both synthetic and real-scanned data; the benchmark includes object- and scene-level surfaces and takes into account various sensing imperfections that are commonly encountered in practical depth scanning. We conduct thorough empirical studies by comparing existing methods on the constructed benchmark, and pay special attention on robustness of existing methods against various scanning imperfections; we also study how different methods generalize in terms of reconstructing complex surface shapes. Our studies help identify the best conditions under which different methods work, and suggest some empirical findings. For example, while deep learning methods are increasingly popular in the research community, our systematic studies suggest that, surprisingly, a few classical methods perform even better in terms of both robustness and generalization; our studies also suggest that the practical challenges of *misalignment of point sets from multi-view scanning*, *missing of surface points*, and *point outliers* remain unsolved by all the existing surface reconstruction methods. We expect that the benchmark and our studies would be valuable both for practitioners and as a guidance for new innovations in future research. We make the benchmark publicly accessible at https://Gorilla-Lab-SCUT.github.io/SurfaceReconstructionBenchmark.

Index Terms—Surface reconstruction, surface modeling, point cloud, benchmarking dataset, literature survey, deep learning.

1 Introduction

Modeling and reconstruction of object or scene surfaces is a fundamental problem in computer vision and graphics research. Its applications range from virtual/augmented reality, computer animation, to computer-aided design and robotics. Given that the mathematical nature of a surface shape is a continuous 2D manifold embedded in the 3D Euclidean space, different approximations are usually adopted when capturing, transmitting, and storing surface shapes, where the prominent examples include point clouds, polygon meshes, and quantized volumes. In this work, we are particularly interested in reconstructing a continuous surface from its discrete approximation of point cloud, since many depth sensors (e.g., those based on multi-view stereo, structured light, or time-of-flight measurements) produce point clouds (or equivalently, the depth maps) as their original forms of data acquisition, and surface reconstructions from the obtained point clouds are subsequently demanded for various downstream applications.

The problem is technically ill-posed — infinitely many solutions of the underlying, continuous surface may exist given an observed set of discrete points. The challenges become even severer considering that various sensing imperfections would appear during the data acquisition process; the captured point clouds could be noisy and distributed in a non-uniform manner, and they could contain outliers and/or cover less on some surface areas; when point clouds are captured at multiple views, they could be subject to less accurate alignments. All these issues pose the classical problem of surface reconstruction from point clouds as a long-standing challenge that draws continuous efforts from the research community.

In literature, a rich set of methods has been proposed for the focused studies; depending on the types of data imperfection they assume, these methods leverage various priors of surface geometry to combat the otherwise ill-posed problem.

While comprehensive reviews of these methods are given in [1], [2], [3], [4], these reviews are short of investigations and analyses on a common benchmark that could distinguish existing methods when they cope with the aforementioned data imperfections. In the meanwhile, the field has witnessed a recent surge of deep learning surface reconstruction, where models of deep networks are learned and employed to either decode surface shapes from point clouds explicitly [5], [6], [7], or generate implicit fields whose zero-level iso-surfaces can be extracted as the results of surface reconstruction [8], [9], [10]. It is thus desirable to benchmark both the classical and the more recent, deep learning solutions in order to understand their respective strengths and limitations; such investigations would be no doubt valuable for use of the appropriate methods by practitioners, and also as a guidance to new innovations in future research.

The present paper aims to provide a comprehensive review and benchmark existing methods in the new era of deep learning surface reconstruction. We organize our review by categorizing existing methods according to what priors of surface geometry they have used to regularize their reconstructions, where we include the more recent priors of deep models and deep learning, in addition to the classical, optimization-based ones. One of our key contributions is a large-scale benchmarking dataset consisting

- *Z. Huang, Y. Wen, Z. Wang and K. Jia are with the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China.*
  E-mail: {eehuangzhangjin, wen.yuxin, eezihaowang}@mail.scut.edu.cn, kuijia@scut.edu.cn
- *J. Ren is with the University of Macau, Macau, China.*
  E-mails: jinjuanren@um.edu.mo
- *Corresponding author: Kui Jia.*

** indicates equal contribution.*

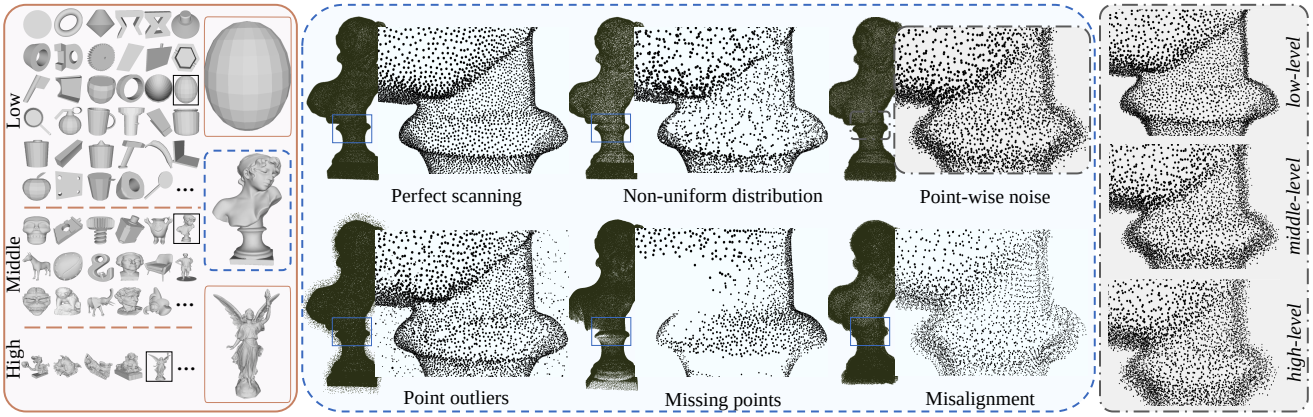


Fig. 1: An illustration of different surface complexities and scanning challenges included in our contributed benchmark. From left to right: example object surfaces of low, middle, and high complexities, the five challenges possibly encountered in practical surface scanning, and examples of different severity levels for the challenge of noisy scanning.

of both synthetic and real-scanned data (cf. Fig. 2, Fig. 3, and Fig. 5 for illustrations on how the benchmark is constructed). The benchmark includes object- and scene-level surfaces and takes into account various sensing imperfections that are commonly contained in the point clouds obtained by practical 3D scanning, such as point-wise noise, non-uniform distribution of surface points, point outliers, missing of surface points, and misalignment of point sets from multi-view scanning; Fig. 1 gives an illustration of these scanning imperfections. We conduct thorough empirical studies on the constructed benchmark, by comparing existing methods in terms of their capabilities to reconstruct surfaces from observed point clouds. We pay special attention on robustness of existing methods against various scanning imperfections; we also study how different methods generalize in terms of reconstructing complex surface shapes. Our thorough studies help identify the strengths and limitations of existing methods from multiple perspectives. We summarize a few important findings as follows.

- While many challenges of surface reconstruction from point clouds can be more or less tackled by existing methods, those of *misalignment*, *missing points*, and *outliers* have been less addressed and remain unsolved.
- Deep learning solutions have shown great promise recently for surface modeling and reconstruction; however, our systematic studies suggest that they struggle in generalizing to reconstruction of complex shapes. It is surprising that some classical methods perform even better in terms of both generalization and robustness.
- The use of surface normals is a key to the success of surface reconstruction from raw, observed point clouds, even when the surface normals are estimated less accurately; in many cases, the reconstruction result improves as long as the interior/exterior of the surface can be identified in the 3D space.
- There exist inconsistencies between different evaluation metrics, and in many cases, good quantitative results are not always concordant with the visually pleasant ones. This suggests that more foundation studies are demanded to better benchmark different methods and advance the field.

1.1 Related Works

In this section, we give a summary of existing literatures on surface reconstruction from either point clouds or other observations. We also summarize existing datasets and benchmarks that have served for advancing the field.

Surface Reconstruction from Point Clouds There exists a rich set of existing methods studying surface reconstruction from point clouds. These methods are reviewed in [1], [2], [3], [4]. Earlier reviews of [1] and [2] organize existing methods according to what functions of surface representation they use, e.g., implicit or explicit functions. More recently, existing methods are categorized in [4] based on the difference of used techniques, including interpolation and approximation techniques, learning-based techniques, and soft computing techniques. Our organization of review in the present paper is more similar to that in [3], which also organizes existing methods based on the used geometry priors. Compared with [3], our review is more comprehensive and includes the recent methods of deep learning surface reconstruction. In addition, we also compare existing methods empirically on a common benchmark, which helps identify the respective strengths and limitations of existing methods.

More General Surface Modeling and Reconstruction Surface reconstruction can be achieved from other raw observations as well, such as single- or multi-view images, motion, and/or illumination and shading. Literature reviews on the traditional methods of multi-view image reconstruction are provided in [11], [12], and [13]. A more recent survey is given in [14] on multi-view image reconstruction with deep learning. Fahim *et al.* [15] focus on a more challenging setting of deep learning surface reconstruction from as few as a single image. Zhu *et al.* [16] provide a more comprehensive survey of 3D modeling methods, including multi-view 3D reconstruction, structure from motion, and shape from shading. Different from the above reviews, the present work focuses on surface reconstruction from raw, observed point clouds, considering that depth sensors are increasingly popularly deployed in either portable or fixed devices.

Datasets and Benchmarks Existing datasets that support surface reconstruction studies are based on synthetic or real-scanned data; they may include object- and/or scene-level surfaces. For synthetic datasets, surface meshes are usually provided from which point clouds can be sampled. For example, the ShapeNet [17] and ModelNet [18] are two commonly used synthetic datasets consisting of simple, object-level shapes. More complex synthetic object surfaces are provided in the datasets of 3DNet [19], ABC [20], Thingi10k [21], and Three D Scans [22]. The datasets of SceneNet [23] and 3D-FRONT [24] provides synthetic, scene-

level surfaces. In the meanwhile, there exist datasets of real-scanned, object-level surfaces [25], [26], [27] and those of real-scanned, scene-level surfaces [28], [29]; however, due to the lack of high-precision scanning, their reconstruction ground truths are usually obtained by appropriate surface reconstruction algorithms, which jeopardizes their roles for benchmarking different methods. Most of the above datasets do not consider sensing imperfections that may appear in practically scanned point clouds, except for [30] that uses virtual scanning to simulate point cloud imperfections; however, the dataset [30] is relatively small, with only eight instances of object surfaces. In contrast, our contributed benchmarking dataset is more comprehensive, including both synthetic and real-scanned data, and covering both object- and scene-level surfaces; we intentionally inject various sensing imperfections into point cloud data of the dataset, including *point-wise noise*, *non-uniform distribution of surface points*, *point outliers*, *missing of surface points*, and *misalignment among point sets from multi-view scanning*. We expect our benchmark would facilitate more thorough studies in future research.

1.2 Contributions

As stated in Section 1, with the surge of deep learning surface reconstruction, the present paper aims to provide a comprehensive review of exiting methods in the new era, and study their respective advantages and disadvantages when reconstructing object- or scene-level surfaces from raw, observed point clouds. To this end, we contribute a large-scale benchmark consisting of both synthetic and real-scanned data. We use the constructed benchmark for systematic studies of existing methods, focusing on the robustness of these methods against various data imperfections, and also on how existing methods generalize in terms of reconstructing complex surface shapes. We summarize our key contributions as follows.

- We provide a comprehensive review of existing surface reconstruction methods, by bridging together the classical, optimization-based methods with the more recent, deep learning-based ones, where we categorize these methods according to what priors of surface geometry they have used to regularize their solutions.
- We contribute a large-scale benchmarking dataset consisting of both synthetic and real-scanned data. The point cloud data in the benchmark have various sensing imperfections that are commonly encountered in practical 3D scanning processes; these imperfections are intentionally included to benchmark the robustness of existing methods.
- We compare existing methods by conducting thorough empirical studies on the constructed benchmark. Our studies help identify the strengths and limitations of existing methods, which are valuable both for choices of appropriate methods by practitioners and for guiding the directions of new innovations in future research.

1.3 Paper Organization

The paper is organized as follows. Section 2 gives the formal definition of our studied problem. Section 3 organizes and reviews existing methods based on priors of surface geometry that they have used to regularize the reconstructions. We present our contributed large-scale benchmark in Section 4, where we give details about how we construct the benchmark and also the benchmark statistics; we make the benchmark, our construction manner of the benchmark, and also implementation codes of representative methods publicly accessible at https://Gorilla-Lab-SCUT.github.io/SurfaceReconstructionBenchmark. Experimental setups of our empirical studies are given in Section 5, before results, analyses, and important insights are presented in Section 6. We finally draw the paper conclusion in Section 7.

2 Problem Statement

Consider a discrete point set $\mathcal{P}$ that may be obtained by scanning an object or scene surface using some 3D sensing devices; each $\boldsymbol{p} \in \mathbb{R}^3$ of its contained points collects the coordinates in the Euclidean space. Our goal of interest is to recover its underlying, continuous surface $\mathcal{S}^*$ from which the points $\{\boldsymbol{p} \in \mathcal{P}\}$ are practically observed. Given that recovering the continuous $\mathcal{S}^*$ from the discrete $\mathcal{P}$ is an ill-posed problem, an appropriate regularization must be imposed in order to recover a geometry-aware approximation $\mathcal{S}$, e.g., a smooth and/or fair surface [31]. This formally amounts to solving the following regularized optimization

$$\min_{\mathcal{S}} L(\mathcal{S}; \mathcal{P}) + \lambda R(\mathcal{S}), \tag{1}$$

where L is a loss term for data fidelity to the observed $\mathcal{P}$, R is a regularizer that constrains the solution with a certain prior of surface geometry, and λ is a scalar penalty. Note that the objective (1) is only in an abstract form, since it is difficult to define both L and R directly on $\mathcal{S}$. In practice, one may represent a surface either explicitly as a parametric mapping $\boldsymbol{f} : \Omega^2 \to \mathcal{S}$, where $\Omega^2 \subset \mathbb{R}^2$ denotes the 2D domain and $\mathcal{S} = \{\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^3\}$ with $\boldsymbol{x} \in \Omega^2$, or implicitly as the zero-level set of an implicit function $F : \mathbb{R}^3 \to \mathbb{R}$, i.e., $\mathcal{S} = \{\boldsymbol{q} \in \mathbb{R}^3 | F(\boldsymbol{q}) = 0\}$. Correspondingly, one can instantiate the objective (1) as the following one that pursues $\mathcal{S}$ by optimizing an explicit mapping

$$\min_{\boldsymbol{f} \in \mathcal{H}_{\boldsymbol{f}}} L^{\text{exp}}(\boldsymbol{f}; \mathcal{P}) + \lambda R^{\text{exp}}(\boldsymbol{f}), \tag{2}$$

where L^{exp} and R^{exp} are respectively the instantiated loss function and regularizer, and $\boldsymbol{f}$ is optionally constrained in a hypothesis space $\mathcal{H}_{\boldsymbol{f}}$ (e.g., by choosing $\boldsymbol{f}$ as a neural network). Alternatively, one may instantiate the objective (1) as the following one that pursues an implicit representation of $\mathcal{S}$

$$\min_{F \in \mathcal{H}_F} L^{\text{imp}}(F; \mathcal{P}) + \lambda R^{\text{imp}}(F) \text{ s.t. } F(\boldsymbol{q}) = 0 \; \forall \, \boldsymbol{q} \in \mathcal{S}, \tag{3}$$

where L^{imp} and R^{imp} are again the instantiated functions, and $\mathcal{H}_F$ denotes a hypothesis space that optionally constrains the implicit function F. We present the subsequent sections based on the notations defined in Table 1, unless specified otherwise.

3 Surface Reconstruction with a Categorization of Geometric Priors

Given an observed point set $\mathcal{P}$, a reconstructed surface $\mathcal{S}$ should be close to $\mathcal{P}$ under some distance metric; this is guaranteed by the first term $L(\mathcal{S}; \mathcal{P})$ of data fidelity in the abstract objective (1). Section 2 also suggests that $L(\mathcal{S}; \mathcal{P})$ can be instantiated either explicitly or implicitly. The explicit form is generally written as

$$L^{\text{exp}}(\boldsymbol{f}; \mathcal{P}) = \frac{1}{n_{\mathcal{P}}} \sum_{\boldsymbol{p} \in \mathcal{P}} \min_{\boldsymbol{x} \in \Omega^2} \|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{p}\|_\ell, \tag{4}$$

where $\|\cdot\|_\ell$ denotes a proper norm of distance, with ℓ typically set as 1 or 2; the above term (4) constrains the learning of mapping

TABLE 1: Math notations.

Notation	Description
$\mathcal{S}^*$	An underlying surface to be recovered; $\mathcal{S}^* \subset \mathbb{R}^3$.
$\mathcal{S}$	A reconstructed surface; $\mathcal{S} \subset \mathbb{R}^3$.
$\mathcal{G}_\mathcal{S}$	A triangular mesh representation of surface $\mathcal{S}$. A mesh with $n_\mathcal{G}$ faces is collectively written as $\mathcal{G}_\mathcal{S} \triangleq \{\mathcal{T}_i\}_{i=1}^{n_\mathcal{G}}$, where each face $\mathcal{T}$ is specified by $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$ containing three vertices; we also write as $\boldsymbol{e}_{ij}$ for the edge connecting vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$.
$\mathcal{P}$	A set of $n_\mathcal{P}$ discrete points $\{\boldsymbol{p}_i \in \mathbb{R}^3\}_{i=1}^{n_\mathcal{P}}$, representing the practical sampling of an underlying surface $\mathcal{S}^*$.
$\boldsymbol{n_p}$	An estimated, oriented surface normal defined at a surface point $\boldsymbol{p}$; $\boldsymbol{n_p} \in \mathbb{R}^3$.
$\mathcal{N}(\boldsymbol{p})$	A local neighborhood of points centered at $\boldsymbol{p}$.
Ω	A domain of subset space, *e.g.* , $\Omega^k \subset \mathbb{R}^k$.
$\mathcal{C}^k$	The smoothness of a function, where k is the number of continuous derivatives the function has over some domain.
$d(\boldsymbol{p})$	A signed or unsigned distance field value at a point $\boldsymbol{p}$.
$\boldsymbol{f_\theta}$ or $F_\boldsymbol{\theta}$	An explicit or implicit model of surface reconstruction parameterized by $\boldsymbol{\theta}$ (e.g., a neural network); the parameters are also denoted as $\boldsymbol{\theta_f}$ or $\boldsymbol{\theta}_F$.
$\nabla_x \boldsymbol{f_\theta}$ or $\nabla_x F_\boldsymbol{\theta}$	Model derivative with respect to x.
$L(\mathcal{S}; \mathcal{P})$	Loss function of a reconstructed surface $\mathcal{S}$ for data fidelity to an observed $\mathcal{P}$.
$R(\mathcal{S})$	Regularizer imposed on a reconstructed surface $\mathcal{S}$.
$\boldsymbol{K}$	Extrinsic matrix of a camera.

function $\boldsymbol{f} \in \mathcal{H}_\boldsymbol{f}$. In practice, one may sample a fixed set $\{\boldsymbol{x} \in \Omega^2\}$ instead of optimizing over the whole domain Ω^2, which gives variants of Eq. (4) based on point-set distances, such as Chamfer or Hausdorff distances. An implicit form of $L(\mathcal{S}; \mathcal{P})$ is generally written as

$$L^{\text{imp}}(F; \mathcal{P}) = \frac{1}{n_\mathcal{P}} \sum_{\boldsymbol{p} \in \mathcal{P}} \|F(\boldsymbol{p})\|_\ell, \tag{5}$$

which learns the implicit function $F \in \mathcal{H}_F$ by minimizing a proper norm of $F(\boldsymbol{p})$ for any $\boldsymbol{p} \in \mathcal{P}$. Advanced versions of the implicit data fidelity loss exist, e.g.,

$$\begin{aligned} L^{\text{imp++}}(F; \mathcal{P}) = \alpha_1 \mathbb{E}_{\boldsymbol{q} \in \mathbb{R}^3} \|F(\boldsymbol{q}) - d(\boldsymbol{q}; \mathcal{P})\|_{\ell_1} + \\ \alpha_2 \mathbb{E}_{\boldsymbol{q} \in \mathbb{R}^3} \|\nabla_\boldsymbol{q} F(\boldsymbol{q}) - \boldsymbol{n}(\boldsymbol{q}; \mathcal{P})\|_{\ell_2} + \cdots, \end{aligned} \tag{6}$$

where when F models a Signed Distance Function (SDF) [8], $d(\boldsymbol{q}; \mathcal{P})$ denotes the signed distance between any space point $\boldsymbol{q} \in \mathbb{R}^3$ and the observed point set $\mathcal{P}$, which vanishes when $\boldsymbol{q}$ hits any $\boldsymbol{p} \in \mathcal{P}$, and when F models an Occupancy Field (OF) [9], $d(\boldsymbol{q}; \mathcal{P}) \in \{0, 1\}$ depending on whether $\boldsymbol{q}$ is inside or outside the surface $\mathcal{S}$, which is practically estimated by comparing $\boldsymbol{q}$ with the observed $\mathcal{P}$; $\boldsymbol{n}(\boldsymbol{q}; \mathcal{P})$ denotes the normal at point $\boldsymbol{q}$, which, when $\boldsymbol{q}$ hits some $\boldsymbol{p} \in \mathcal{P}$, can be estimated by computing the local tangent plane of $\mathcal{P}$ at $\boldsymbol{p}$, and $\|\boldsymbol{n}(\boldsymbol{q}; \mathcal{P})\|_2 = 1$ otherwise (when F models an SDF, the second term in Eq. (6) is correspondingly written as $\mathbb{E}_{\boldsymbol{q} \in \mathbb{R}^3} |\|\nabla_\boldsymbol{q} F(\boldsymbol{q})\|_2 - 1|$); one may use $\{\alpha_1, \alpha_2, \cdots\}$ to weight or switch on/off different terms in Eq. (6).

Due to the ill-posed nature of surface reconstruction from $\mathcal{P}$, neither of the data fidelity loss terms (4) or (5) is sufficient to reconstruct a geometry-plausible $\mathcal{S}$. In literature, various instantiations of the regularization $R(\mathcal{S})$ in Eq. (1) have been proposed, in order to make the problem be better posed. In the remainder of this section, we discuss the essence of existing surface reconstruction methods by categorizing their adopted regularization of geometric priors, including *triangulation-based prior*, *smoothness prior*, *template-based prior*, *modeling prior*, *learning-based prior*, and *hybrid prior*, where we include both classical and the recent, deep learning ones. We expect our categorization and discussion would foster new innovations by bridging classical surface reconstruction methods with the more recent deep learning solutions.

3.1 Triangulation-based Prior

A closed surface is continuous and could be locally differentiable up to different orders. When the $\mathcal{S}$ to be recovered from $\mathcal{P}$ is locally differentiable up to the first order, i.e., $\boldsymbol{f}$ is locally of $\mathcal{C}^1$, a piecewise linear assumption stands as a good prior for modeling the surface, which gives a *mesh* representation of the surface. Among various meshing schemes, Delaunay triangulation [32] is a classical one that approximates $\mathcal{S}$ as a mesh that satisfies

$$\begin{aligned} \mathcal{G}_\mathcal{S} = \{\mathcal{T}_i\}_{i=1}^{n_\mathcal{G}} \\ \text{s.t.} \quad \boldsymbol{v}_1^\mathcal{T} \in \mathcal{P}, \boldsymbol{v}_2^\mathcal{T} \in \mathcal{P}, \boldsymbol{v}_3^\mathcal{T} \in \mathcal{P} \; \forall \, \mathcal{T} \in \mathcal{G}_\mathcal{S}, \\ \boldsymbol{p} \notin \text{CC}(\mathcal{T}) \; \forall \, \boldsymbol{p} \in \mathcal{P} / \{\boldsymbol{v}_1^\mathcal{T}, \boldsymbol{v}_2^\mathcal{T}, \boldsymbol{v}_3^\mathcal{T}\} \; \wedge \; \forall \, \mathcal{T} \in \mathcal{G}_\mathcal{S}, \end{aligned} \tag{7}$$

where $\mathcal{T}$ is a triangular face with its three vertices denoted as $\{\boldsymbol{v}_1^\mathcal{T}, \boldsymbol{v}_2^\mathcal{T}, \boldsymbol{v}_3^\mathcal{T}\}$, $n_\mathcal{G}$ is the number of faces, and $\text{CC}(\mathcal{T})$ denotes the space enclosed by the circumcircle of $\mathcal{T}$ passing through its three vertices; note that by Delaunay triangulation, the explicit data fidelity Eq. (4) is satisfied simultaneously.

Representative methods [33], [34] of Delaunay triangulation first generate a set of triangular faces directly from the observed $\mathcal{P}$, and then select the optimal subset from them to generate the final triangular mesh. Greedy Delaunay (GD) [33] proposes greedy algorithm based on topological constraints to select valid triangles sequentially, where the initial triangles are generated by Delaunay triangulation; Ball-Pivoting Algorithm (BPA) [34] uses balls with various radii rolling over the points in $\mathcal{P}$ to generate triangles, where every three points touched by a rolling ball will construct a new triangle if the triangle does not encompass any other points, which can also be regarded as an approximation of Delaunay triangulation.

3.2 Surface Smoothness Priors

In more general cases, the surface $\mathcal{S}$ to be recovered is expected to be *smooth* or continuously differentiable up to a certain order [31]. Surface smoothness is usually enforced by the following two manners.

Given that the observed points in $\mathcal{P}$ could be noisy, the first manner smoothes out $\{\boldsymbol{p} \in \mathcal{P}\}$ via local weighted combination when fitting the explicit mapping function $\boldsymbol{f}$, resulting in an *regularized* version of Eq. (4) as

$$\begin{aligned} \min_\boldsymbol{f} L^{\text{exp}}(\boldsymbol{f}; \mathcal{P}) = \frac{1}{n_\mathcal{P}} \sum_{\boldsymbol{p} \in \mathcal{P}} \min_{\boldsymbol{x} \in \Omega^2} \|\boldsymbol{f}(\boldsymbol{x}) - \hat{\boldsymbol{p}}(\boldsymbol{p})\|_2^2 \\ \text{s.t. } \hat{\boldsymbol{p}}(\boldsymbol{p}) = \sum_{\boldsymbol{p}' \in \mathcal{N}(\boldsymbol{p})} \boldsymbol{p}' / g(\|\boldsymbol{p} - \boldsymbol{p}'\|_2), \end{aligned} \tag{8}$$

where $\mathcal{N}(\boldsymbol{p})$ denotes a local neighborhood of the observed $\boldsymbol{p}$, containing $\{\boldsymbol{p}' \in \mathcal{P}\}$, and $g(\|\boldsymbol{p} - \boldsymbol{p}'\|_2)$ denotes a function (e.g., a Radial Basis Function or RBF) whose value is proportional to the distance $\|\boldsymbol{p} - \boldsymbol{p}'\|_2$. A similar implicit objective exists by regularizing the second-order term in Eq. (6), giving rise to the method of Poisson Surface Reconstruction (PSR) [35]

$$\begin{aligned} \min_F L^{\text{PSR}}(F; \mathcal{P}) = \mathbb{E}_{\boldsymbol{q} \in \mathbb{R}^3} \|\nabla_\boldsymbol{q} F(\boldsymbol{q}) - \hat{\boldsymbol{n}}(\boldsymbol{q}; \mathcal{P})\|_2^2 \\ \text{s.t. } \hat{\boldsymbol{n}}(\boldsymbol{q}; \mathcal{P}) = \sum_{\boldsymbol{p}' \in \mathcal{N}(\boldsymbol{q})} \boldsymbol{n}(\boldsymbol{p}'; \mathcal{P}) / g(\|\boldsymbol{q} - \boldsymbol{p}'\|_2), \end{aligned} \tag{9}$$

where $\mathcal{N}(\boldsymbol{q})$ denotes a local neighborhood of a space point $\boldsymbol{q} \in \mathbb{R}^3$, containing observed points $\{\boldsymbol{p}' \in \mathcal{P}\}$. PSR constrains the normal field only and usually produces over-smooth results. As a remedy, Screened Poisson Surface Reconstruction (SPSR) [36] improves over PSR by incorporating regularized versions of both the first- and second-order terms in Eq. (6). More recently, Shape As Points [37] develops a differentiable Poisson solver in a spectral manner, which enables an end-to-end optimization and thus could be integrated into the learning of deep neural networks.

The second manner achieves surface smoothness by constraining the function complexities of $\boldsymbol{f}$ or F. This can be equivalently achieved by constraining the hypothesis space of $\mathcal{H}_{\boldsymbol{f}}$ or $\mathcal{H}_F$, e.g., by constraining $\mathcal{H}_{\boldsymbol{f}}$ as B-spline functions [38], [39], [40] or NURBS [41], [42]. Intuitively, a more complex function is able to fit a surface of complex geometry; but it also tends to be overfitted to the observed $\mathcal{P}$, producing a less smooth surface. For example, using RBFs in [43] means that the approximate function is in the form of low-degree polynomials with an interpolation of many basic functions centered at the observed points. The works [44], [45] are similar to [43] but approximate their respective implicit field functions using different basis functions. More specifically, Kazhdan [44] firstly computes the Fourier coefficients of its implicit field function with the help of Monte-Carlo approximation of Divergence Theorem, and then uses inverse Fourier transform to obtain the implicit function for extraction of iso-surface; a regular grid is needed to perform fast Fourier transform in [44], and instead Manson *et al.* [45] use wavelets, which provide a localized, multi-resolution representation of the implicit function.

Methods such as Point Set Surfaces (PSS) [46], [47], [48], [49] combine both of the above smoothing strategies. PSS is derived from Moving Least Squares (MLS) [50], [51], [52], [53], whose surface can be defined either by an explicit function with stationary projection operator [46], [47], [50], [51], [48] or by an implicit function [52], [53], [49]. In either case, a weighted combination of spatially-varying low-degree polynomials acts as the most important ingredient to locally approximate the observed points and construct the surfaces. In Simple Point Set Surfaces (SPSS) [48], the authors iteratively project all the given points along the normal directions onto the local reference planes, which are defined by a weighted average of the points to be projected and their neighborhood points; then the local reference plane at each evaluation point would give a local orthogonal coordinate system to compute a local bivariate polynomial approximation to the surface. However, the local reference plane can hardly be a good approximation and sometimes even becomes unstable when the observed points are sparse. Algebraic Point Set Surface (APSS) [49] overcomes this issue by using an algebraic sphere to fit the observed points, which forms an implicit function to represent the algebraic distance between the evaluation point and the fitted sphere; the fitting problem is then solved by a least squares problem between the gradient field of the algebraic sphere and the normals of the observed points. In Robust Implicit MLS (RIMLS) [53], the authors combine kernel regression and statistical robustness with MLS to cope with the limitation that MLS can only reconstruct smooth surfaces. Other PSS methods share the same principle; more details can be found in [54] and [3].

3.3 Template-based Priors

Template-based priors assume that a surface could be represented by combination of a group of templates, where the templates could be geometric primitives such as spheres or cubes, or complex ones from an auxiliary dataset. As such, surface reconstruction boils down as the problem of estimating and fitting the correct templates to the observed $\mathcal{P}$. This can be formally written as

$$\min_{\{w\},\{\boldsymbol{\theta}\}} D\left(\sum_{i=1}^{|\{\mathcal{M}\}|} w_i \mathcal{M}_i(\boldsymbol{\theta}_i), \mathcal{P}\right), \tag{10}$$

where $\{\mathcal{M}\}$ denotes the set of predefined templates, and each template $\mathcal{M}$ is parameterized by (the possibly learnable) $\boldsymbol{\theta}$ and weighted by w; $D(\cdot,\cdot)$ denotes a proper distance between the formed surface and the observed $\mathcal{P}$. By minimizing the fitting error (10), a surface is reconstructed by the determined $\{w\}$ and/or $\{\boldsymbol{\theta}\}$.

3.3.1 Geometric Primitives

Templates of geometric primitives are simple shapes that can be analytically represented by a certain number of parameters, *e.g.* , cuboids, spheres, cylinders, cones, etc. Random sample consensus (RANSAC) [55], [56] is the most commonly used method to solve Eq. (10) that fits the right templates to the observed $\mathcal{P}$, where $D(\cdot,\cdot)$ sums up the element-wise (Euclidean) distances between randomly sampled $\{\boldsymbol{p} \in \mathcal{P}\}$ and their projections onto the fitted templates. More specifically, Schnabel *et al.* [55] introduce an efficient RANSAC-based algorithm with a novel sampling strategy and an efficient score evaluation scheme, where the primitive with maximum score is extracted iteratively. Nan and Wonka [56] propose to only extract planar primitives based on RANSAC, obtaining the final surface by minimizing a weighted sum of several energy terms to select the optimal set of faces.

3.3.2 Retrieval-based Templates

Given an auxiliary set $\{\mathcal{M}\}$ of shape models, retrieving-and-deforming methods solve Eq. (10) to find the closest shapes and deform them, via optimization of model parameters $\{\boldsymbol{\theta}\}$, to fit the observed $\mathcal{P}$. For example, in scene reconstruction [57], [58], [59], [60], the observed scene points are segmented into semantic classes, each of which is then fit with a retrieved shape model followed by rigid or non-rigid deformation; in object reconstruction, Pauly *et al.* [61] warp and blend multiple retrieved object shapes to conform with the observed points, and Shen *et al.* [62] retrieve individual object parts to form a surface by part assembly.

3.4 Modeling Priors

While constraining the complexity of hypothesis space of $\mathcal{H}_{\boldsymbol{f}}$ or $\mathcal{H}_F$ would promote reconstruction of smoother surfaces, as discussed in Section 3.2, the choice of $\mathcal{H}_{\boldsymbol{f}}$ or $\mathcal{H}_F$ itself regularizes the reconstruction given that only specific types of surface can be modeled by the choice. A prominent example is the recent trend of using deep neural networks for geometric modeling and surface reconstruction. We term the geometric priors provided by the respective designs of models themselves as *modeling priors*.

Motivated from deep image prior [63], Deep Geometric Prior (DGP) [64] verifies the efficacy of deep networks as a prior for geometric surface modeling, *even when the networks are not trained*. Latter on, Point2Mesh [65] and SAIL-S3 [66] extend the global modeling adopted in [64] as local ones, where the former constructs its local, implicit functions as a weight-shared MeshCNN [67] and the later method constructs them as a weight-shared Multi-Layer Perceptron (MLP). Deep Manifold Prior [68]

delivers mathematical analyses on such modeling properties for MLP as well as convolutional networks. There have also been a few works making use of modeling priors while not explicitly mentioning it. Atzmon *et al.* [69] theoretically prove that MLP with Rectified Linear Units (ReLUs) generate piecewise linear surfaces, and a meshing algorithm of Analytic Marching is also proposed in [70] that is able to analytically compute the piecewise linear surface mesh from such a network, both of which deliver mathematical analyses on how the structure of MLP itself acts as a regularizer. Later on, Deep Manifold Prior [68] extends such modeling properties for both MLP and convolutional networks. Implicit Geometric Regularization (IGR) [71] shows that additional regularization on the gradient of neural network would further encourage the generated surfaces to be smooth. Sign Agnostic Learning (SAL) [72] and its variants [73], [74] study reconstructing a surface from an un-oriented point cloud via a specially initialized neural network, and Davies *et al.* [75] adopt the same strategy for surface reconstruction from an oriented point cloud. Neural Splines [76] performs reconstruction based on random feature kernels arising from infinitely-wide shallow ReLU networks and shows that such solutions bias toward reconstruction of smooth surface.

3.5 Learning-based Priors

Given the parametrization of $\boldsymbol{\theta}_{\boldsymbol{f}}$ for $\boldsymbol{f} \in \mathcal{H}_{\boldsymbol{f}}$ and $\boldsymbol{\theta}_F$ for $F \in \mathcal{H}_F$, priors can be learned from an auxiliary set of training shapes as *optimized model parameters*. Note that such learning-based priors are different from modeling priors presented in the preceding section, where priors are provided by the models themselves. Let $\{\mathcal{P}_i, \mathcal{S}_i^*\}_{i=1}^N$ be the training set containing N pairs of observed point sets and their corresponding ground-truth surfaces. By adapting the objective (2), an explicit surface reconstruction based on a learned prior can be generally written as

$$\min_{\boldsymbol{z}} L^{\text{exp}}\left(\boldsymbol{f}(\boldsymbol{z}; \tilde{\boldsymbol{\theta}}_{\boldsymbol{f}}); \mathcal{P}\right) + \lambda R^{\text{exp}}(\boldsymbol{z}) \tag{11}$$

$$\text{s.t. } \tilde{\boldsymbol{\theta}}_{\boldsymbol{f}} = \arg\min_{\boldsymbol{\theta}_{\boldsymbol{f}}} \frac{1}{N} \sum_{i=1}^{N} \tilde{L}^{\text{exp}}\left(\boldsymbol{\theta}_{\boldsymbol{f}}; \{\mathcal{P}_i, \mathcal{S}_i^*\}\right) + \tilde{\lambda} \tilde{R}^{\text{exp}}(\boldsymbol{\theta}_{\boldsymbol{f}}), \tag{12}$$

where the constraint (12) learns the prior as the optimized model parameter $\tilde{\boldsymbol{\theta}}_{\boldsymbol{f}}$, $\tilde{L}^{\text{exp}}$ could be different from L^{exp}, and an objective for smooth and/or fair surface may also be incorporated into $\tilde{R}^{\text{exp}}$ in addition to a simple norm constraint of $\boldsymbol{\theta}_{\boldsymbol{f}}$; given the learned and then fixed $\tilde{\boldsymbol{\theta}}_{\boldsymbol{f}}$, the objective (11) fits the model prediction to any observed $\mathcal{P}$ by optimizing a latent code $\boldsymbol{z}$, where norm of $\boldsymbol{z}$ is usually penalized to prevent overfitting. Learning a prior for implicit surface reconstruction can be similarly written as follows, by adapting the objective (3)

$$\min_{\boldsymbol{z}} L^{\text{imp}}\left(F(\boldsymbol{z}; \tilde{\boldsymbol{\theta}}_F); \mathcal{P}\right) + \lambda R^{\text{imp}}(\boldsymbol{z}) \tag{13}$$
$$\text{s.t. } F(\boldsymbol{q}, \boldsymbol{z}; \tilde{\boldsymbol{\theta}}_F) = 0 \;\forall\, \boldsymbol{q} \in \mathcal{S} \quad \text{and}$$

$$\tilde{\boldsymbol{\theta}}_F = \arg\min_{\boldsymbol{\theta}_F} \frac{1}{N} \sum_{i=1}^{N} \tilde{L}^{\text{imp}}\left(\boldsymbol{\theta}_F; \{\mathcal{P}_i, \mathcal{S}_i^*\}\right) + \tilde{\lambda} \tilde{R}^{\text{imp}}(\boldsymbol{\theta}_F). \tag{14}$$

The model $\boldsymbol{f}$ can also be constructed as an auto-encoder architecture [9], [77], [78], i.e., $\boldsymbol{f} = \boldsymbol{f}_{\text{encoder}} \circ \boldsymbol{f}_{\text{decoder}}$. In such a case, given the prior $\tilde{\boldsymbol{\theta}}_{\boldsymbol{f}} = \{\tilde{\boldsymbol{\theta}}_{\boldsymbol{f}_{\text{encoder}}}, \tilde{\boldsymbol{\theta}}_{\boldsymbol{f}_{\text{decoder}}}\}$ already learned by Eq. (12), a latent code can be directly obtained as

$$\boldsymbol{z} = \boldsymbol{f}_{\text{encoder}}(\mathcal{P}; \tilde{\boldsymbol{\theta}}_{\boldsymbol{f}_{\text{encoder}}}), \tag{15}$$

instead of optimizing the objective (11). The above applies to an auto-encoder based implicit function $F = F_{\text{encoder}} \circ F_{\text{decoder}}$ as well, and given the learned prior $\tilde{\boldsymbol{\theta}}_F = \{\tilde{\boldsymbol{\theta}}_{F_{\text{encoder}}}, \tilde{\boldsymbol{\theta}}_{F_{\text{decoder}}}\}$, one can compute its latent code directly as

$$\boldsymbol{z} = F_{\text{encoder}}(\mathcal{P}; \tilde{\boldsymbol{\theta}}_{F_{\text{encoder}}}). \tag{16}$$

Note that the most recent implicit methods [79], [80] allow the priors of model parameters to be further optimized when fitting to the observed points, i.e., optimizing over both the latent code and model parameters in Eq. (11) or Eq. (13), and thus potentially bridge the gap between the learning-based priors and classical ones.

Depending on how the training shapes in $\{\mathcal{P}_i, \mathcal{S}_i^*\}_{i=1}^N$ are organized, the priors learned via Eq. (12) or Eq. (14) may be either at a global, semantic level or as local, shape primitives. For example, when pairs in $\{\mathcal{P}_i, \mathcal{S}_i^*\}_{i=1}^N$ capture surfaces of object instances belonging to a semantic category, the learned priors would encode shape patterns common to this object category. To learn priors for reconstruction of arbitrary surface shapes, one may have to prepare $\{\mathcal{P}_i, \mathcal{S}_i^*\}_{i=1}^N$ as those encoding surface patches, and expect a global surface of arbitrary shape can be better reconstructed by providing priors on its local shape primitives.

3.5.1 Learning Semantic Priors

The recent trend of deep learning surface reconstruction starts from learning deep priors at the semantic, object-level. For example, an explicit method of Deep Marching Cubes [81] proposes a novel, differentiable layer of marching cubes, which connects mesh surface generation with the learning of semantic prior via a shape encoding network that generates displaced voxel grids. The seminal deep learning-based implicit methods [9], [82], [8] model either SDF or OF via deep neural networks. Specifically, OccNet [9] and IM-Net [82] adopt an auto-encoder structure; after training, the encoder generates the latent code representing the shape via a single forward propagation in Eq. (16) for any observed point set, and the decoder predicts the probability of occupancy according to the given space point along with the latent code. Different from OccNet and IM-Net, DeepSDF [8] adopts the structure of decoder-only model, which optimizes the latent code of the given point clouds via maximum a posteriori in Eq. (13), and predicts signed distances according to the given space point along with the latent code. Curriculum DeepSDF [83] improves DeepSDF by introducing a progressive learning strategy to learn local details; in the meanwhile, Yao *et al.* [84] implement such learning using Graph Neural Networks, which converts the global, semantic latent code into more sophisticated local ones, before feeding into the decoder. MeshUDF [85] further extends DeepSDF to reconstruct open surfaces, which predicts unsigned distances of any given space point and generates the surface using their customized marching cubes. More recently, a few methods [86], [87] adopt networks based on transformer [88], [89] to help reconstruction. Note also that learning semantic priors enables reconstruction of surfaces from raw observations that are originally of no or less 3D shape information (e.g., as few as a single RGB image [5], [90], [91]), by training encoders that learn latent shape spaces from such observations.

3.5.2 Learning Priors as Local, Shape Primitives

To reconstruct a scene surface or surface of an object that cannot be semantically categorized, existing methods resort to modeling

and learning local priors of shape primitives either at regular grids that partition the 3D space [92], [77], [93], [94], [10], [95], [96], or on local patches along the surface manifold [97], [98], [78]. Among the former methods, Implicit Feature Networks (IF-Net) [92] and Convolutional OccNet [77] (ConvOccNet) obtain latent codes for local grids via auto-encoder structure, where IF-Net [92] adopts 3D convolution that convolves each input point with its surrounding points to get the latent code at each local grid, and ConvOccNet [77] uses PointNet [99] as its encoder to get the latent code for each point and then encapsulates all the latent codes into a volumetric feature via average pooling. Later on, Chibane *et al.* [93] extend IF-Net [92] to support sign-agnostic learning. Deep Local Shape (DeepLS) [94] divides the whole 3D space into regular voxels, and trains an implicit function whose parameters are shared among different local voxels. Local Implicit Grid (LIG) [10] trains an auto-encoder to extract the latent codes of local voxels during training (via Eq. (16)), while retaining the voxel-shared decoder with fixed parameters during inference only. To improve the efficiency of local encoding, Scalable Surface Reconstruction Network (SSRNet) [95] makes use of octree to partition the whole 3D space, and uses fully-convolutional U-shaped network with skip connections, which is based on modified tangent convolution [100] to get the latent code for each local grid. Neural Geometric Level of Detail (LOD) [96] adopts the structure of sparse octree to further improve the efficiency, where latent codes are computed only for local voxels that intersect with the surface. Ummenhofer *et al.* [101] aggregate latent codes for local grids via adaptive grid convolution on multiple levels of the octree input space. As for the methods based on local surface patches, Badki *et al.* [97] introduce the concept of meshlets, and train a variational auto-encoder to learn the latent space of pose-disentangled meshlets; by back-propagating the error with respect to the given points and the meshlets, the method updates the meshlets' latent codes and deforms the meshlets to fit the given points at inference time. PatchNets [98] leverages the structure of implicit auto-decoder to learn across different local surface patches, and the decoder is also trained with an elaborate loss function to ensure the smoothness of the reconstructed patches. Points2Surf [78] learns features from both local patches and the global surface, and reconstructs the surface with an implicit decoder, where the former takes a local encoder to learn the absolute distance of a queried point from the local surfaces, and the latter learns the interior/exterior of the surface with a global encoder. POCO [102] and [103] encodes each input point into a single latent code, and then perform weighted interpolation among a point and its neighboring ones to get the local latent code.

3.6 Hybrid Priors

Methods discussed in the preceding sections are organized according to their respectively used, main priors of surface geometry. To improve the plausibility of reconstructed surfaces, existing methods usually combine multiple priors, e.g., by combining smoothness priors with triangulation or template-based ones [104], [105], [106], by imposing additional priors on top of modeling ones [107], [108], [109], [110], [111], [112], [113], or by learning priors to improve over regularization provided by pre-defined ones [6], [7], [114], [115], [116], [40]. In this section, we focus our discussion on the last case, considering that such hybrid priors are popularly used in the new era of deep learning surface reconstruction.

Given auxiliary sets of training shapes, learning-based priors are combined with triangulation-based prior in [7], [117], [114] that encourage deep networks to learn particular properties for triangulation from the observed point clouds. More specifically, PointTriNet [7] introduces a framework to generate triangles from observed point clouds directly, where a proposal network suggests candidates of triangles and a classification network predicts whether a proposed candidate should appear in the reconstructed surface or not, and two networks iteratively take effects until the final surface of triangular mesh is generated. Liu *et al.* [117] show that connecting those pairs of vertices whose geodesic distance approaches to their Euclidean distance can approximately reconstruct the underlying surface; they construct a network to select candidate triangles satisfying this property. Delaunay Surface Elements (DSE) [114] introduces small triangulated patches generated by combining Delaunay triangulation with learned logarithmic maps, from which candidate triangles of the output surface are then iteratively selected via their proposed adaptive voting algorithm. DeepDT [6] learns deep networks to extract geometric features from observed point clouds, and then integrates the learned features together with graph structural information to vote for the inside/outside labels of Delaunay tetrahedrons; the surface is reconstructed by extracting triangular faces between tetrahedrons of different labels. Gao *et al.* [118] learn a network to predict the offsets of vertices of tetrahedrons and a second network to predict the occupancy of each tetrahedron, and object surface is generated on the facet that belongs to two tetrahedrons with different occupancies.

Learning-based priors are commonly combined with smoothness priors (e.g., the Laplacian regularization in [119] and the cosine smoothness in [120]). IMLSNet [115] trains an implicit network that evaluates signed distances at grids of an octree-based 3D space, where smoothness is regularized by defining the signed distances in a way similar to implicit moving least-squares [53]. Xiao *et al.* [121] learn a network to implicitly model an indicator function derived from Gauss lemma, which is smooth and linearly approximates the surface; similar to Points2Surf [78], they learn both local and global features.

There have also been plenty of works [116], [40], [122] combining learning-based priors with template-based ones. Li *et al.* [116] propose to train a deep network to fit geometric primitives according to the observed point clouds; the trained network predicts point-wise features that are fed into a differentiable estimator for algebraic computation of primitive parameters. ParseNet [40] uses a neural decomposition module to partition an observed point cloud into multiple subsets, each of which is assumed to be a primitive type modeled as an open or close B-spline by a deep network. Local Deep Implicit Function (LDIF) [122] represents a surface shape as a set of shape elements, each of which is parameterized by analytic shape variables and latent shape codes; the method learns such variables and latent codes by its SIF encoder and PointNet encoder respectively. For reconstruction of a large-scale scene surface, RetrievalFuse [123] retrieves a set of object templates from an auxiliary scene dataset, and learns a network with attention-based refinement to produce the reconstruction.

4 A Surface Reconstruction Benchmark

As discussed in Section 3, a rich set of methods exist that aim to address the ill-posed problem of surface reconstruction from

point observations. These methods have their respective merits, yet it is less clear on their advatanges/disandvatanges under different working conditions, due to the lack of a comprehensive surface reconstruction benchmark that identifies and includes the main challenges faced by the studied problem. In this work, we contribute such a benchmark by both *synthesizing* and *practically scanning* point clouds of object and scene surfaces. We identify the main challenges of surface reconstruction from point clouds obtained by imperfect surface scanning, including *point-wise noise*, *point outliers*, *non-uniform distribution* of points, *misalignment* among point sets obtained by scanning different but overlapped, partial surfaces of an object or scene, and *missing points* of one or several surface patches. An illustration of such challenges is given in Fig. 1. We include all the five challenges in synthetic data of the benchmark, and expect that an arbitrary combination of these challenges may appear in any sample of the real-scanned data. The benchmark is organized as the synthetic data of object surfaces, the synthetic data of scene surfaces, and the data of real-scanned surfaces. Tables 3 and 4 summarize the statistics. We make the benchmark publicly accessible at https://Gorilla-Lab-SCUT.github.io/SurfaceReconstructionBenchmark, where we also release the code implementation of our synthetic scanning pipeline to facilitate future research in the community.

4.1 The Synthetic Data of Object Surfaces

To prepare synthetic data of object instances in the benchmark, we collect CAD models from existing repositories [21], [19], [20], [22], and pre-process them before synthetically scanning their point clouds. For each CAD model, we simulate the aforementioned five ways of imperfect scanning, in addition to a perfect scanning that gives a clean object point cloud; we in total have six kinds of scanned point clouds for each instance. We organize the collected object instances into three levels of (algebraic) surface complexity [124], in order to study how different methods perform on surface reconstruction of varying complexities.

4.1.1 Data Collection

We collect CAD models of object instances from the existing repositories of Thingi10k [21], ABC [20], 3DNet Cat200 subset [19], and Three D Scans [22]. More specifically, we randomly select $3,000$ objects from Thingi10k [21] (a dataset collected from online-shared 3D printing models), $3,000$ industrial components from the set of Chunk 0080 in ABC [20], $3,400$ commodities from 3DNet Cat200 subset [19], and 106 art sculptures from Three D Scans [22]. All these instances are represented in the form of triangular mesh. As illustrated in Fig. 2, some meshes of the collected instances could be non-watertight, with self-occlusion, and/or topologically too complex, and are thus less convenient to be synthetically scanned to simulate practical sensing conditions; we filter out these instances by determining their states of being watertight and 2D-manifold [125], checking self-occlusion [126], and calculating their surface genus [31]. We finally normalize each mesh of the remaining instances by centering it at the origin and scaling it isotropically to fit into the unit sphere. We obtain a total of $1,620$ object surface meshes in the benchmark.

4.1.2 Groups of Varying Surface Complexities

It is possible that performance of different methods depends on the complexities of object surfaces to be reconstructed. To identify better working conditions for different methods, we intend to divide the above processed object surface meshes into groups of varying complexities. Surface complexities can be measured under different metrics, among which algebraic complexity and topological complexity are the measures more relevant to surface reconstruction from point clouds [124]; simply put, the former measures the degree of polynomials needed to represent a surface, and the latter can be measured as the surface genus (e.g., the number of holes on the surface). Given that our instances of surface meshes have been processed to satisfy the conditions of being watertight, 2D manifold, and topologically simpler (equal to or smaller than genus 5), we divide all the $1,620$ instances into three groups of *low-*, *middle-*, and *high-complexity* based on the measure of algebraic complexity.

The algebraic complexity of a surface is usually computed as the highest degree of polynomial functions that approximate/fit local patches on the surface [124]. However, given fixed budget of approximation errors, it is usually unstable to fit local surface patches with polynomials of high degrees [50], causing inaccurate prediction of polynomial degrees and thus that of algebraic complexity. Instead, we take the strategy of fixing the highest function degree and measuring the averaged approximation errors of local surface patches; technical details are given in Appendix A. We finally obtain the three groups respectively of 972 instances, 486 instances, and 162 instances (at the ratio of around $6:3:1$ for low-, middle-, and high-complexity groups). Table 2 summarizes the group statistics.

TABLE 2: The benchmark collection of synthetic object instances from existing repositories and their distributions in the three groups of low, middle, and high complexities.

Surface complexity	Low	Middle	High	Total
Thingi10k [21]	516	230	97	845
3DNet Cat200 subset [19]	144	89	33	266
ABC [20]	312	160	6	478
Three D Scans [22]	0	7	26	31
Our benchmark	972	486	162	1,620

4.1.3 Synthetic Point Cloud Scanning

We use Blender Sensor Simulation Toolbox (BlenSor) [127] to synthetically scan our collected surface meshes of object instances. We describe in this section our scanning pipeline, including how we implement different ways of imperfect scanning that simulate point cloud sensing happening in practical conditions.

Perfect scanning – We first show our way of perfect scanning to give a whole picture of how we conduct our virtual scanning pipline as shown in Fig. 2. As described in Section 4.1.1, our instances of surface meshes have been normalized at the center of a unit sphere in the simulator. To scan an instance, we place a virtual time-of-flight (TOF) camera [128] on viewing spheres whose radii range from $r_{\text{min}} = 2.5$ to $r_{\text{max}} = 3.5$; a viewpoint on any of the spheres can be specified as the camera extrinsic $\boldsymbol{K} = [\boldsymbol{R}|\boldsymbol{t}] \in \mathbb{R}^{4\times4}$, where the rotation $\boldsymbol{R} \in \mathbb{R}^{4\times3}$ and translation $\boldsymbol{t} \in \mathbb{R}^{4\times1}$ together specify how the camera is positioned. Denote as $\mathcal{P}_{\boldsymbol{K}}$ the point cloud obtained by scanning from the viewpoint $\boldsymbol{K}$; one may transform it into the world coordinate system as $\boldsymbol{K} \circ \mathcal{P}_{\boldsymbol{K}}$, where $\circ$ denotes an operator[1]. In our setting, we sample $1,000$ viewpoints on the spheres of different radii, resulting in a collection of scanned point clouds

[1]Note that a point $\boldsymbol{p} \in \mathcal{P}_{\boldsymbol{K}}$ in the camera coordinate system can be transformed to point $\boldsymbol{p}^{\text{world}} \in \mathcal{P}^{\text{world}}_{\boldsymbol{K}}$ in the world coordinate system in terms of a homogeneous equation as $[\boldsymbol{p}^{\text{world}}; 1]^\top = [\boldsymbol{p}; 1]^\top \boldsymbol{K}^{-1}$; here we transform $\mathcal{P}_{\boldsymbol{K}}$ to $\mathcal{P}^{\text{world}}_{\boldsymbol{K}}$ and write collectively as $\mathcal{P}^{\text{world}}_{\boldsymbol{K}} = \boldsymbol{K} \circ \mathcal{P}_{\boldsymbol{K}}$.

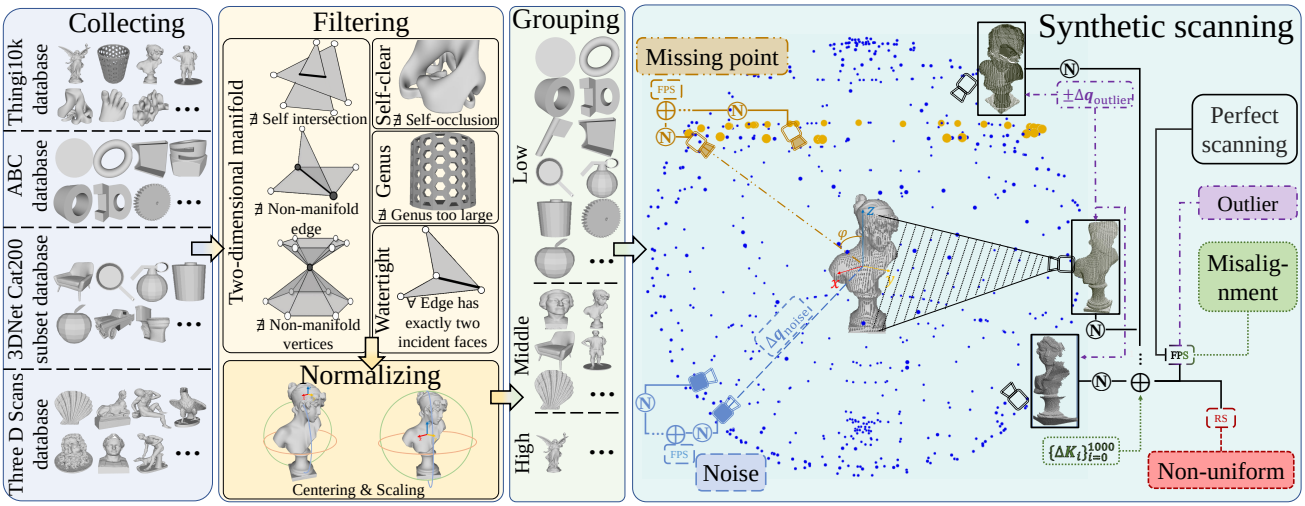


Fig. 2: The pipeline of constructing our synthetic object-level dataset (cf. Section 4.1 for the details). We firstly collect CAD models of object surface from four repositories, as shown in the leftmost of the figure. We then filter the collected surfaces by dropping those failing to meet our requirements, and normalize the remaining ones, as shown in the middle of the figure. We further organize the normalized object surfaces into three groups of low, middle, and high complexities at a ratio of around $6:3:1$, based on the criterion of (algebraic) surface complexity. We finally perform synthetic scanning as shown in the right of the figure, where different challenges possibly encountered in practical scanning are simulated. Blue spots indicate the scanning positions on viewing spheres; after estimating oriented normals (shown as N in the figure), we make registration between different viewpoints of scanning and get the final scanned point cloud, where farthest point sampling (FPS) is used for sampling a fixed number of points per point cloud (except otherwise mentioned cases in which random sampling (RS) is used).

$\{\boldsymbol{K}_1 \circ \mathcal{P}_{\boldsymbol{K}_1}, \cdots, \boldsymbol{K}_{1000} \circ \mathcal{P}_{\boldsymbol{K}_{1000}}\}$, each of which partially covers the surface. Then we can register and fuse different partial point clouds to cover the complete surface $\bigcup_{i=1}^{1000} \boldsymbol{K}_i \circ \mathcal{P}_{\boldsymbol{K}_i}$. We obtain the final, uniformly distributed point cloud scanning $\mathcal{P}$ by applying Farthest Point Sampling (FPS) [99] to $\bigcup_{i=1}^{1000} \boldsymbol{K_i} \circ \mathcal{P}_{\boldsymbol{K_i}}$; $\mathcal{P}$ is set to contain 80k points for low-complexity surfaces, 120k points for middle-complexity surfaces, and 160k points for high-complexity surfaces.

Since some of existing methods require surface normals to preform reconstruction, we compute the oriented surface normals as follows. For any point $\boldsymbol{p} \in \mathcal{P}$, we first compute its un-oriented normal $\bar{\boldsymbol{n}}_{\boldsymbol{p}}$ by performing PCA on the local neighborhood constructed by $k = 40$ nearest neighbors of the point; orientation of $\bar{\boldsymbol{n}}_{\boldsymbol{p}}$ can be simply determined by comparing $\boldsymbol{p}$ with the camera position, giving rise to the oriented normal $\boldsymbol{n}_{\boldsymbol{p}}$.

Point-wise noise – Due to sensor noise, ambient noise, reflective nature of the surface, and the incapable precision of the scanning devices, point clouds from practical scanning are inevitably noisy. In this case, each scanned point is not exactly on the underlying surface, deviating away from the surface in a point-wise, independent manner; and to simulate such noise, we add point-wise perturbations to the points obtained by the aforementioned perfect scanning. Specifically, for any surface point $\boldsymbol{q}$, we generate $\Delta\boldsymbol{q}_{\text{noise}} \in \mathbb{R}^3$ by randomly sampling its element values from a Gaussian distribution $\mathcal{N}(0, \sigma^2_{\text{noise}})$ with a truncated values $[-2\sigma_{\text{noise}}, 2\sigma_{\text{noise}}]$. Given a point $\boldsymbol{p} \in \mathcal{P}$, the corresponding noisy point from noisy scanning is obtained as $\boldsymbol{p} = \boldsymbol{q} + \Delta\boldsymbol{q}_{\text{noise}}$. We set σ_{noise} respectively as 0.001, 0.003, and 0.006 in our benchmark to simulate different severity levels of point-wise noise. Note that the truncation above is to prevent individual $\{\boldsymbol{p} \in \mathcal{P}\}$ from deviating too far away from the surface, which would become point outliers to be discussed shortly.

Non-uniform distribution of points – Practical scanning often produces a point cloud whose points are not uniformly distributed over the surface. For example, the surface patches that are scanned for multiple times (possibly from different viewpoints) would have more points, and a closer scanning position would produce denser points as well. To simulate such phenomena, we replace the final step of uniformity-promoting FPS in perfect scanning with Random Sampling (RS), and local point densities of the resulting $\mathcal{P}$ would be less uniform over the surface.

Point outliers – As mentioned above, outliers of a surface point cloud are defined as those deviating far away from the surface. They are often caused by impulsive noise of practical scanning. We simulate such outliers as follows. Given a point cloud obtained by perfect scanning, we first randomly sample a ratio r_{outlier} of its points, and for any sampled point $\boldsymbol{q}$ that is on the surface, we generate $\Delta\boldsymbol{q}_{\text{outlier}} \in \mathbb{R}^3$ by randomly sampling its element values from a uniform distribution $\mathcal{U}[a_{\text{outlier}}, b_{\text{outlier}}]$; the corresponding point outlier $\boldsymbol{p}_{\text{outlier}} \in \mathcal{P}$ is then obtained as $\boldsymbol{p} = \boldsymbol{q} \pm \Delta\boldsymbol{q}_{\text{outlier}}$. We set $a_{\text{outlier}} = 0.01$ to distinguish point outliers from noisy points and set $b_{\text{outlier}} = 0.1$ to prevent the outliers from being less relevantly distancing. The ratio r_{outlier} is respectively set as 0.1%, 0.3%, and 0.6% for varying numbers of outliers in each obtained $\mathcal{P}$.

Misalignment – As described for perfect scanning, a scanning viewpoint is specified by the camera extrinsic $\boldsymbol{K} = [\boldsymbol{R}|\boldsymbol{t}]$; scanning from the viewpoint $\boldsymbol{K}$ would produce a point cloud $\mathcal{P}_{\boldsymbol{K}}$ that covers the surface partially; a complete point cloud is obtained by scanning from $1,000$ viewpoints and then registering and fusing the obtained point clouds as $\bigcup_{i=1}^{1000} \boldsymbol{K}_i \circ \mathcal{P}_{\boldsymbol{K}_i}$. However, misalignment would happen when the camera extrinsics are less accurate. To simulate such a misalignment, for each viewpoint $\boldsymbol{K}$, we generate the perturbation $\Delta\boldsymbol{K} = [\Delta\boldsymbol{R}|\Delta\boldsymbol{t}] \in \mathbb{R}^{4\times4}$ where $\Delta\boldsymbol{R} \in \mathbb{R}^{4\times3}$ is obtained by the XYZ Euler angle convention [129], i.e., $\Delta\boldsymbol{R} = [\boldsymbol{R}_x(\alpha)\boldsymbol{R}_y(\beta)\boldsymbol{R}_z(\gamma); \boldsymbol{0}_3^\top]$ with α, β, and γ uniformly sampled from $\mathcal{U}[a_{\text{rotation}}, b_{\text{rotation}}]$ and those of

the translation perturbation $\Delta \boldsymbol{t} \in \mathbb{R}^{4\times 1}$ uniformly sampled from $\mathcal{U}[a_{\text{translation}}, b_{\text{translation}}]$. The final point cloud $\mathcal{P}$ with misalignment is obtained by applying FPS to $\bigcup_{i=1}^{1000}(\boldsymbol{K}_i + \Delta\boldsymbol{K}_i) \circ \mathcal{P}_{\boldsymbol{K}_i}$. We set $[a_{\text{rotation}}, b_{\text{rotation}}]$ as $[-0.5°, 0.5°]$, $[-1°, 1°]$, and $[-2°, 2°]$, and set $[a_{\text{translation}}, b_{\text{translation}}]$ as $[-0.005, 0.005]$, $[-0.01, 0.01]$, and $[-0.02, 0.02]$, which are respectively for different severities of misalignment.

Missing points – Due to surface reflection, self-occlusion, and/or simply insufficient covering of the surface, practical scanning often produces a point cloud that does not cover the whole object surface of interest. Surface reflection depends on a mixed effect of lighting and surface material, and the latter is not included in our collected CAD models. Instead, we take the following simple approach in our benchmark to simulate missing surface points. Rather than allowing the scanning viewpoint $\boldsymbol{K}_i$ to locate on the whole viewing spheres as in perfect scanning (shown as the blue points in Fig. 2), we only allow it to locate on a limited number of viewing positions to simulate missing points. Specifically, we define the viewing positions in a few narrow bands of trajectories with the polar angle of $\varphi \pm \varphi_\Delta$ (shown as the yellow points in Fig. 2). For different severities, we respectively set the number of trajectories to be 3, 2, and 1, with the polar angle φ set to be $[20°, 40°, 60°]$, $[20°, 40°]$, and $[20°]$ respectively, and with $\varphi_\Delta = 3°$. The scanned surface areas approximately cover 99%, 94%, and 86% of the whole surface for different severities.

4.2 The Synthetic Data of Scene Surfaces

We adopt a pipeline similar to that presented in Section 4.1 for synthetic scanning of the scene-level data. Since the scale of a scene surface is larger and practical scanning usually produces a point cloud that includes multiple types of imperfections. As such, we include all the five challenges of point-wise noise, point outliers, non-uniform distribution of points, misalignment, and missing points into our synthetic scanning of a single scene. Fig. 3 gives the illustration.

Data collection – We choose to collect CAD surface models of indoor scenes in the benchmark. To form the collection, we randomly select 4 indoor scenes from SceneNet [23], 4 from 3D-FRONT [24], and 2 from Replica [28]. The scenes from the three datasets are diverse in terms of varying room types, varying room sizes, and the contained different furniture and furnishings. The collected surfaces are represented in the form of triangular mesh as well. We normalize each scene mesh by centering it at the origin but keeping its original size.

Synthetic scanning – We still use BlenSor [127] to perform our synthetical scene scanning. In practical scanning of indoor scenes, the scanner is usually placed at a medium distancing from the scene surface; we thus choose the sensor of Kinect V2 [130] whose working distance ranges from 0.75m to 2.1m. We prepare the scanning by placing each surface mesh of indoor scene in a bounded 3D space, where the scene center has been aligned at the origin and the space size is set to be just enclosing the scene surface (cf. Fig. 3). We firstly partition the 3D space into the volume of 1m^3-sized, 0.5m^3-overlapped cubes; some of the cubes would be empty while others contain certain patches of the scene surface. We choose the centers of those empty cubes as the positions from which the virtual camera views the scene, and abandon others to ensure that the working distance between the camera and scene surface satisfy the aforementioned Kinect V2 requirements. From each position, the camera could view a certain patch of the scene surface towards arbitrary directions on a viewing sphere originated at the cube center. More specifically, for a center of empty cube positioned at $\boldsymbol{x} \in \mathbb{R}^3$, we make it homogeneous in the form of $\boldsymbol{t}_i = [\boldsymbol{x}; 1] \in \mathbb{R}^4$ and randomly sample 100 directions to get the camera extrinsics $\{\boldsymbol{K}_{ij} = [\boldsymbol{R}_j|\boldsymbol{t}_i]\}_{j=1}^{100}$, each of which would be used to generate a point cloud $\mathcal{P}_{\boldsymbol{K}_{ij}}$ covering a certain patch of the scene surface. Similar to object scanning, a final point cloud $\mathcal{P}$ is obtained by registering and fusing all the point clouds obtained by the preceding scanning $\{\boldsymbol{K}_{ij} \circ \mathcal{P}_{\boldsymbol{K}_{ij}}\}_{i,j}$. Note that we do not conduct FPS as object-level synthetic scanning does; and consequently, the challenge of non-uniform distribution of points naturally appears here. Since some methods require surface normals, we compute the oriented normal $\boldsymbol{n_p}$ for all the point $\boldsymbol{p} \in \mathcal{P}$ in the same way as object scanning does. Apart from the natural challenges of non-uniform distribution and missing points (due to self-occlusion), we also simulate the other challenges of point-wise noise, point outliers, and misalignment in the same way as described in Section 4.1.3. Specifically, following the settings of Kinect V2 camera, we set $\sigma_{\text{noise}} = 0.005$m to control the level of point-wise noise, set $a_{\text{outlier}} = 0.01$m, $b_{\text{outlier}} = 0.1$m, and the ratio $r_{\text{outlier}} = 0.4\%$ to control the level of outliers, and set $[a_{\text{rotation}}, b_{\text{rotation}}]$ as $[-1.5°, 1.5°]$ and $[a_{\text{translation}}, b_{\text{translation}}]$ as $[-0.015\text{m}, 0.015\text{m}]$ to control the level of misalignment. Such a point cloud $\mathcal{P}$ contains around one million points containing all the five challenges.

TABLE 3: Statistics of synthetic data in the benchmark. Elements in each triple $\cdot$ / $\cdot$ / $\cdot$ represent the hyper-parameters that control the scanning with three levels of severity; # denotes the number. (cf. Sections 4.1 and 4.2 for specific meanings of the math notations.)

		Object level		Scene level
normalization $[r_{\text{min}}, r_{\text{max}}]$ (camera distancing)		centering+scaling [2.5, 3.5]		centering [0.75m, 2.1m]
point-wise noise	σ_{noise}	0.001/0.003/0.006		0.005m
point outliers	$[a_{\text{outlier}}, b_{\text{outlier}}]$	[0.01, 0.1]		[0.01m, 0.1m]
	r_{outlier}	0.1%/0.3%/0.6%		0.4%
missing points	φ	$[20°, 40°, 60°]/[20°, 40°]/[20°]$		-
	$\Delta\varphi$	3°		
misalignment	$[a_{\text{rotation}}, b_{\text{rotation}}]$	$[-0.5°, 0.5°]/[-1°, 1°]/[-2°, 2°]$		$[-1.5°, 1.5°]$
	$[a_{\text{translation}}, b_{\text{translation}}]$	$[-0.005, 0.005]/[-0.01, 0.01]/[-0.02, 0.02]$		$[-0.015\text{m}, 0.015\text{m}]$
#viewpoints		1000		$100/\text{m}^3$
#scanned points		low complexity	80k	1000k
		middle complexity	120k	
		high complexity	160k	
#surfaces		1620		10

4.3 The Real-scanned Data

We provide real-scanned data of the benchmark by scanning real object instances via two depth cameras of varying precisions. To scan an object, we use SHINING 3D Einscan SE [2] whose precision is of 100 micrometers to get the input point cloud, and use SHINING 3D OKIO 5M [3] whose precision is of 5 micrometers to get the approximate ground-truth.[4]

Data collection – We collect 20 object instances of varying surface complexities, including commodities, instruments, and artwares, and also of varying materials, including metal, plastic, ceramic, and cloth; this is to ensure that various sensing imperfections would appear in the obtained point clouds. Fig. 5 shows these objects.

[2] https://www.einscan.com/desktop-3d-scanners/einscan-se/

[3] https://www.shining3d.com/solutions/optimscan-5m

[4] Our surface reconstruction evaluation is based on metrics of point set distances (cf. Section 5.3), for which we obtain the ground-truth point clouds by sampling from the corresponding surface meshes. We thus choose to directly use the raw point clouds scanned by SHINING 3D OKIO 5M, instead of converting the scanned point clouds as surface meshes using its in-built software.

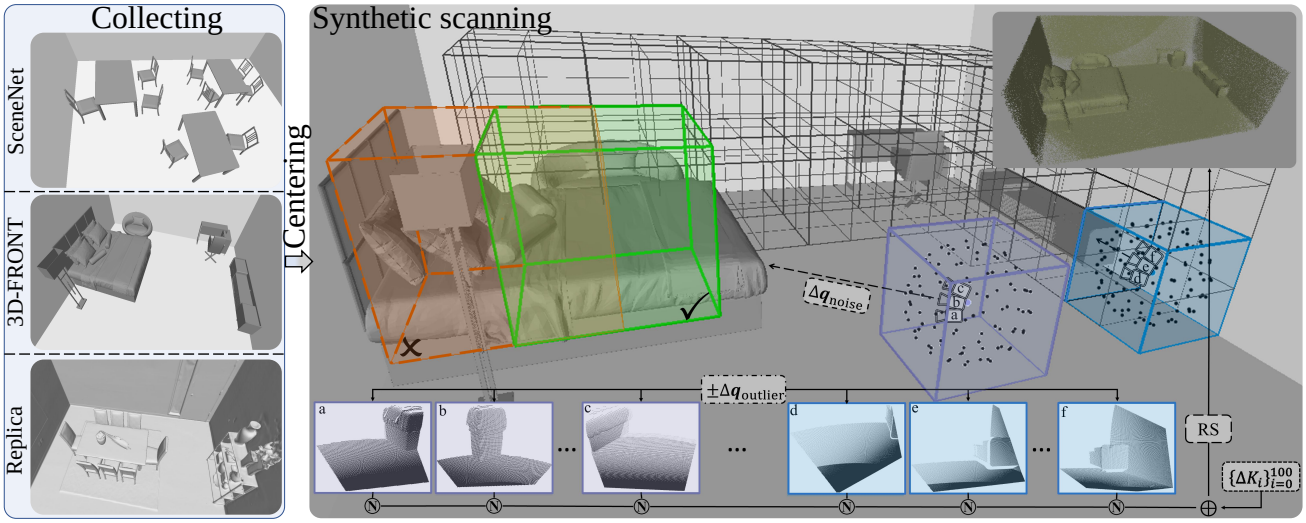


Fig. 3: The pipeline of constructing our synthetic scene-level dataset (cf. Section 4.2 for the details). We collect CAD surface models of indoor scenes from three datasets, as shown in the left of the figure, and normalize them by centering at the origin in the 3D space. We partition the 3D space into overlapped cubes, as shown in the right of the figure; some of the cubes contain certain patches of the scene surface, while others are empty. The camera is positioned at centers of those empty cubes (e.g., the purple and blue cubes); for each positioned camera, we randomly sample 100 viewpoints for the scanning. We include all the five challenges of practical scanning into each scanning of the scene surface, whose individual implementations are similar to those for synthetic object scanning.

Real scanning – In general, better scanning results could be obtained by increasing the numbers of scanning shots from multiple viewpoints, as empirically verified in Fig. 4. Since qualities of the scanned point clouds start to saturate at around 40 shots for Einscan SE and 20 shots for OKIO 5M, we conduct 40 shots for Einscan SE and 20 shots for OKIO 5M shots when scanning an object. After scanning an object, we use CloudCompare [131] to align different point clouds obtained from different shots. Table 4 gives the statistics of our real-scanned data in the benchmark. Some scanned pairs from the two scanners are visualized in Fig. 6.

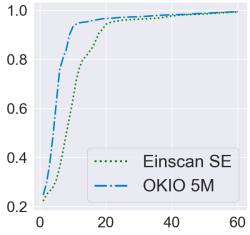


Fig. 4: The relationship between scan quality and scan shots.

Fig. 5: An illustration of the objects used in our real-scanned dataset. The objects are organized by the types of material, where "a" shows the objects made of plastic, "b" for ceramic, "c" for cloth, and "d" for metal.

TABLE 4: Statistics of real-scanned data. # denotes the number.

		Einscan SE	OKIO 5M
precision		100 micrometers	5 micrometers
professional operators		×	✓
#resolution		1300k	5000k
#viewpoints		30-50	13-35
#scanned points		210k-3000k	330k-2000k
#surfaces		20	
#material	metal	4	
	plastic	9	
	ceramic	5	
	cloth	2	

Fig. 6: Examples of real-scanned object pairs.

5 Experimental Set-up for Benchmarking Existing Surface Reconstruction Methods

With the benchmark prepared in Section 4, we aim to empirically compare existing surface reconstruction methods, and identify their advantages and disadvantages in different working conditions (e.g., various scanning imperfections considered in our benchmark). We expect such studies would both provide insights for future research in this area, and guide the use of appropriate methods for practical surface reconstruction from point observations. Such a comprehensive investigation could be timely for the community,

given that a plethora of new methods have been proposed recently, in particular those based on deep learning.

5.1 Data

We conduct the empirical studies using a subset of our collected benchmark, while publicly releasing the whole benchmark to facilitate the community research. More specifically, we randomly sample 22 synthetic surfaces of object instances, whose distribution in the three groups of low-, middle-, and high-complexity is $12:6:4$; as described in Section 4.1, for each instance we conduct six ways of synthetic scanning, which produce either a clean point cloud or point clouds with various imperfections, giving rise to a total of 308 input-output pairs for benchmarking algorithms. We also use all the 10 synthetic scene surfaces and all the 20 real-scanned surfaces for the studies. For some of existing methods using learning-based priors, we prepare an auxiliary set of training data consisting of surfaces from ShapeNet [17] and ABC [20], and also the remaining synthetic data of object instances from our benchmark.

5.2 Pre-processing

While existing surface reconstruction methods can be compared using the data prepared in Section 5.1, for most of them, their performance could be greatly improved via some standard pipeline of point cloud pre-processing. In this work, we compare existing methods both without and with such a pre-processing pipeline. For synthetic data, the pipeline is in the order of outlier removal, de-noising, and point re-sampling; details are given as follows. For real-scanned data, we use the inbuilt pre-processing of different scanners, and use a final step of FPS to re-sample $200,000$ points for each point cloud.

Outlier removal – Performance of surface reconstruction degrades severely when extreme outliers exist in a point cloud; fortunately, these outliers are easy to be removed. We use a statistical method [132] to remove extreme point outliers. For a point cloud $\mathcal{P}$, it regards any $\boldsymbol{p} \in \mathcal{P}$ as an outlier and remove it when $\boldsymbol{p}$ is very far away from its local neighborhood. More precisely, for any $\boldsymbol{p} \in \mathcal{P}$, we first compute the averaged distance $\bar{d}_{\boldsymbol{p}}$ between $\boldsymbol{p}$ and its k nearest neighbors in $\mathcal{P}$; we then compute the mean $m_{\bar{d}}$ and standard deviation $\sigma_{\bar{d}}$ of such distances for all $\{\boldsymbol{p} \in \mathcal{P}\}$; a point $\boldsymbol{p}$ is regarded as an outlier when its corresponding $\bar{d}_{\boldsymbol{p}} > 5 \cdot \sigma_{\bar{d}}$. We set $k = 35$ in this work for outlier removal.

De-noising – The inevitable existence of point-wise noise influences surface reconstruction as well. For an input point cloud $\mathcal{P}$, we choose to suppress such noise using Jets smoothing [133], which smoothes out the point cloud without sacrificing its surface curvatures. It works by first fitting a parametric surface patch to a local neighborhood $\mathcal{N}$ of k points in $\mathcal{P}$, and then projecting $\{\boldsymbol{p} \in \mathcal{N}\}$ onto the fitted surface patch. We set $k = 18$ in this work for point-wise de-noising.

Point re-sampling – Empirical results show that surface reconstruction benefits from more uniform distribution of points, even when reducing the number of points contained in $\mathcal{P}$ [134]. For the synthetic data of object or scene surfaces, we simply use farthest point sampling [99] as the method to re-sample a more uniform distribution of points; we preserve 40% of original points during re-sampling.

5.3 Evaluation Metrics

We quantitatively compare reconstruction results from different methods using the popular metrics of Chamfer Distance (CD) [135], F-score [136], and Normal Consistency Score (NCS) [9]; Appendix B specifies their computations. We also propose a neural metric, termed Neural Feature Similarity (NFS), focusing on perceptual consistency between each reconstruction and the ground-truth; intuitively speaking, NFS compares the similarity of two shapes in the deep feature space, and thus depends more on the high-level semantic information consistent with human perception [137]; details are given in Appendix B as well.

CD and F-score are used for measuring the overall similarity between two shapes; NCS is more useful for measuring the nuance of two similar shapes by measuring their consistency of surface normals; NFS measures semantic difference related to human perception.

5.4 Methods and Implementation Details

In Section 3.5, we have categorized existing methods according to what geometric priors they have respectively used to regularize the reconstruction. It is less feasible to study and empirically compare all the existing methods; instead, we take the strategy of selecting representative ones from each method group of geometric priors, assuming that our studies and conclusions would generalize in the same groups of existing methods. More specifically, we adopt the most representative Greedy Delaunay (GD) [33] and BPA [34] as the methods to be studied for triangulation-based prior; for priors of surface smoothness, we adopt SPSR [36] using the first manner of surface smoothness (cf. Eq. (9)) and RIMLS [53] using both two manners of surface smoothness (cf. Eq. (8) and constraining $\mathcal{H}_{\boldsymbol{f}}$); for modeling priors, we adopt SALD [73] that is able to reconstruct surfaces from un-oriented point clouds, and IGR [71] that can do so from oriented ones; for learning-based priors, we adopt the global, semantic learning methods of OccNet [9] and DeepSDF [8], and also the local learning methods of LIG [10] and Points2Surf [78]; we consider three methods that use hybrid priors, including Delaunay Surface Elements (DSE) [114] that combines triangulation-based prior with learning-based prior, IMLSNet [115] that combines surface smoothness prior with learning-based prior, and ParseNet [40] that combines template-based prior with learning-based prior; we do not consider methods using template-based priors *only*, given that their performance largely depends on whether there would exist a good match between a surface to be reconstructed and the assumed templates (e.g., the assumed geometric primitives or the templates that can be retrieved in an auxiliary dataset), and that even the one with learning-based templates [40] fail to reconstruct surfaces of certain complexities.

Our implementations of the more classical, learning-free methods are based on established libraries; for example, we implement GD using the CGAL library [138], and directly execute BPA [34], SPSR [36], and RIMLS [53] with proper parameter tunings in MeshLab [139]. For those learning-based methods, we use the implementation codes publicly released by the authors when they are available, again with proper tuning of hyper-parameters for individual surfaces to be reconstructed. For those without code releasing, we re-implement their algorithms and tune the respective hyper-parameters as the optimal ones.

In Section 6, we report and discuss the results *that are obtained with the pre-processing mentioned in Section 5.2*. We note that

those without pre-processing are of similar comparative qualities, and we put them in Appendix E.

6 Main Results

We first summarize our key empirical findings, before presenting details of our series of experiments; insights are drawn subsequently.

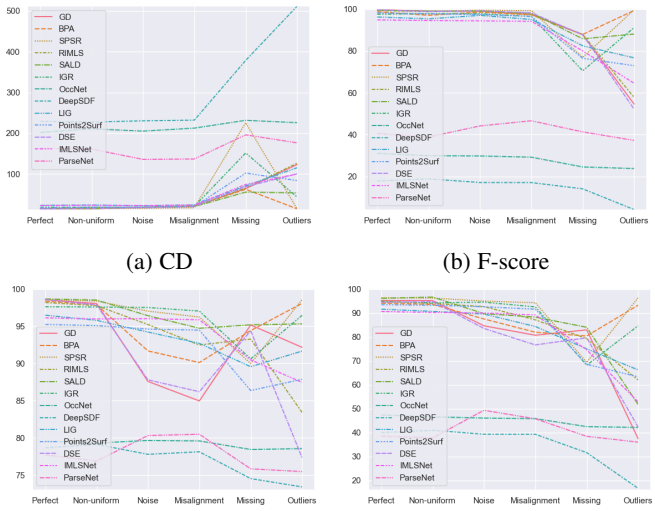
- While many challenges of surface reconstruction from point clouds can be more or less tackled by using different regularization/priors of surface geometry, the challenges of *misalignment*, *missing points*, and *outliers* have been less addressed and remain unsolved.
- Data-driven solutions using deep learning have recently shown great promise for surface modeling and reconstruction, including their potential to deal with various data imperfections, however, our systematic experiments suggest that they struggle in generalizing to reconstruction of complex shapes; it is surprising that some classical methods such as SPSR [36] perform even better in terms of generalization and robustness.
- Use of surface normals is a key to success of surface reconstruction from raw, observed point clouds, even when the surface normals are estimated less accurately; in many cases, the reconstruction result improves as long as the interior and exterior of a surface can be identified in the 3D space.
- There exist inconsistencies between different evaluation metrics, and in many cases, good quantitative results do not translate as visually pleasant ones. For example, quantitative results measured by CD and F-score are not much affected by the challenge of misalignment; however, the reduced scores of NCS and NFS suggest that the recovered surfaces might be less pleasant to human perception.

Quantitative results of comparative methods are given in Tables 5, 6, and 7, which are respectively for synthetic data of object surfaces, synthetic data of scene surfaces, and real-scanned data. Qualitative results are presented in the following sections accompanying our discussions.

6.1 The Remaining Challenges

For ease of analysis, we plot in Fig. 7 the quantitative results in Table 5 under the metrics of CD, F-score, NCS, and NFS. As presented in Section 5.3, the four metrics focus on different measure perspectives. By diagnosing the comparative methods using these measures and investigating their capabilities to cope with different challenges of imperfect scanning, Fig. 7 helps in identifying the remaining challenges.

Fig. 7 shows that, under all the metrics, the challenge of *non-uniform distribution of points* is relatively easy to be tackled by almost all the methods, except those learning semantics or geometric primitives (i.e., DeepSDF [8], OccNet [9], and ParseNet [40]), achieving similar results as those on data of perfect scanning. The discussion on why some learning-based methods fail to generalize is given in Section 6.2. For the challenge of *point-wise noise*, though the overall shape structures (measured by CD and F-Score) can be roughly recovered by most of the methods (again, except some learning-based ones), the reconstructions might be short of surface details, as verified by the reduced scores of NCS and NFS, especially for triangulation-based methods such as GD [33], BPA [34], and the hybrid one of DSE [114] that combines the triangulation-based prior. This is intuitive since methods based on triangulation of points rely heavily on cleanness of input points.



(a) CD (b) F-score

(c) NCS (d) NFS

Fig. 7: Plotting of quantitative results in Table 5.

There exists a similar but severer phenomenon for the challenge of *misalignment*. As indicated by Fig. 7, when measured by CD and F-score, most of the methods give reasonably good results, suggesting that the overall shape structures have been recovered. However, the reduced scores of NCS and NFS suggest that some of the recovered surfaces might be less pleasant to human perception. In fact, as shown by the example in Fig. 8, most methods fail in reconstructing the surface on the misaligned areas, generating thickened or even multiple layers of the local surface. Misalignment is a practical issue in 3D scanning, especially when using hand-held, consumer scanners. Fig. 7 and Fig. 8 show that methods using smoothness and/or modeling priors have the advantage in handling misalignment.

Fig. 7 also shows that the two challenges of *missing points* and *point outliers* are much more difficult to be handled. When an input point cloud has missing points, as shown in Fig. 9, most of the methods ignore reconstruction of the missing surface areas, resulting in incomplete surfaces. The implicit methods (e.g., IGR [71], OccNet [9], and Points2Surf [78]) tend to generate watertight surfaces by filling the holes with concave/convex hulls; however, such envelopes may not represent the surface correctly, possibly making the reconstruction even poorer under all the evaluation metrics (cf. Fig. 7).

As for *point outliers*, although a pre-processing step of outlier removal has already been adopted for all the comparative methods (cf. Section 5.2), performance of different methods still varies drastically. Fig. 10 gives an example; the results depend on whether the respective methods have their inbuilt mechanisms of outlier removal. For example, BPA [34] requires the sizes of its triangular faces to satisfy certain conditions, making it naturally suitable for handling outliers; methods using a global implicit field (e.g., SPSR [48] and IGR [71]) ignore the outliers implicitly, and are thus capable of handling point outliers as well.

Summarizing the above analyses gives us the following empirical findings: (1) the challenge of *missing points* remains unsolved by all the comparative methods; (2) for the challenges of *misalignment* and *point outliers*, most of the methods (except few ones such as SPSR [48]) give unsatisfactory results; (3) there

TABLE 5: Quantitative results on the testing synthetic data of object surfaces. Comparisons are made on data of perfect scanning and those of all the five challenges of imperfect scanning specified in Section 4.1.3; for those challenges with varying levels of severity, we use the data of middle-level severity for the comparison (cf. Appendix C for the overall results). Results of the **best** and **second best** methods are highlighted in each column. Comparative methods are also grouped according to what priors of surface geometry they have used (cf. Section 3 for the grouping and Section 5.4 for how these representative methods are selected).

Prior	Method	CD ($\times 10^{-4}$) ↓						F-score (%) ↑					
		Perfect scanning	Non-uniform distribution	Point-wise noise	Point outliers	Missing points	Mis-alignment	Perfect scanning	Non-uniform distribution	Point-wise noise	Point outliers	Missing points	Mis-alignment
Triangulation-based	GD [33]	**14.24**	**14.87**	18.20	123.19	64.15	21.15	**99.66**	**99.07**	98.91	54.59	87.72	97.41
	BPA [34]	14.89	17.13	18.58	**14.66**	**62.55**	20.88	98.70	97.02	98.51	**99.31**	**87.85**	97.24
Smoothness	SPSR [36]	14.47	15.36	**16.05**	**14.71**	225.66	**17.24**	99.59	99.02	**99.46**	**99.65**	76.91	**99.27**
	RIMLS [53]	15.73	16.74	**17.17**	126.40	65.36	21.12	99.27	98.76	**99.24**	57.78	**87.92**	97.53
Modeling	SALD [73]	15.10	**14.96**	18.77	53.65	**55.63**	20.09	99.45	**99.10**	98.94	88.05	85.78	98.09
	IGR [71]	18.40	18.33	18.57	43.60	151.53	20.72	97.54	97.79	97.75	91.11	70.49	96.46
Learning Semantics	OccNet [9]	201.96	210.80	205.21	225.85	231.65	212.51	31.03	29.90	29.77	23.75	24.52	29.19
	DeepSDF [8]	229.18	227.42	230.40	511.36	378.58	232.16	17.79	18.81	17.08	4.15	14.06	17.05
Local Learning	LIG [10]	23.09	24.03	22.05	115.38	70.98	24.30	96.20	95.32	96.99	76.69	82.43	95.01
	Points2Surf [78]	17.18	18.81	18.48	83.91	102.18	20.36	98.14	97.72	97.39	72.94	76.46	96.49
Hybird	DSE [114]	**14.26**	15.34	17.89	100.37	68.88	**20.06**	**99.64**	98.84	99.17	52.21	87.71	**98.20**
	IMLSNet [115]	22.56	23.17	22.67	99.95	74.35	23.77	94.82	94.51	94.36	64.55	80.01	94.14
	ParseNet [40]	162.94	161.14	135.84	176.38	195.98	136.86	40.52	38.82	44.13	37.21	41.28	46.60

Prior	Method	NCS ($\times 10^{-2}$) ↑						NFS ($\times 10^{-2}$) ↑					
		Perfect scanning	Non-uniform distribution	Point-wise noise	Point outliers	Missing points	Mis-alignment	Perfect scanning	Non-uniform distribution	Point-wise noise	Point outliers	Missing points	Mis-alignment
Triangulation-based	GD [33]	98.57	98.05	87.58	92.17	**95.17**	84.96	95.22	95.19	84.63	37.47	**82.93**	80.69
	BPA [34]	98.37	97.89	91.68	**98.07**	94.42	90.12	94.10	93.60	87.49	**93.24**	80.36	81.96
Smoothness	SPSR [36]	98.58	**98.38**	**97.03**	**98.56**	89.99	**96.24**	**96.38**	**96.22**	**94.98**	**96.31**	69.34	**94.28**
	RIMLS [53]	98.19	97.77	95.23	83.42	93.27	92.48	95.01	94.02	92.67	62.01	79.23	87.12
Modeling	SALD [73]	**98.67**	**98.52**	96.42	95.32	**95.19**	94.73	**96.11**	**96.65**	89.72	51.74	**84.01**	88.26
	IGR [71]	97.62	97.59	**97.52**	96.47	90.61	**97.04**	94.71	94.22	**94.52**	84.63	68.14	**92.54**
Learning Semantics	OccNet [9]	79.55	79.30	79.64	78.55	78.43	79.58	47.33	46.55	46.03	42.11	42.46	45.80
	DeepSDF [8]	78.65	79.11	77.78	73.40	74.52	78.12	39.94	40.91	39.29	16.65	31.59	39.26
Local Learning	LIG [10]	96.49	95.79	94.22	91.66	89.56	92.70	91.58	90.66	89.55	66.21	74.98	84.34
	Points2Surf [78]	95.24	95.09	94.62	87.87	86.36	94.48	93.45	93.23	92.59	63.30	68.53	91.59
Hybird	DSE [114]	**98.60**	97.86	87.79	77.34	94.40	86.20	94.50	94.75	83.53	42.32	79.62	76.63
	IMLSNet [115]	96.13	95.98	96.02	87.45	90.48	95.87	90.61	90.20	89.97	52.82	74.59	89.19
	ParseNet [40]	77.71	76.89	80.31	75.46	75.83	80.48	38.54	37.71	49.30	35.98	38.40	45.73

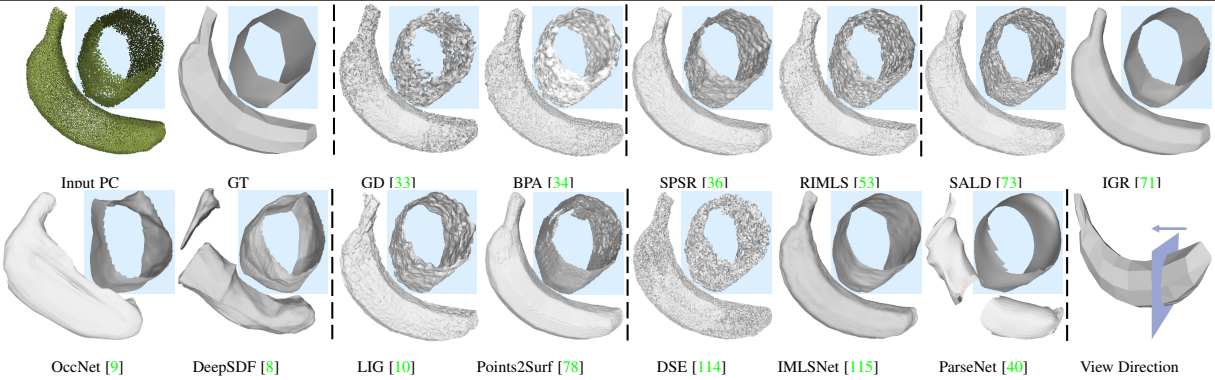


Fig. 8: An example of qualitative results from different methods when dealing with the challenge of *misalignment*.

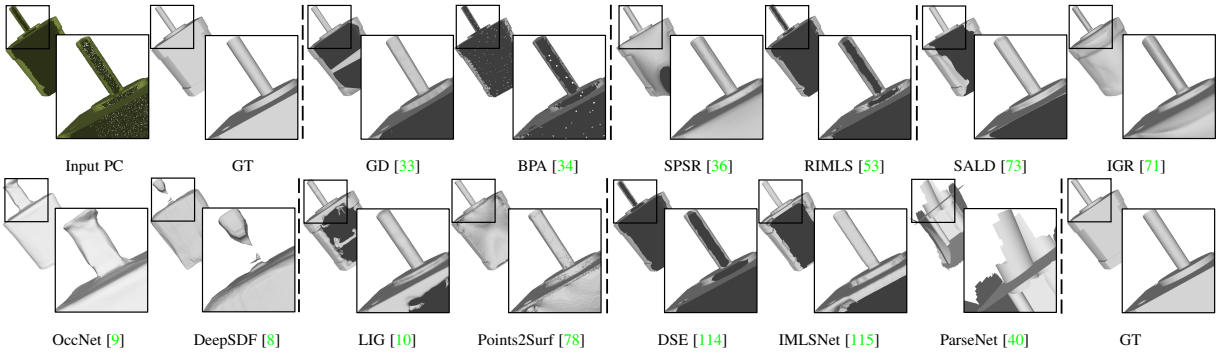


Fig. 9: An example of qualitative results from different methods when dealing with the challenge of *missing points*.

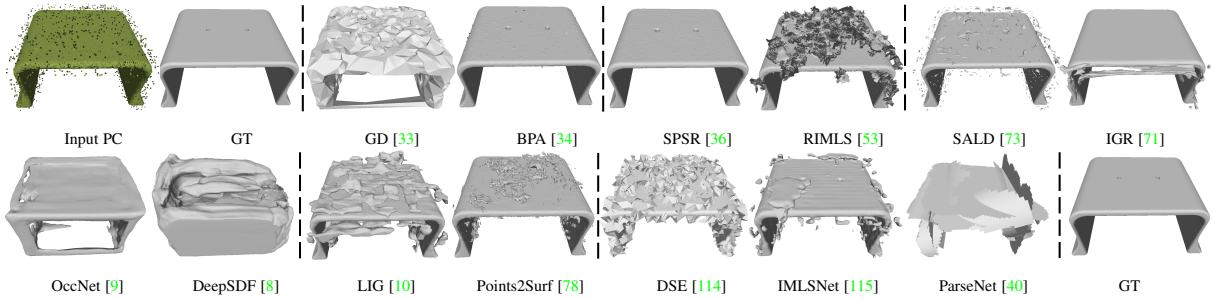


Fig. 10: An example of qualitative results from different methods when dealing with the challenge of *point outliers*.

TABLE 6: Quantitative results on the testing synthetic data of scene surfaces. Results of the **best** and **second best** methods are highlighted in each column. Comparative methods are grouped according to what priors of surface geometry they have used (cf. Section 3 for the grouping and Section 5.4 for how these representative methods are selected). "-" indicates the method cannot produce reasonable results due to their limited generalization.

Prior	Method	CD ($\times 10^{-3}$) ↓	F-score (%) ↑	NCS ($\times 10^{-2}$) ↑	NFS ($\times 10^{-2}$) ↑
Triangulation-based	GD [33]	33.85	75.95	62.08	41.92
	BPA [34]	45.82	53.46	58.25	44.19
Smoothness	SPSR [36]	**30.47**	**83.22**	83.74	**63.03**
	RIMLS [53]	41.45	74.56	69.93	38.13
Modeling	SALD [73]	32.43	79.03	**91.58**	52.79
	IGR [71]	**31.41**	**81.63**	**91.26**	**67.58**
Learning Semantics	OccNet [9]	93.12	37.75	85.98	50.34
	DeepSDF [8]	-	-	-	-
Local Learning	LIG [10]	41.40	78.03	88.12	59.97
	Points2Surf [78]	36.24	76.14	83.60	61.82
Hybird	DSE [114]	32.97	77.53	57.99	41.56
	IMLSNet [115]	35.52	78.05	87.17	61.98
	ParseNet [40]	-	-	-	-

TABLE 7: Quantitative results on the real-scanned data. Results of the **best** and **second best** methods are highlighted in each column. Comparative methods are grouped according to what priors of surface geometry they have used (cf. Section 3 for the grouping and Section 5.4 for how these representative methods are selected).

Prior	Method	CD ($\times 10^{-2}$) ↓	F-score (%) ↑	NCS ($\times 10^{-2}$) ↑	NFS ($\times 10^{-2}$) ↑
Triangulation-based	GD [33]	31.72	87.51	88.86	82.20
	BPA [34]	40.37	80.95	87.56	68.69
Smoothness	SPSR [36]	**31.05**	**87.74**	**94.94**	**89.38**
	RIMLS [53]	32.80	87.05	91.97	85.19
Modeling	SALD [73]	**31.13**	**87.72**	94.68	86.86
	IGR [71]	32.70	87.18	**95.99**	**89.10**
Learning Semantics	OccNet [9]	232.71	17.11	80.96	39.70
	DeepSDF [8]	263.92	19.83	77.95	40.95
Local Learning	LIG [10]	48.75	83.76	92.57	81.48
	Points2Surf [78]	48.93	80.89	89.52	81.83
Hybird	DSE [114]	32.16	86.88	87.20	76.81
	IMLSNet [115]	38.46	82.44	93.31	85.30
	ParseNet [40]	149.96	38.92	81.51	45.67

exist inconsistencies between different evaluation metrics, and in many cases, good quantitative results do not translate as visually pleasant ones; (4) methods that learn semantics or pre-defined shape patterns may fail to generalize even on clean data of perfect scanning, when the testing point clouds do not fall in the learned data domains; we will discuss more on this issue shortly.

6.2 Optimization-based, Learning-free Methods *Versus* Learning-based, Data-driven Ones

In this section, we investigate the behaviors of comparative methods by organizing them into two groups of *optimization-based, learning-free methods* and *learning-based, data-driven ones*. The former group includes SPSR [36], RIMLS [53], SALD [73], IGR [71], and we also include GD [33] and BPA [34] into the group for a complete coverage of the studied methods; the latter group includes OccNet [9], DeepSDF [8], LIG [10], Points2Surf [78], DSE [114], IMLSNet [115], and ParseNet [40]. By doing so, we aim to investigate how the two groups perform in terms of generalizing to complex shapes, where we pay special attention to methods of global, semantic learning (i.e., OccNet [9] and DeepSDF [8]) whose advantages may only be manifested when the categories of object surfaces exist in the auxiliary training set. Note that our testing data of synthetic object surfaces include 22 randomly selected objects, as described in Section 5.1, which contain both those belonging to popular semantic categories (e.g., *chair*) and those without clearly defined semantics; for the former case, our auxiliary training set contains rich shape instances of same categories from ShapeNet [17]. In this section, we also study robustness of the two groups of methods against imperfect scanning at varying levels of severity.

Quantitative results in Table 8 show that on reconstruction of synthetic object surfaces, optimization-based, learning-free methods generalize better under different evaluation metrics, since our testing shapes contain both semantic ones and non-semantic, complex ones; we have consistent observations from examples in Fig. 11 — while learning-based, data-driven methods are good at reconstructing an *chair* surface, they fail in generalizing to non-semantic shapes. Table 8 also show that learning-based methods are good in terms of robustness against higher levels of data imperfections. We observe similar phenomena on reconstruction of real-scanned data (cf. Appendix F for more details), by re-organizing Table 7 according to the two method groups.

To further investigate the behaviors of learning from auxiliary data, we conduct experiments of synthetic scene surface reconstruction. Results in Table 6 (after re-organization of method groups) and Fig. 12 show that among the learning-based methods, LIG [10], Points2Surf [78] and IMLSNet [115] are capable of handling reconstruction of scene-level surfaces via local modeling and aggregation, while semantic learning methods of OccNet and DeepSDF fail, as expected.

We summarize the above analyses as follows: (1) learning-based, data-driven methods are able to reconstruct object surfaces when the training set contains object instances of the same semantic categories, and they show a certain degree of robustness against data imperfections; however, these methods fail to generalize when the condition is not satisfied; (2) for reconstruction of scene-level surfaces, local learning methods succeed by local modeling and aggregation, and in contrast, global, semantic learning methods fail to do so; (3) some optimization-based, learning-free methods (e.g., SPSR [36]) perform surprisingly well in robustness and generalization for both object-level and scene-level surfaces.

6.3 The Importance of Orientations of Surface Normals

Some of our studied methods compute surface normals from observed point clouds, and use the computed normals for surface reconstruction, including BPA [34], SPSR [36], RIMLS [53], IGR [71], OccNet [9], DeepSDF [8], LIG [10], and ParseNet [40]; a few other methods (e.g., Points2Surf [78] and IMLSNet [115]) compute surface normals on training data, and then train models to estimate surface normals when reconstructing testing surfaces. To investigate how these methods benefit from surface normal computation/estimation, we conduct experiments on our testing data with scanning imperfections, since robustness of these methods would be tested when less accurate surface normals are computed from imperfectly scanned data. Assume that the relative pose of a camera w.r.t. an observed point cloud is given; for any observed point, we compute its *oriented* surface normal by performing PCA on its local neighborhood of points (cf. Section 4.1.3 for the details); as such, the computed surface normals may not be precise but their *inward or outward orientations* must be correct.

Table 9 gives the quantitative results for synthetic data of object surfaces; under different evaluation metrics, the additional computation or estimation of surface normals does help in improving surface reconstruction. When the relative pose of camera w.r.t. observed surface points is not available, the computed surface normals would be *wrongly oriented* at some local surface neighborhoods, since principal directions of PCA on local neighborhoods of noisy points are less reliable. To investigate the

TABLE 8: Comparison between optimization-based, learning-free methods and learning-based, data-driven methods on the testing synthetic data of object surfaces. We report quantitative results for the imperfect scanning of *point-wise noise* at three levels of severity; results are in the format of "· / · / ·", where the most left one is the absolute value under each evaluation metric, and the right two ones are those relative to the most left one. The **best** and **second best** methods are highlighted in each column.

Algorithms	Priors	CD ($\times 10^{-4}$) ↓			F-score (%) ↑			NCS ($\times 10^{-2}$) ↑			NFS ($\times 10^{-2}$) ↑		
		low-	*/middle-*	*/high-level*	*low-*	*/middle-*	*/high-level*	*low-*	*/middle-*	*/high-level*	*low-*	*/middle-*	*/high-level*
GD [33]	learning-free	16.52 /	1.68 /	13.21	99.09 /	-0.18 /	-7.44	94.00 /	-6.42 /	-25.68	91.57 /	-6.94 /	-35.27
BPA [34]		16.59 /	1.99 /	12.71	98.63 /	-0.12 /	-9.04	95.36 /	-3.68 /	-17.39	90.71 /	-3.22 /	-25.56
SPSR [36]		**15.50** /	0.55 /	2.66	**99.51** /	-0.05 /	**-0.35**	**97.84** /	-0.81 /	-3.95	**95.60** /	-0.62 /	-3.61
RIMLS [53]		16.13 /	1.04 /	9.75	99.36 /	-0.12 /	-4.67	97.36 /	-2.13 /	-11.23	94.19 /	-1.52 /	-14.43
SALD [73]		**15.33** /	3.44 /	12.26	**99.54** /	-0.60 /	-7.06	**98.07** /	-1.65 /	-9.16	**95.66** /	-5.94 /	-28.48
IGR [71]		18.21 /	0.36 /	0.99	97.87 /	-0.12 /	**-0.35**	97.64 /	-0.12 /	-0.54	94.37 /	0.15 /	-0.70
OccNet [9]	learning-based	209.04 /	-3.83 /	0.97	29.85 /	-0.08 /	-1.93	79.43 /	0.21 /	-0.38	46.17 /	-0.14 /	-1.13
DeepSDF [8]		241.28 /	**-10.88** /	**-1.65**	16.70 /	**0.38** /	-0.71	78.01 /	-0.23 /	**0.51**	38.55 /	0.74 /	**0.15**
LIG [10]		23.96 /	-1.91 /	2.31	94.50 /	**2.49** /	-0.70	93.96 /	**0.26** /	-5.70	86.71 /	**2.84** /	-5.68
Points2Surf [78]		17.74 /	0.74 /	4.89	98.02 /	-0.63 /	-3.81	94.97 /	-0.35 /	-1.53	92.83 /	-0.24 /	-2.62
DSE [114]		16.07 /	1.82 /	11.54	99.40 /	-0.23 /	-6.65	94.43 /	-6.64 /	-23.04	90.05 /	-6.52 /	-32.97
IMLSNet [115]		22.64 /	0.03 /	1.09	94.41 /	-0.05 /	**-0.01**	96.08 /	-0.06 /	-0.46	90.13 /	-0.16 /	-0.47
ParseNet [40]		154.11 /	**-18.27** /	**-14.14**	44.80 /	-0.67 /	-2.14	78.55 /	**1.76** /	**2.60**	46.21 /	**3.09** /	**2.79**

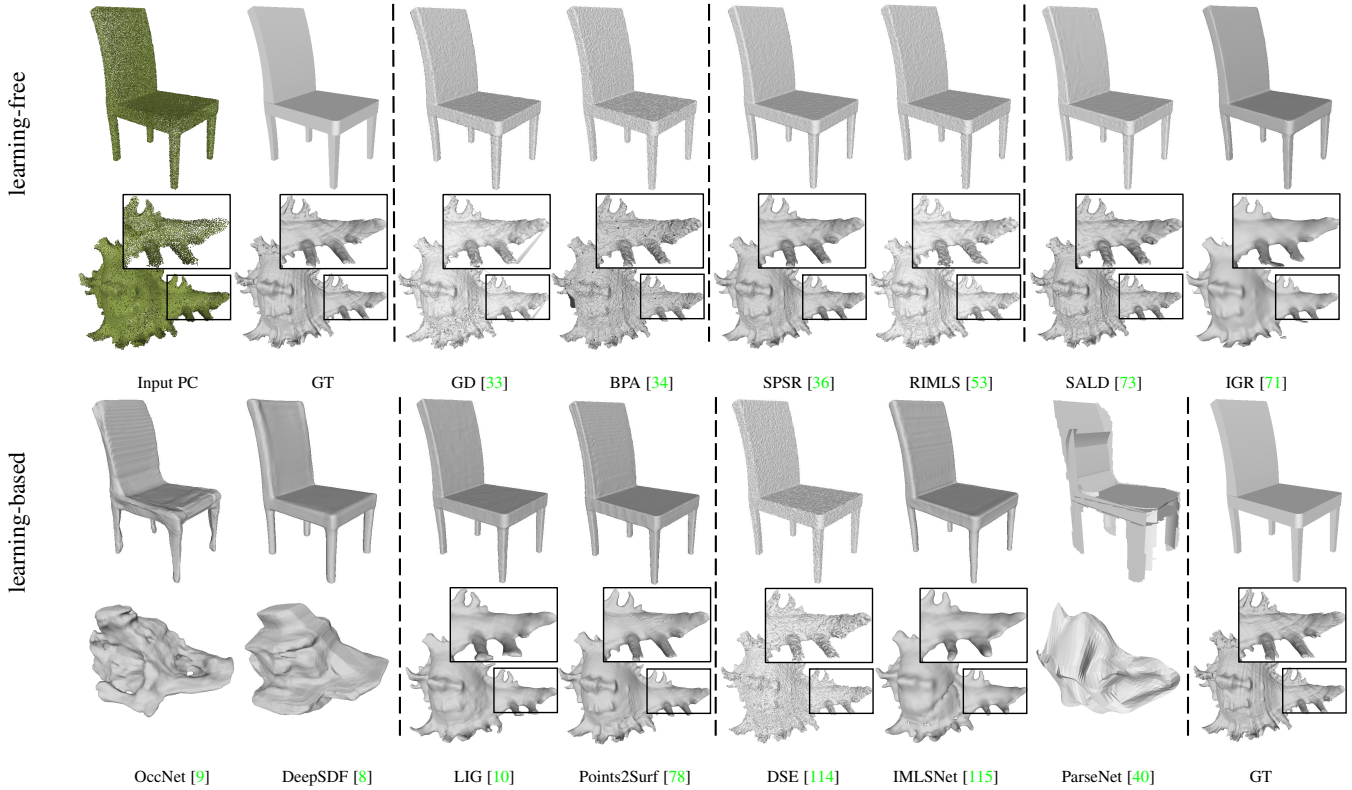


Fig. 11: Qualitative results respectively from optimization-based, learning-free methods and learning-based, data-driven methods on the testing synthetic data of object surfaces. The input point clouds have imperfect scanning of *point-wise noise*. The example of *chair* is of popular semantic categories, for which our auxiliary training set contains rich instances of the same category from ShapeNet [17]; the other one is non-semantic.

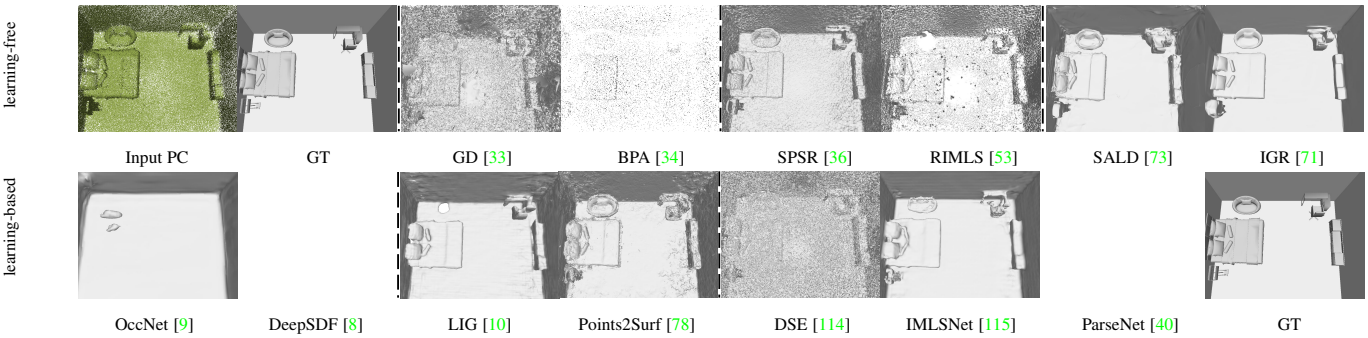


Fig. 12: Qualitative results respectively from optimization-based, learning-free methods and learning-based, data-driven methods on the testing synthetic data of scene surfaces. The scene is a bedroom instance from the 3D-FRONT [24].

TABLE 9: Comparison on testing synthetic data of object surfaces among methods without using surface normals ($\times$), methods using surfaces normals ($\checkmark$), and methods using surface normals only during learning ($*$). We report quantitative results for the imperfect scanning of *point-wise noise* at the middle level of severity; results are in the format of "· / ·", where the left one is obtained assuming the availability of ground-truth camera poses, and the right one is obtained without knowing the camera poses. The **best** and **second best** methods are highlighted in each column.

Algorithms	Normals	CD ($\times 10^{-4}$) ↓	F-score (%) ↑	NCS ($\times 10^{-2}$) ↑	NFS ($\times 10^{-2}$) ↑
GD [33]		18.20 /	98.91 /	87.58 /	84.63 /
SALD [73]	$\times$	18.77 /	98.94 /	96.42 /	89.72 /
DSE [114]		17.89 /	99.17 /	87.79 /	83.53 /
BPA [34]		18.58 / 18.61	98.51 / 98.56	91.68 / 91.79	87.49 / 85.08
SPSR [36]		**16.05** / 60.42	**99.46** / 91.45	**97.03** / 94.68	**94.98** / 84.34
RIMLS [53]		**17.17** / 18.58	**99.24** / 97.47	95.23 / 94.06	92.67 / 86.63
IGR [71]	✓	18.57 / 262.40	97.75 / 85.98	**97.52** / 95.28	**94.52** / 79.61
OccNet [9]		205.21 / 214.12	29.77 / 28.83	79.64 / 79.01	46.03 / 45.12
DeepSDF [8]		230.40 / 569.41	17.08 / 15.64	77.78 / 77.09	39.29 / 33.78
LIG [10]		22.05 / 34.78	96.99 / 89.60	94.22 / 91.36	89.55 / 80.02
ParseNet [40]		135.84 / 197.28	44.13 / 35.67	80.31 / 76.19	49.30 / 41.65
Points2Surf [78]	*	18.48 /	97.39 /	94.62 /	92.59 /
IMLSNet [115]		22.67 /	94.36 /	96.02 /	89.97 /

importance of inward or outward orientations of surface normals, we also report experiments in Table 9 where each result on the right of the "/" symbol is obtained without knowing the relative camera poses; compared with results on the left side of "/" that are obtained assuming the ground-truth camera poses, results on the right side drops drastically. We also conduct experiments on our synthetic data of scene surfaces and real-scanned data, and observe similar phenomena; these results are presented in Appendix D.

We have following empirical findings based on the above analyses: (1) computation or estimation of oriented surface normals help in surface reconstruction from point clouds; (2) compared with precisions of surface normals, it is more important to have the correct inward or outward orientations of surface normals.

7 Conclusion

In this paper, we have reviewed both the classical and the more recent deep learning-based methods for surface reconstruction from point clouds; we have organized our reviews by categorizing these methods according to what priors of surface geometry they had used to regularize their solutions. To better understand the respective strengths and limitations of existing methods, we contribute a large-scale benchmarking dataset consisting of both synthetic and real-scanned data, which provides various sensing imperfections that are commonly encountered in practical 3D scanning. We conduct thorough empirical studies on the constructed benchmark, evaluating the robustness and generalization of different methods. Our studies help identify the remaining challenges faced by existing methods, and we expect that our studies would be useful for guiding the directions in future research.

References

[1] R. Bolle and B. Vemuri, "On three-dimensional surface reconstruction methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 1, pp. 1–13, 1991. 1, 2

[2] S. P. Lim and H. Haron, "Surface reconstruction techniques: a review," *Artificial Intelligence Review*, vol. 42, no. 1, pp. 59–78, 2014. 1, 2

[3] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva, "A survey of surface reconstruction from point clouds," in *Computer Graphics Forum*, vol. 36, no. 1. Wiley Online Library, 2017, pp. 301–329. 1, 2, 5

[4] C. C. You, S. P. Lim, S. C. Lim, J. San Tan, C. K. Lee, and Y. M. J. Khaw, "A survey on surface reconstruction techniques for structured and unstructured data," in *2020 IEEE Conference on Open Systems (ICOS)*. IEEE, 2020, pp. 37–42. 1, 2

[5] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 216–224. 1, 6

[6] Y. Luo, Z. Mi, and W. Tao, "Deepdt: Learning geometry from delaunay triangulation for surface reconstruction," in *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021, pp. 2277–2285. 1, 7

[7] N. Sharp and M. Ovsjanikov, "Pointtrinet: Learned triangulation of 3d point sets," in *European Conference on Computer Vision*. Springer, 2020, pp. 762–778. 1, 7

[8] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174. 1, 4, 6, 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27

[9] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4460–4470. 1, 4, 6, 12, 13, 14, 15, 16, 17, 21, 23, 24, 25, 26, 27

[10] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser, "Local implicit grid representations for 3d scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6001–6010. 1, 7, 12, 14, 15, 16, 17, 23, 24, 25, 26, 27

[11] G. Slabaugh, R. Schafer, T. Malzbender, and B. Culbertson, "A survey of methods for volumetric scene reconstruction from photographs," in *Volume Graphics 2001*. Springer, 2001, pp. 81–100. 2

[12] F. Remondino and S. El-Hakim, "Image-based 3d modelling: a review," *The photogrammetric record*, vol. 21, no. 115, pp. 269–291, 2006. 2

[13] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *2006 IEEE computer society conference on computer vision and pattern recognition*, vol. 1. IEEE, 2006, pp. 519–528. 2

[14] X. Han, H. Laga, and M. Bennamoun, "Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era," *IEEE transactions on pattern analysis and machine intelligence*, 2019. 2

[15] G. Fahim, K. Amin, and S. Zarif, "Single-view 3d reconstruction: A survey of deep learning methods," *Computers & Graphics*, vol. 94, pp. 164–190, 2021. 2

[16] H. Zhu, Y. Nie, T. Yue, and X. Cao, "The role of prior in image based 3d modeling: a survey," *Frontiers of Computer Science*, vol. 11, no. 2, pp. 175–191, 2017. 2

[17] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015. 2, 12, 15, 16

[18] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920. 2

[19] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze, "3dnet: Large-scale object class recognition from cad models," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 5384–5391. 2, 8

[20] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, "Abc: A big cad model dataset for geometric deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9601–9611. 2, 8, 12

[21] Q. Zhou and A. Jacobson, "Thingi10k: A dataset of 10, 000 3d-printing models," *CoRR*, 2016. 2, 8

[22] V. Albertina, V. Kunsthistorisches Museum, V. Theater Museum, P. Musée Guimet, P. Musée des Monuments français, Cité de l'architecture et du patrimoine, D. des sculptures de la Ville de Paris, P. Musée Carnavalet, L. The Collection, L. Usher Gallery, M. A. N. di Firenze, and B. KODE Artmuseums, "Three d scans," https://threedscans.com/. 2, 8

[23] A. Handa, V. Pătrăucean, S. Stent, and R. Cipolla, "Scenenet: An annotated model generator for indoor scene understanding," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 5737–5743. 2, 10

[24] H. Fu, B. Cai, L. Gao, L.-X. Zhang, J. Wang, C. Li, Q. Zeng, C. Sun, R. Jia, B. Zhao *et al.*, "3d-front: 3d furnished rooms with layouts and semantics," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 933–10 942. 2, 10, 16

[25] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *2014 IEEE international*

conference on robotics and automation (ICRA). IEEE, 2014, pp. 509–516. 3

[26] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-cmu-berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017. 3

[27] A. Kasper, Z. Xue, and R. Dillmann, "The kit object models database: An object model database for object recognition, localization and manipulation in service robotics," *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 927–934, 2012. 3

[28] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019. 3, 10

[29] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017. 3

[30] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva, "A benchmark for surface reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 2, pp. 1–17, 2013. 3

[31] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon mesh processing*. AK Peters / CRC Press, Sep. 2010. 3, 4, 8

[32] H. Edelsbrunner and N. R. Shah, "Triangulating topological spaces," in *Proceedings of the tenth annual symposium on Computational geometry*, 1994, pp. 285–292. 4

[33] D. Cohen-Steiner and F. Da, "A greedy delaunay-based surface reconstruction algorithm," *The visual computer*, vol. 20, no. 1, pp. 4–16, 2004. 4, 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27

[34] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE transactions on visualization and computer graphics*, vol. 5, no. 4, pp. 349–359, 1999. 4, 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27

[35] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006. 4

[36] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, pp. 1–13, 2013. 5, 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27

[37] S. Peng, C. M. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger, "Shape as points: A differentiable poisson solver," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 5

[38] M. Eck and H. Hoppe, "Automatic reconstruction of b-spline surfaces of arbitrary topological type," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 325–334. 5

[39] Y. He and H. Qin, "Surface reconstruction with triangular b-splines," in *Geometric Modeling and Processing, 2004. Proceedings*. IEEE, 2004, pp. 279–287. 5

[40] G. Sharma, D. Liu, S. Maji, E. Kalogerakis, S. Chaudhuri, and R. Měch, "Parsenet: A parametric surface fitting network for 3d point clouds," in *European Conference on Computer Vision*. Springer, 2020, pp. 261–276. 5, 7, 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27

[41] N. Leal, E. Leal, and J. W. Branch, "Simple method for constructing nurbs surfaces from unorganized points," in *Proceedings of the 19th international meshing roundtable*. Springer, 2010, pp. 161–175. 5

[42] A. Hashemian and S. F. Hosseini, "An integrated fitting and fairing approach for object reconstruction using smooth nurbs curves and surfaces," *Computers & Mathematics with Applications*, vol. 76, no. 7, pp. 1555–1575, 2018. 5

[43] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 67–76. 5

[44] M. Kazhdan, "Reconstruction of solid models from oriented point sets," in *Proceedings of the Third Eurographics Symposium on Geometry Processing*, ser. SGP '05. Goslar, DEU: Eurographics Association, 2005, p. 73–es. 5

[45] J. Manson, G. Petrova, and S. Schaefer, "Streaming surface reconstruction using wavelets," in *Computer Graphics Forum*, vol. 27, no. 5. Wiley Online Library, 2008, pp. 1411–1420. 5

[46] D. Levin, "Mesh-independent surface interpolation," in *Geometric Modeling for Scientific Visualization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 37–49. 5

[47] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Point set surfaces," in *Proceedings of the Conference on Visualization '01*, ser. VIS '01. USA: IEEE Computer Society, 2001, p. 21–28. 5

[48] A. Adamson and M. Alexa, "Approximating and intersecting surfaces from points," in *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 2003, pp. 230–239. 5, 13

[49] G. Guennebaud and M. Gross, "Algebraic point set surfaces," *ACM Trans. Graph.*, vol. 26, no. 3, p. 23–es, Jul. 2007. 5

[50] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on visualization and computer graphics*, vol. 9, no. 1, pp. 3–15, 2003. 5, 8, 21

[51] S. Fleishman, D. Cohen-Or, and C. T. Silva, "Robust moving least-squares fitting with sharp features," *ACM transactions on graphics (TOG)*, vol. 24, no. 3, pp. 544–552, 2005. 5

[52] R. Kolluri, "Provably good moving least squares," *ACM Transactions on Algorithms (TALG)*, vol. 4, no. 2, pp. 1–25, 2008. 5

[53] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 493–501. 5, 7, 12, 14, 15, 16, 17, 23, 24, 25, 26, 27

[54] Z.-Q. Cheng, Y.-Z. Wang, B. Li, K. Xu, G. Dang, and S.-Y. Jin, "A survey of methods for moving least squares surfaces." in *Volume graphics*, 2008, pp. 9–23. 5

[55] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," in *Computer graphics forum*, vol. 26, no. 2. Wiley Online Library, 2007, pp. 214–226. 5

[56] L. Nan and P. Wonka, "Polyfit: Polygonal surface reconstruction from point clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2353–2361. 5

[57] Y. Li, A. Dai, L. Guibas, and M. Nießner, "Database-assisted object retrieval for real-time 3d reconstruction," in *Computer Graphics Forum*, vol. 34, no. 2. Wiley Online Library, 2015. 5

[58] Y. M. Kim, N. J. Mitra, Q. Huang, and L. Guibas, "Guided real-time scanning of indoor objects," in *Computer Graphics Forum*, vol. 32, no. 7. Wiley Online Library, 2013, pp. 177–186. 5

[59] L. Nan, K. Xie, and A. Sharf, "A search-classify approach for cluttered indoor scene understanding," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, pp. 1–10, 2012. 5

[60] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas, "Acquiring 3d indoor environments with variability and repetition," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, pp. 1–11, 2012. 5

[61] M. Pauly, N. J. Mitra, J. Giesen, M. H. Gross, and L. J. Guibas, "Example-based 3d scan completion," in *Symposium on Geometry Processing*, no. CONF, 2005, pp. 23–32. 5

[62] C.-H. Shen, H. Fu, K. Chen, and S.-M. Hu, "Structure recovery by part assembly," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, pp. 1–11, 2012. 5

[63] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9446–9454. 5

[64] F. Williams, T. Schneider, C. Silva, D. Zorin, J. Bruna, and D. Panozzo, "Deep geometric prior for surface reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 130–10 139. 5

[65] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or, "Point2mesh: A self-prior for deformable meshes," *ACM Trans. Graph.*, vol. 39, no. 4, Jul. 2020. 5

[66] W. Zhao, J. Lei, Y. Wen, J. Zhang, and K. Jia, "Sign-agnostic implicit learning of surface self-similarities for shape modeling and reconstruction from raw point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 256–10 265. 5

[67] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "Meshcnn: a network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019. 5

[68] M. Gadelha, R. Wang, and S. Maji, "Deep manifold prior," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 1107–1116. 5, 6

[69] M. Atzmon, N. Haim, L. Yariv, O. Israelov, H. Maron, and Y. Lipman, "Controlling neural level sets," in *Advances in Neural Information Processing Systems*, 2019, pp. 2034–2043. 6

[70] J. Lei and K. Jia, "Analytic marching: An analytic meshing solution from deep implicit surface networks," in *International Conference on Machine Learning 2020 ICML-20*, 7 2020. 6

[71] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," in *Proceedings of Machine Learning and Systems 2020*, 2020, pp. 3569–3579. 6, 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27

[72] M. Atzmon and Y. Lipman, "Sal: Sign agnostic learning of shapes from raw data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2565–2574. 6

[73] ——, "Sald: Sign agnostic learning with derivatives," in *International Conference on Learning Representations*, 2021. 6, 12, 14, 15, 16, 17, 23, 24, 25, 26, 27

[74] A. Basher, M. Sarmad, and J. Boutellier, "Lightsal: Lightweight sign agnostic learning for implicit surface representation," *arXiv preprint arXiv:2103.14273*, 2021. 6

[75] T. Davies, D. Nowrouzezahrai, and A. Jacobson, "On the effectiveness of weight-encoded neural implicit 3d shapes," *arXiv preprint arXiv:2009.09808*, 2021. 6

[76] F. Williams, M. Trager, J. Bruna, and D. Zorin, "Neural splines: Fitting 3d surfaces with infinitely-wide neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9949–9958. 6

[77] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *European Conference on Computer Vision (ECCV)*. Cham: Springer International Publishing, Aug. 2020. 6, 7

[78] P. Erler, P. Guerrero, S. Ohrhallinger, N. J. Mitra, and M. Wimmer, "Points2surf learning implicit surfaces from point clouds," in *European Conference on Computer Vision*. Springer, 2020, pp. 108–124. 6, 7, 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27

[79] M. Yang, Y. Wen, W. Chen, Y. Chen, and K. Jia, "Deep optimized priors for 3d shape modeling and reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3269–3278. 6

[80] J. Tang, J. Lei, D. Xu, F. Ma, K. Jia, and L. Zhang, "Sa-convonet: Sign-agnostic optimization of convolutional occupancy networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6504–6513. 6

[81] Y. Liao, S. Donne, and A. Geiger, "Deep marching cubes: Learning explicit surface representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2916–2925. 6

[82] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5939–5948. 6

[83] Y. Duan, H. Zhu, H. Wang, L. Yi, R. Nevatia, and L. J. Guibas, "Curriculum deepsdf," in *European Conference on Computer Vision*. Springer, 2020, pp. 51–67. 6

[84] S. Yao, F. Yang, Y. Cheng, and M. G. Mozerov, "3d shapes local geometry codes learning with sdf," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2110–2117. 6

[85] B. Guillard, F. Stella, and P. Fua, "Meshudf: Fast and differentiable meshing of unsigned distance field networks," *arXiv preprint arXiv:2111.14549*, 2021. 6

[86] S. Giebenhain and B. Goldluecke, "Air-nets: An attention-based framework for locally conditioned implicit representations," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021. 6

[87] X. Yan, L. Lin, N. J. Mitra, D. Lischinski, D. Cohen-Or, and H. Huang, "Shapeformer: Transformer-based shape completion via sparse representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 6

[88] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017. 6

[89] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 259–16 268. 6

[90] J. Tang, X. Han, M. Tan, X. Tong, and K. Jia, "Skeletonnet: A topology-preserving solution for learning mesh reconstruction of object surfaces from rgb images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. 6

[91] J. Pan, X. Han, W. Chen, J. Tang, and K. Jia, "Deep mesh reconstruction from single rgb images via topology modification networks," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9963–9972. 6

[92] J. Chibane, T. Alldieck, and G. Pons-Moll, "Implicit functions in feature space for 3d shape reconstruction and completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6970–6981. 7

[93] J. Chibane, A. Mir, and G. Pons-Moll, "Neural unsigned distance fields for implicit function learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, December 2020. 7

[94] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe, "Deep local shapes: Learning local sdf priors for detailed 3d reconstruction," in *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020, pp. 608–625. 7

[95] Z. Mi, Y. Luo, and W. Tao, "Ssrnet: Scalable 3d surface reconstruction network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 970–979. 7

[96] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, "Neural geometric level of detail: Real-time rendering with implicit 3d shapes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 358–11 367. 7

[97] A. Badki, O. Gallo, J. Kautz, and P. Sen, "Meshlet priors for 3d mesh reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2849–2858. 7

[98] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Stoll, and C. Theobalt, "Patchnets: Patch-based generalizable deep implicit 3d shape representations," in *European Conference on Computer Vision*. Springer, 2020, pp. 293–309. 7

[99] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660. 7, 9, 12

[100] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent convolutions for dense prediction in 3d," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3887–3896. 7

[101] B. Ummenhofer and V. Koltun, "Adaptive surface reconstruction with multiscale convolutional kernels," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 5651–5660. 7

[102] A. Boulch and R. Marlet, "Poco: Point convolution for surface reconstruction," *arXiv preprint arXiv:2201.01831*, 2022. 7

[103] H. Jiang, J. Cai, J. Zheng, and J. Xiao, "Neighborhood-based neural implicit reconstruction from point clouds," in *2021 International Conference on 3D Vision (3DV)*, 2021, pp. 1259–1268. 7

[104] D. Attali, J.-D. Boissonnat, and A. Lieutier, "Complexity of the delaunay triangulation of points on surfaces the smooth case," in *Proceedings of the nineteenth annual symposium on Computational geometry*, 2003, pp. 201–210. 7

[105] J.-D. Boissonnat and F. Cazals, "Smooth surface reconstruction via natural neighbour interpolation of distance functions," *Computational Geometry*, vol. 22, no. 1-3, pp. 185–203, 2002. 7

[106] F. Lafarge and P. Alliez, "Surface reconstruction through point set structuring," in *Computer Graphics Forum*, vol. 32, no. 2pt2. Wiley Online Library, 2013, pp. 225–234. 7

[107] R. R. Paulsen, J. A. Bærentzen, and R. Larsen, "Markov random field surface reconstruction," *IEEE transactions on visualization and computer graphics*, vol. 16, no. 4, pp. 636–646, 2009. 7

[108] J. Giesen, S. Spalinger, and B. Schölkopf, "Kernel methods for implicit surface modeling," in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2005. 7

[109] F. Williams, Z. Gojcic, S. Khamis, D. Zorin, J. Bruna, S. Fidler, and O. Litany, "Neural fields as learnable kernels for 3d reconstruction," 2021. 7

[110] M. Baorui, H. Zhizhong, L. Yu-Shen, and Z. Matthias, "Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces," in *International Conference on Machine Learning (ICML)*, 2021. 7

[111] Y. Lipman, "Phase transitions, distance functions, and implicit neural representations," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 6702–6712. [Online]. Available: http://proceedings.mlr.press/v139/lipman21a.html 7

[112] Z. Wang, P. Wang, Q. Dong, J. Gao, S. Chen, S. Xin, and C. Tu, "Neural-imls: Learning implicit moving least-squares for surface reconstruction from unoriented point clouds," *arXiv preprint arXiv:2109.04398*, 2021. 7

[113] P.-S. Wang, Y. Liu, Y.-Q. Yang, and X. Tong, "Spline positional encoding for learning 3d implicit signed distance fields," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence,*

IJCAI-21, Z.-H. Zhou, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2021, pp. 1091–1097. 7

[114] M.-J. Rakotosaona, P. Guerrero, N. Aigerman, N. J. Mitra, and M. Ovsjanikov, "Learning delaunay surface elements for mesh reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 22–31. 7, 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27

[115] S.-L. Liu, H.-X. Guo, H. Pan, P.-S. Wang, X. Tong, and Y. Liu, "Deep implicit moving least-squares functions for 3d reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1788–1797. 7, 12, 14, 15, 16, 17, 23, 24, 25, 26, 27

[116] L. Li, M. Sung, A. Dubrovina, L. Yi, and L. J. Guibas, "Supervised fitting of geometric primitives to 3d point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2652–2660. 7

[117] M. Liu, X. Zhang, and H. Su, "Meshing point clouds with predicted intrinsic-extrinsic ratio guidance," in *European Conference on Computer Vision*. Springer, 2020, pp. 68–84. 7

[118] J. Gao, W. Chen, T. Xiang, A. Jacobson, M. McGuire, and S. Fidler, "Learning deformable tetrahedral meshes for 3d reconstruction," *Advances In Neural Information Processing Systems*, vol. 33, pp. 9936–9947, 2020. 7

[119] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 52–67. 7

[120] H. Kato, Y. Ushiku, and T. Harada, "Neural 3d mesh renderer," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3907–3916. 7

[121] D. Xiao, S. Lin, Z. Shi, and B. Wang, "Learning modified indicator functions for surface reconstruction," *Computers & Graphics*, vol. 102, pp. 309–319, 2022. 7

[122] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser, "Local deep implicit functions for 3d shape," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4857–4866. 7

[123] Y. Siddiqui, J. Thies, F. Ma, Q. Shan, M. Nießner, and A. Dai, "Retrievalfuse: Neural 3d scene reconstruction with a database," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 7

[124] J. Rossignac, "Shape complexity," *The visual computer*, vol. 21, no. 12, pp. 985–996, 2005. 8

[125] D. Stutz, "A formal definition of watertight meshes," https://davidstutz.de/a-formal-definition-of-watertight-meshes/, 2018, accessed: 2021-12. 8

[126] J. Kontkanen and S. Laine, "Ambient occlusion fields," in *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, 2005, pp. 41–48. 8

[127] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree, "Blensor: Blender sensor simulation toolbox," in *International Symposium on Visual Computing*. Springer, 2011, pp. 199–208. 8, 10

[128] L. Li, "Time-of-flight camera–an introduction," *Technical white paper*, no. SLOA190B, 2014. 8

[129] E. W. Weisstein, "Euler angles," *https://mathworld. wolfram. com/*, 2009. 9

[130] J. Jiao, L. Yuan, W. Tang, Z. Deng, and Q. Wu, "A post-rectification approach of depth images of kinect v2 for 3d reconstruction of indoor scenes," *ISPRS International Journal of Geo-Information*, vol. 6, no. 11, p. 349, 2017. 10

[131] D. Girardeau-Montaut, "Cloudcompare," *France: EDF R&D Telecom ParisTech*, 2016. 11

[132] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008. 12

[133] F. Cazals and M. Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *Computer Aided Geometric Design*, vol. 22, no. 2, pp. 121–146, 2005. 12

[134] J. Yang, R. Li, Y. Xiao, and Z. Cao, "3d reconstruction from non-uniform point clouds via local hierarchical clustering," in *Ninth International Conference on Digital Image Processing (ICDIP 2017)*, vol. 10420. International Society for Optics and Photonics, 2017, p. 1042038. 12

[135] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 605–613. 12, 21

[136] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–13, 2017. 12, 21

[137] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595. 12, 22

[138] A. Fabri and S. Pion, "Cgal: The computational geometry algorithms library," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2009, pp. 538–539. 12

[139] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," in *Eurographics Italian Chapter Conference*, V. Scarano, R. D. Chiara, and U. Erra, Eds. The Eurographics Association, 2008. 12

[140] J. Lin, X. Shi, Y. Gao, K. Chen, and K. Jia, "Cad-pu: A curvature-adaptive deep learning solution for point set upsampling," *arXiv preprint arXiv:2009.04660*, 2020. 21

[141] *3D Shape Analysis: Fundamentals, Theory, and Applications*. Wiley, Mar. 2019. 21

[142] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014. 22

Appendix A
Technical Details for Computing the Algebraic Surface Complexity

As discussed in Section 4.1.2, we have divided all the object surfaces into different groups according to their algebraic complexities measured by the averaged approximation errors, which are computed using a highest-degree fixed function to fit the local surface patches. We present the details in this section.

Assume that the underlying surface $\mathcal{S}^*$ of a given point cloud $\mathcal{P}$ is $\mathcal{C}^\infty$ smooth. By differential geometry, if we approximate the surface $\mathcal{S}^*_{\boldsymbol{p}}$ locally at a point $\boldsymbol{p}$ with a polynomial $\mathcal{S}_{\boldsymbol{p}}(m)$ of degree m, we have the following error bound [50]

$$\|\mathcal{S}^*_{\boldsymbol{p}} - \mathcal{S}_{\boldsymbol{p}}(m)\| \leq C(\|\mathcal{S}^{*(m+1)}_{\boldsymbol{p}}\|) \cdot \omega^{m+1}_{\boldsymbol{p}},$$

where $\omega_{\boldsymbol{p}}$ denotes the width of local 2D domain centered at $\boldsymbol{p}$ for $\mathcal{S}_{\boldsymbol{p}}(m)$ and $C(\|\mathcal{S}^{*(m+1)}_{\boldsymbol{p}}\|)$ is a constant depending on the $(m+1)^{th}$ derivatives of the local surface. Given a surface $\mathcal{S}^*$, we can define $\{\boldsymbol{p}_i\}$ and $\{\omega_{\boldsymbol{p}_i}\}$ to cover the whole surface by $\{\mathcal{S}_{\boldsymbol{p}_i}(m)\}$ with no holes. And in this way, the union of $\{\mathcal{S}_{\boldsymbol{p}_i}(m)\}$ approximates $\mathcal{S}^*$ with the following error bound

$$\int_{\mathcal{S}^*} \|\mathcal{S}^*_{\boldsymbol{p}_i} - \mathcal{S}_{\boldsymbol{p}_i}(m)\| \leq \int_{\mathcal{S}^*} C(\|\mathcal{S}^{*\,(m+1)}_{\boldsymbol{p}_i}\|) \cdot \omega^{m+1}_{\boldsymbol{p}_i}. \quad (17)$$

Considering that the surface $\mathcal{S}^*$ is represented as a triangular mesh $\mathcal{G}_{\mathcal{S}^*}$, a discrete version of Eq. (17) can be written as

$$\sum_{i=1}^{n} \|\mathcal{S}^*_{\boldsymbol{v}_i} - \mathcal{S}_{\boldsymbol{v}_i}(m)\| \leq \sum_{i=1}^{n} C(\|\mathcal{S}^{*\,(m+1)}_{\boldsymbol{v}_i}\|) \cdot \omega^{m+1}_{\boldsymbol{v}_i}, \quad (18)$$

where $\{\boldsymbol{v}_i\}_{i=1}^n$ are the vertices of $\mathcal{G}_{\mathcal{S}^*}$. It is practically reasonable to set $m = 1$ to have a piecewise linear surface approximation, since our visual system is insensitive to surface smoothness beyond second order [140]. Considering a fixed number n of vertices for all meshes, which is the case in our benchmark,; it is clear that comparing algebraic complexities of different surfaces is to compare their respective constants $C(\|\mathcal{S}^{*(2)}_{\boldsymbol{v}}\|)$, which depend on the curvatures of local surfaces. We can compute the curvature as the integral of local, squared normal curvatures

$$\begin{aligned}
&\frac{1}{\pi}\int_0^\pi \kappa_n^2(\boldsymbol{v},\theta)d\theta \\
&= \frac{1}{\pi}\int_0^\pi (\kappa_1(\boldsymbol{v})\cos^2(\theta) + \kappa_2(\boldsymbol{v})\sin^2(\theta))^2 d\theta \\
&= \frac{3}{8}\kappa_1^2(\boldsymbol{v}) + \frac{2}{8}\kappa_1(\boldsymbol{v})\kappa_2(\boldsymbol{v}) + \frac{3}{8}\kappa_2^2(\boldsymbol{v}) \\
&= \frac{3}{2}(\frac{\kappa_1(\boldsymbol{v}) + \kappa_2(\boldsymbol{v})}{2})^2 - \frac{1}{2}\kappa_1(\boldsymbol{v})\kappa_2(\boldsymbol{v}) \\
&= \frac{3}{2}\varkappa^2(\boldsymbol{v}) - \frac{1}{2}\varrho(\boldsymbol{v}),
\end{aligned} \quad (19)$$

where $\kappa_n(\boldsymbol{v},\theta)$ denotes the normal curvature at $\boldsymbol{v}$ in the direction of θ using Euler's famous formula [141]; $\kappa_1(\boldsymbol{v})$ and $\kappa_2(\boldsymbol{v})$ denote the principal curvatures; $\varkappa(\boldsymbol{v})$ and $\varrho(\boldsymbol{v})$ denote the mean and Gaussian curvatures at $\boldsymbol{v}$, respectively. For $\mathcal{S}^*$, we take the expectation of the integral of local, squared normal curvatures over points on the surface as its algebraic complexity

$$\mathbb{E}_{\{\boldsymbol{v}_i\}_{i=1}^n \in \mathcal{G}_{\mathcal{S}^*}}[\frac{3}{2}\varkappa^2(\boldsymbol{v}) - \frac{1}{2}\varrho(\boldsymbol{v})]. \quad (20)$$

Note that a higher value of Eq. (20) indicates a more complicated surface. We finally divide all the object surfaces in our benchmark into three groups respectively of 972, 486, and 162 instances (at the ratio of around 6 : 3 : 1 for low-, middle-, and high-complexity groups) according to their computed surface complexities.

Appendix B
Details of Evaluation Metrics

We use three popular metrics of Chamfer Distance (CD) [135], F-score [136], and Normal Consistency Score (NCS) [9], and also a newly proposed neural metric, termed Neural Feature Similarity (NFS), to quantitatively compare surface meshes obtained from different methods. Details are presented as follows.

B.1 Popular Evaluation Metrics

Chamfer Distance – Chamfer Distance (CD) is a metric between two point sets; in our case, it is defined over two point sets $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^{n_\mathcal{P}}$ and $\mathcal{Q} = \{\boldsymbol{q}_i\}_{i=1}^{n_\mathcal{Q}}$ sampled from the reconstructed surface $\mathcal{G}_\mathcal{S}$ and ground-truth surface $\mathcal{G}_{\mathcal{S}^*}$, respectively. It is a symmetric metric computed as

$$\frac{1}{2n_\mathcal{P}}\sum_{\boldsymbol{p}\in\mathcal{P}}\min_{\boldsymbol{q}\in\mathcal{Q}}\|\boldsymbol{p}-\boldsymbol{q}\|_2 + \frac{1}{2n_\mathcal{Q}}\sum_{\boldsymbol{q}\in\mathcal{Q}}\min_{\boldsymbol{p}\in\mathcal{P}}\|\boldsymbol{p}-\boldsymbol{q}\|_2.$$

We set $n_\mathcal{P} = n_\mathcal{Q} = 200$k and $n_\mathcal{P} = n_\mathcal{Q} = 1500$k for evaluation on the synthetic data of object surfaces and scene surfaces, respectively.

F-score – Similar to CD, F-score is also a metric between two point sets. Different from CD, F-score acts more like an l_0-norm measurement, since it adopts an indication function $\mathbf{1}_{\text{condition}}[\cdot]$ (shortened as $\mathbf{1}_c[\cdot]$) to measure the precision and recall between two point sets, and it also takes the harmonic mean between the precision and recall, which means that it will be dominated by the minimum of either precision or recall. Therefore, though it still specializes in measuring the overall similarity between two shapes, it is more sensitive to extreme cases in which either precision or recall is bad. Specifically, let $\Phi(\boldsymbol{p};\mathcal{Q}) = \min_{\boldsymbol{q}\in\mathcal{Q}}\|\boldsymbol{p}-\boldsymbol{q}\|_2$ and $\Phi(\boldsymbol{q};\mathcal{P}) = \min_{\boldsymbol{p}\in\mathcal{P}}\|\boldsymbol{p}-\boldsymbol{q}\|_2$; the precision can be defined as $D(\mathcal{P};\mathcal{Q},\tau) = \frac{1}{n_\mathcal{P}}\sum_{\boldsymbol{p}\in\mathcal{P}}\mathbf{1}_c[\Phi(\boldsymbol{p};\mathcal{Q}) < \tau]$, and the recall can be defined as $D(\mathcal{Q};\mathcal{P},\tau) = \frac{1}{n_\mathcal{Q}}\sum_{\boldsymbol{q}\in\mathcal{Q}}\mathbf{1}_c[\Phi(\boldsymbol{q};\mathcal{P}) < \tau]$. The overall F-score is computed as

$$\frac{2D(\mathcal{P};\mathcal{Q},\tau)D(\mathcal{Q};\mathcal{P},\tau)}{D(\mathcal{P};\mathcal{Q},\tau) + D(\mathcal{Q};\mathcal{P},\tau)} \times 100,$$

where τ is a hyper-parameter controlling the sensitivity of F-score. We set $\tau = 0.005$ with $n_\mathcal{P} = n_\mathcal{Q} = 200$k and $\tau = 0.03$ with $n_\mathcal{P} = n_\mathcal{Q} = 1500$k for evaluation on the synthetic data of object surfaces and scene surfaces, respectively, and set $n_\mathcal{P} = n_\mathcal{Q}$ to be the number of points in the obtained point clouds by high-precision, real scanning (i.e., the ground-truth surfaces of our real-scanned data), with $\tau = 0.5$.

Normal Consistency Score – Normal Consistency Score (NCS) is a metric measuring the consistency between two vector fields. As such, it measures the difference of higher order information between different surfaces, and is more sensitive to subtle shape differences. It can be computed as

$$\frac{1}{2}\left(\frac{1}{n_\mathcal{P}}\sum_{\boldsymbol{p}\in\mathcal{P}}\left|\left\langle \boldsymbol{n}_{\mathcal{T}_\boldsymbol{p}}, \boldsymbol{n}_{\mathcal{T}_{\Phi(\boldsymbol{p};\mathcal{Q})}}\right\rangle\right| + \frac{1}{n_\mathcal{Q}}\sum_{\boldsymbol{q}\in\mathcal{Q}}\left|\left\langle \boldsymbol{n}_{\mathcal{T}_\boldsymbol{q}}, \boldsymbol{n}_{\mathcal{T}_{\Phi(\boldsymbol{q};\mathcal{P})}}\right\rangle\right|\right),$$

where $\boldsymbol{n}_{\mathcal{T}_\boldsymbol{p}}$ denotes the normal estimated from the continuous triangular facet $\mathcal{T}_\boldsymbol{p}$ at point $\boldsymbol{p}$. We use $\boldsymbol{n}_{\mathcal{T}_\boldsymbol{p}}$ instead of the $\boldsymbol{n}_\boldsymbol{p}$

estimated from the discrete point cloud to reduce the error brought by discrete point sampling. We set $n_{\mathcal{P}} = n_{\mathcal{Q}} = 200$k and $n_{\mathcal{P}} = n_{\mathcal{Q}} = 1500$k for evaluation on the synthetic data of object surfaces and scene surfaces respectively, and set $n_{\mathcal{P}} = n_{\mathcal{Q}}$ to be the number of points in the obtained point clouds by high-precision, real scanning (i.e., the ground-truth surfaces of our real-scanned data).

B.2 The Proposed Neural Evaluation Metric

Neural Feature Similarity – The aforementioned three metrics are popularly used in existing surface reconstruction evaluation; however, they may fail to capture the shape differences that are more consistent with our human perception. Inspired by [137], we propose to compare the similarity between two shapes in the deep feature space, which depends more on the high-level semantic information and the resulting comparison would be more consistent with human perception. More specifically, given the deep features for different surfaces, we use cosine similarity to compute the proposed Neural Feature Similarity (NFS)

$$\frac{\boldsymbol{g_\theta}(\mathcal{P}) \cdot \boldsymbol{g_\theta}(\mathcal{Q})}{\|\boldsymbol{g_\theta}(\mathcal{P})\|_2 \|\boldsymbol{g_\theta}(\mathcal{Q})\|_2},$$

where the feature-extracting network $\boldsymbol{g_\theta}$ takes in a point cloud $\mathcal{P}$ and outputs a feature vector.

We train the network $\boldsymbol{g_\theta}$ in a self-supervised manner, such that features of point clouds sampled from the same surface are as invariant as possible, and features of point clouds sampled from different surfaces are different. We use the following self-supervision training objective

$$\sum_{i,i'} \left| \frac{\boldsymbol{g_\theta}(\mathcal{P}_i) \cdot \boldsymbol{g_\theta}(\mathcal{P}_{i'})}{\|\boldsymbol{g_\theta}(\mathcal{P}_i)\|_2 \|\boldsymbol{g_\theta}(\mathcal{P}_{i'})\|_2} - 1 \right| + \sum_{i,j} \left| \frac{\boldsymbol{g_\theta}(\mathcal{P}_i) \cdot \boldsymbol{g_\theta}(\mathcal{P}_j)}{\|\boldsymbol{g_\theta}(\mathcal{P}_i)\|_2 \|\boldsymbol{g_\theta}(\mathcal{P}_j)\|_2} \right|, \tag{21}$$

where the index pair $\{i, i'\}$ means that the point clouds $\{\mathcal{P}_i, \mathcal{P}_{i'}\}$ are sampled from the same surface, and $\{i, j\}$ for those from different surfaces. To further improve the accuracy, we take the aligned local surface patches as the input, instead of the whole surface, and take the average of the differences overs all the local surfaces to be the final result. We adopt a 6-layer, MLP-based auto-decoder (with the activation function of Leaky ReLU, and 256 channels per layer) as $\boldsymbol{g_\theta}(\cdot)$, and each local surface patch is represented as a 256 dimensional latent vector. We use Adam [142] with the learning rate initially set as 0.0001 and decreased by half every 200 epochs; we train for a total of 1000 epochs.

Appendix C
More Results for the Synthetic Data of Object Surfaces

In this section, we present the overall results on the testing synthetic data of object surfaces as in Table 10.

Appendix D
More Results for Studying the Importance of Orientations of Surface Normals

In this section, we show more results of scene surfaces and real-scanned data for studying the importance of orientations of surface normals. Table 11 shows qualitative results of scene surfaces that compare methods using surfaces normals, those without using surfaces normals, and those using surface normals only during learning. Those on real-scanned data are shown in Table 12.

Appendix E
Experimental Results Without Data Pre-processing

We have reported and discussed the results in Section 5.2 that are obtained *with* the pre-processing pipeline. In this section, we report results *without* the pre-processing pipeline. The results with and without pre-processing are of similar comparative qualities, which are given in Table 13 and Table 14 respectively for synthetic data of object surfaces and synthetic data of scene surfaces. Note that the results of real-scanned data are not involved in this section as the scanners we used automatically pre-process the point clouds with their inbuilt pre-processing methods.

Appendix F
More Results for the Real-scanned Data

Fig. 13 shows more qualitative results on the real-scanned data. The results suggest that quality of a scanned point cloud depends heavily on the surface material. Other observations are consistent with those in Section 6.

TABLE 10: Quantitative results on the testing synthetic data of object surfaces. For those challenges with varying levels of severity, we report results in the format of "· / · / ·", which, from left to right, are results for low-, middle-, and high-level severities of each challenge. Results of the **best** and **second best** methods are highlighted in each column.

Algorithms	CD ($\times 10^{-4}$) ↓					
	Perfect scanning	Non-uniform distrubution	Point-wise noise *low- /middle- /high-level*	Point outliers *low- /middle- /high-level*	Missing points *low- /middle- /high-level*	Misalignment *low- /middle- /high-level*
GD [33]	**14.24**	**14.87**	16.52 / 18.20 / 29.73	103.77 /123.19 / 155.98	**44.33** / 64.15 / **110.74**	16.68 / 21.15 / 39.49
BPA [34]	14.89	17.13	16.59 / 18.58 / 29.30	**14.41** / **14.66** / **16.15**	50.99 / **62.55** / 121.59	16.61 / 20.88 / 42.46
SPSR [36]	14.47	15.36	**15.50** / **16.05** / **18.16**	**14.35** / **14.71** / **28.60**	200.72 /225.66 / 307.75	**15.69** / **17.24** / **25.96**
RIMLS [53]	15.73	16.74	16.13 / **17.17** / 25.88	17.52 /126.40 / 223.90	48.93 / 65.36 / 113.44	17.42 / 21.12 / 41.12
SALD [73]	15.10	**14.96**	**15.33** / 18.77 / 27.59	16.83 / 53.65 / 145.05	**40.96** / **55.63** / **105.23**	**15.50** / 20.09 / 37.16
IGR [71]	18.40	18.33	18.21 / 18.57 / **19.20**	49.11 / 43.60 / 67.25	94.92 /151.53 / 223.64	19.65 / 20.72 / **25.98**
OccNet [9]	201.96	210.80	209.04 /205.21 / 210.01	218.23 /225.85 / 299.09	220.23 /231.65 / 264.24	207.90 /212.51 / 220.07
DeepSDF [8]	229.18	227.42	241.28 /230.40 / 239.63	336.64 /511.36 / 548.08	372.86 /378.58 / 447.27	234.42 /232.16 / 241.64
LIG [10]	23.09	24.03	23.96 / 22.05 / 26.27	25.34 /115.38 / 168.64	52.27 / 70.98 / 115.09	26.03 / 24.30 / 33.34
Points2Surf [78]	17.18	18.81	17.74 / 18.48 / 22.63	28.25 / 83.91 / 144.02	80.84 /102.18 / 159.02	17.82 / 20.36 / 33.98
DSE [114]	**14.26**	15.34	16.07 / 17.89 / 27.61	21.32 /100.37 / 158.73	50.11 / 68.88 / 118.86	16.06 / **20.06** / 37.98
IMLSNet [115]	22.56	23.17	22.64 / 22.67 / 23.73	24.35 / 99.95 / 157.23	54.93 / 74.35 / 123.95	23.24 / 23.77 / 30.76
ParseNet [40]	162.94	161.14	154.11 /135.84 / 139.97	157.15 /176.38 / 213.48	187.96 /195.98 / 248.70	141.95 /136.86 / 141.87

Algorithms	F-score (%) ↑					
	Perfect scanning	Non-uniform distrubution	Point-wise noise *low- /middle- /high-level*	Point outliers *low- /middle- /high-level*	Missing points *low- /middle- /high-level*	Misalignment *low- /middle- /high-level*
GD [33]	**99.66**	**99.07**	99.09 / 98.91 / 91.65	69.42 / 54.59 / 48.63	**91.48** / 87.72 / **80.71**	98.97 / 97.41 / 73.91
BPA [34]	98.70	97.02	98.63 / 98.51 / 89.59	**99.37** / **99.31** / **98.69**	89.98 / **87.85** / 78.56	98.56 / 97.24 / 68.27
SPSR [36]	99.59	99.02	**99.51** / **99.46** / **99.16**	**99.67** / **99.65** / **97.14**	81.85 / 76.91 / 65.41	**99.42** / **99.27** / **91.99**
RIMLS [53]	99.27	98.76	99.36 / **99.24** / 94.69	98.66 / 57.78 / 19.84	**90.78** / **87.92** / **81.00**	98.89 / 97.53 / 74.85
SALD [73]	99.45	**99.10**	**99.54** / 98.94 / 92.48	98.75 / 88.05 / 59.41	89.43 / 85.78 / 76.66	**99.48** / 98.09 / 75.83
IGR [71]	97.54	97.79	97.87 / 97.75 / **97.52**	95.22 / 91.11 / 81.87	79.30 / 70.49 / 56.99	97.01 / 96.46 / **92.55**
OccNet [9]	31.03	29.90	29.85 / 29.77 / 27.92	26.43 / 23.75 / 15.11	25.20 / 24.52 / 21.68	31.23 / 29.19 / 26.42
DeepSDF [8]	17.79	18.81	16.70 / 17.08 / 15.99	12.10 / 4.15 / 3.07	15.43 / 14.06 / 13.73	18.36 / 17.05 / 14.63
LIG [10]	96.20	95.32	94.50 / 96.99 / 93.80	96.48 / 76.69 / 57.33	86.92 / 82.43 / 73.88	93.80 / 95.01 / 84.50
Points2Surf [78]	98.14	97.72	98.02 / 97.39 / 94.21	92.69 / 72.94 / 50.56	81.42 / 76.46 / 66.68	97.98 / 96.49 / 82.90
DSE [114]	**99.64**	98.84	99.40 / 99.17 / 92.75	95.13 / 52.21 / 30.82	90.72 / 87.71 / 80.19	99.29 / **98.20** / 75.07
IMLSNet [115]	94.82	94.51	94.41 / 94.36 / 94.40	94.78 / 64.55 / 38.53	83.39 / 80.01 / 72.60	94.18 / 94.14 / 87.44
ParseNet [40]	40.52	38.82	44.80 / 44.13 / 42.66	41.20 / 37.21 / 34.04	37.12 / 41.28 / 41.68	44.99 / 46.60 / 43.84

Algorithms	NCS ($\times 10^{-2}$) ↑					
	Perfect scanning	Non-uniform distrubution	Point-wise noise *low- /middle- /high-level*	Point outliers *low- /middle- /high-level*	Missing points *low- /middle- /high-level*	Misalignment *low- /middle- /high-level*
GD [33]	98.57	98.05	94.00 / 87.58 / 68.32	94.94 / 92.17 / 86.34	**96.44** / **95.17** / **93.19**	94.94 / 84.96 / 65.74
BPA [34]	98.37	97.89	95.36 / 91.68 / 77.97	98.17 / **98.07** / **97.17**	95.18 / 94.42 / 91.91	96.03 / 90.12 / 71.33
SPSR [36]	98.58	**98.38**	**97.84** / **97.03** / 93.89	**98.61** / **98.56** / **97.87**	91.52 / 89.99 / 85.91	**97.99** / **96.24** / 91.86
RIMLS [53]	98.19	97.77	97.36 / 95.23 / 86.13	97.70 / 83.42 / 67.51	94.43 / 93.27 / 90.93	96.66 / 92.48 / 79.68
SALD [73]	**98.67**	**98.52**	**98.07** / 96.42 / 88.91	**98.39** / 95.32 / 89.92	**96.17** / **95.19** / 91.71	**98.10** / 94.73 / 86.26
IGR [71]	97.62	97.59	97.64 / **97.52** / **97.10**	96.99 / 96.47 / 94.30	92.86 / 90.61 / 87.69	97.35 / **97.04** / **95.97**
OccNet [9]	79.55	79.30	79.43 / 79.64 / 79.05	79.33 / 78.55 / 74.60	78.97 / 78.43 / 77.10	79.08 / 79.58 / 78.69
DeepSDF [8]	78.65	79.11	78.01 / 77.78 / 78.52	74.65 / 73.40 / 75.85	74.97 / 74.52 / 72.29	78.59 / 78.12 / 77.69
LIG [10]	96.49	95.79	93.96 / 94.22 / 88.26	96.81 / 91.66 / 87.75	91.76 / 89.56 / 85.37	93.98 / 92.70 / 87.32
Points2Surf [78]	95.24	95.09	94.97 / 94.62 / 93.44	94.04 / 87.87 / 83.17	88.41 / 86.36 / 82.05	95.14 / 94.48 / 90.33
DSE [114]	**98.60**	97.86	94.43 / 87.79 / 71.39	94.98 / 77.34 / 73.29	95.43 / 94.40 / **92.49**	95.13 / 86.20 / 68.12
IMLSNet [115]	96.13	95.98	96.08 / 96.02 / **95.62**	95.75 / 87.45 / 79.76	92.22 / 90.48 / 86.66	95.93 / 95.87 / **93.84**
ParseNet [40]	77.71	76.89	78.55 / 80.31 / 81.15	77.73 / 75.46 / 75.48	75.10 / 75.83 / 74.82	78.40 / 80.48 / 79.31

Algorithms	NFS ($\times 10^{-2}$) ↑					
	Perfect scanning	Non-uniform distrubution	Point-wise noise *low- /middle- /high-level*	Point outliers *low- /middle- /high-level*	Missing points *low- /middle- /high-level*	Misalignment *low- /middle- /high-level*
GD [33]	95.22	95.19	91.57 / 84.63 / 56.30	57.51 / 37.47 / 30.62	**86.87** / **82.93** / **75.28**	92.04 / 80.69 / 51.41
BPA [34]	94.10	93.60	90.71 / 87.49 / 65.15	**96.46** / **93.24** / **82.34**	84.23 / 80.36 / 70.09	91.91 / 81.96 / 59.61
SPSR [36]	**96.38**	**96.22**	**95.60** / **94.98** / **91.99**	**96.69** / **96.31** / **90.02**	74.48 / 69.34 / 55.89	**95.58** / **94.28** / **87.03**
RIMLS [53]	95.01	94.02	94.19 / 92.67 / 79.76	92.61 / 62.01 / 24.49	83.71 / 79.23 / 69.24	92.19 / 87.12 / 68.26
SALD [73]	**96.11**	**96.65**	**95.66** / 89.72 / 67.18	83.22 / 51.74 / 34.70	**87.38** / **84.01** / **73.50**	**95.38** / 88.26 / 63.13
IGR [71]	94.71	94.22	94.37 / **94.52** / **93.67**	90.31 / 84.63 / 74.59	75.56 / 68.14 / 56.29	93.16 / **92.54** / **90.18**
OccNet [9]	47.33	46.55	46.17 / 46.03 / 45.04	44.29 / 42.11 / 30.77	44.18 / 42.46 / 39.09	46.75 / 45.80 / 43.73
DeepSDF [8]	39.94	40.91	38.55 / 39.29 / 38.70	31.69 / 16.65 / 11.92	33.54 / 31.59 / 28.93	39.31 / 39.26 / 37.39
LIG [10]	91.58	90.66	86.71 / 89.55 / 81.03	89.40 / 66.21 / 46.81	80.10 / 74.98 / 66.64	85.20 / 84.34 / 77.92
Points2Surf [78]	93.45	93.23	92.83 / 92.59 / 90.21	86.59 / 63.30 / 41.44	74.47 / 68.53 / 57.08	92.96 / 91.59 / 83.19
DSE [114]	94.50	94.75	90.05 / 83.53 / 57.08	85.39 / 42.32 / 27.29	85.40 / 79.62 / 71.06	89.87 / 76.63 / 52.40
IMLSNet [115]	90.61	90.20	90.13 / 89.97 / 89.66	89.16 / 52.82 / 29.13	79.12 / 74.59 / 65.23	89.68 / 89.19 / 85.55
ParseNet [40]	38.54	37.71	46.21 / 49.30 / 49.00	38.19 / 35.98 / 34.58	36.52 / 38.40 / 35.07	44.52 / 45.73 / 45.76

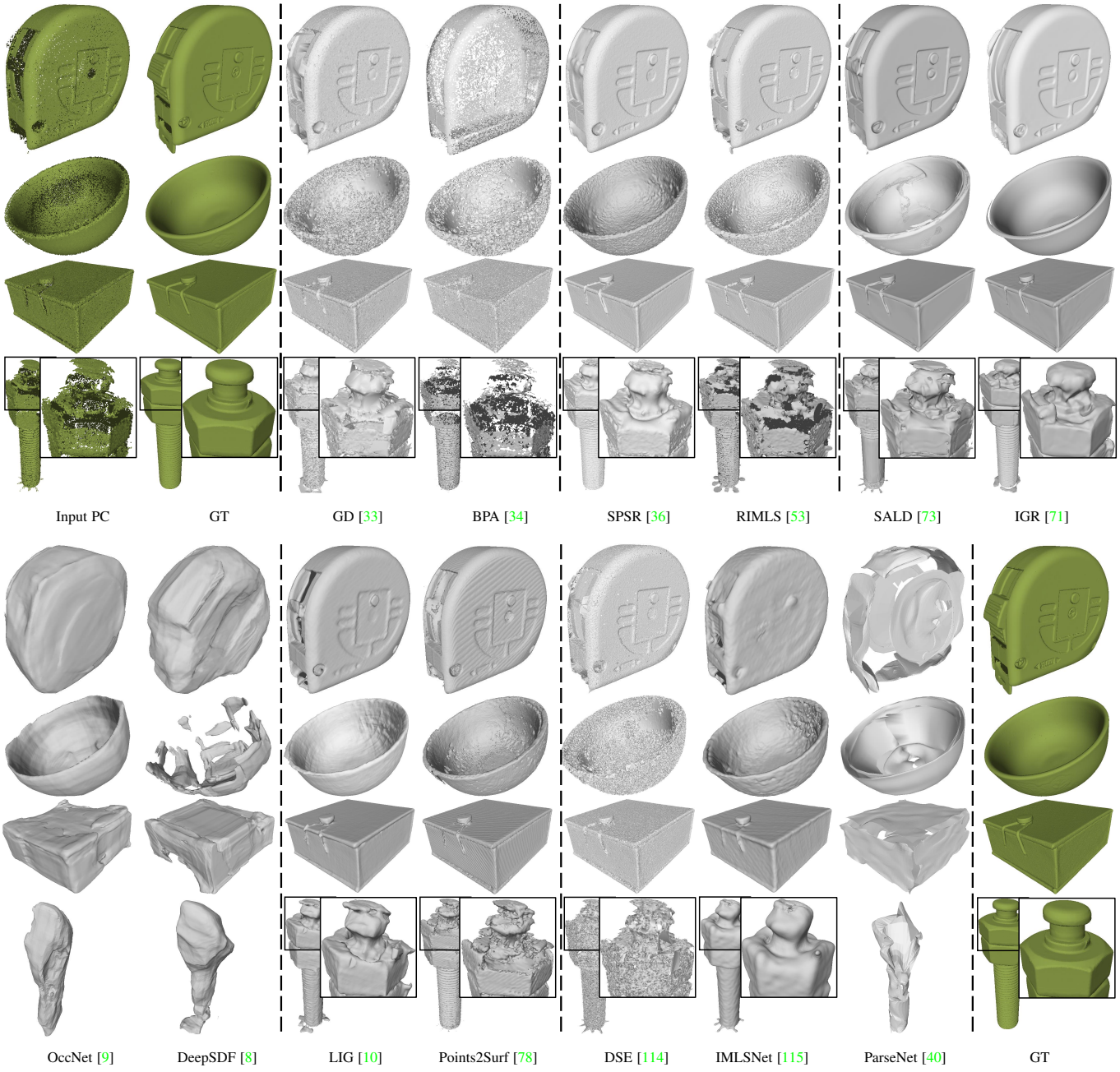


Fig. 13: Additional qualitative results on the real-scanned data.

TABLE 11: Comparison on the testing synthetic data of scene surfaces among methods without using surface normals ($\times$), methods using surfaces normals ($\checkmark$), and methods using surface normals only during learning ($*$). Results are in the format of "$\cdot$ / $\cdot$", where the left one is obtained assuming the availability of ground-truth camera poses, and the right one is obtained without knowing the camera poses. The **best** and **second best** methods are highlighted in each column.

Algorithms	Normals	CD ($\times 10^{-3}$) $\downarrow$	F-score (%) $\uparrow$	NCS ($\times 10^{-2}$) $\uparrow$	NFS ($\times 10^{-2}$) $\uparrow$
GD [33]		33.85 /	75.95 /	62.08 /	41.92 /
SALD [73]	$\times$	32.43 /	79.03 /	**91.58** /	52.79 /
DSE [114]		32.97 /	77.53 /	57.99 /	41.56 /
BPA [34]		45.82 / 57.08	53.46 / 47.57	58.25 / 44.04	44.19 / 32.01
SPSR [36]		**30.47** / 142.65	**83.22** / 68.06	83.74 / 61.31	**63.03** / 41.69
RIMLS [53]		41.45 / 55.79	74.56 / 65.14	69.93 / 52.16	38.13 / 26.55
IGR [71]	$\checkmark$	**31.41**/ 352.04	**81.63** / 63.37	**91.26** / 67.33	**67.58** / 42.20
OccNet [9]		93.12 / 120.85	37.75 / 32.42	85.98 / 64.12	50.34 / 36.75
DeepSDF [8]		- / -	- / -	- / -	- / -
LIG [10]		41.40 / 81.22	78.03 / 64.12	88.12 / 64.53	59.97 / 39.92
ParseNet [40]		- / -	- / -	- / -	- / -
Points2Surf [78]	$*$	36.24 /	76.14 /	83.60 /	61.82 /
IMLSNet [115]		35.52 /	78.05 /	87.17 /	61.98 /

TABLE 12: Comparison on the testing real-scanned data among methods without using surface normals ($\times$), methods using surfaces normals ($\checkmark$), and methods using surface normals only during learning ($*$) . Results are in the format of "$\cdot$ / $\cdot$", where the left one is obtained assuming the availability of ground-truth camera poses, and the right one is obtained without knowing the camera poses. The **best** and **second best** methods are highlighted in each column.

Algorithms	Normals	CD ($\times 10^{-2}$) $\downarrow$	F-score (%) $\uparrow$	NCS ($\times 10^{-2}$) $\uparrow$	NFS ($\times 10^{-2}$) $\uparrow$
GD [33]		31.72 /	87.51 /	88.86 /	82.20 /
SALD [73]	$\times$	**31.13** /	**87.72** /	94.68 /	86.86 /
DSE [114]		32.16 /	86.88 /	87.20 /	76.81 /
BPA [34]		40.37 / 40.94	80.95 / 80.49	87.56 / 87.17	68.69 / 66.80
SPSR [36]		**31.05** / 126.89	**87.74** / 80.67	**94.94** / 92.64	**89.38** / 79.17
RIMLS [53]		32.80 / 36.39	87.05 / 85.50	91.97 / 90.84	85.19 / 79.64
IGR [71]	$\checkmark$	32.70 / 467.06	87.18 / 76.68	**95.99** / 93.79	**89.10** / 75.04
OccNet [9]		232.71 / 252.81	17.11 / 16.57	80.96 / 80.92	39.70 / 38.52
DeepSDF [8]		263.92 / 552.25	19.83 / 15.16	77.95 / 77.66	40.95 / 35.21
LIG [10]		48.75 / 79.89	83.76 / 76.38	92.57 / 89.46	81.48 / 72.21
ParseNet [40]		149.96 / 217.29	38.92 / 31.06	81.51 / 77.30	45.67 / 38.48
Points2Surf [78]	$*$	48.93 /	80.89 /	89.52 /	81.83 /
IMLSNet [115]		38.46 /	82.44 /	93.31 /	85.30 /

TABLE 13: Quantitative results on the testing synthetic data of object surfaces when the input point clouds are NOT pre-processed (cf. Section 5.2 for details). For those challenges with varying levels of severity, we report results in the format of "· / · / ·", which, from left to right, are the results for low-, middle-, and high-level severities of each challenge. Results of the **best** and **second best** methods are highlighted in each column.

Algorithms	CD ($\times 10^{-4}$) ↓					
	Perfect scanning	Non-uniform distrubution	Point-wise noise *low- /middle- /high-level*	Point outliers *low- /middle- /high-level*	Missing points *low- /middle- /high-level*	Misalignment *low- /middle- /high-level*
GD [33]	**14.35**	**14.87**	21.39 / 34.53 / 46.66	210.20 /581.89 / 233.24	**33.66** / **65.23** / **97.21**	21.65 / 40.52 / 56.94
BPA [34]	18.60	32.68	23.76 / 49.42 / 55.63	**17.81** / **18.94** / **25.73**	83.17 / 78.99 / 126.46	56.64 / 36.90 / 68.30
SPSR [36]	14.53	15.51	**15.89** / **18.57** / **22.99**	**14.29** / **14.97** / **16.50**	194.67 /220.89 / 300.03	**16.39** / **19.72** / **31.65**
RIMLS [53]	15.83	17.23	18.44 / 26.38 / 43.81	40.14 /344.67 / 375.24	41.91 / 66.17 / 98.40	21.89 / 30.58 / 62.71
SALD [73]	15.47	**14.99**	22.98 / 29.76 / 43.16	26.57 / 58.14 / 133.80	**32.38** / **56.09** / 102.65	20.34 / 28.50 / 51.31
IGR [71]	17.38	18.36	**18.29** / **19.91** / **21.83**	18.88 / 47.09 / 72.82	149.17 /210.41 / 285.78	**19.14** / 26.10 / **29.28**
OccNet [9]	205.54	207.77	201.56 /214.18 / 205.90	215.38 /235.77 / 277.20	213.21 /227.19 / 271.17	213.89 /215.39 / 231.11
DeepSDF [8]	232.13	238.46	241.98 /244.25 / 257.42	496.53 /681.94 / 797.70	369.52 /362.66 / 419.42	250.07 /246.97 / 264.91
LIG [10]	25.79	34.28	26.99 / 30.14 / 29.71	93.13 /180.10 / 274.45	88.36 / 93.67 / 134.45	29.28 / 31.76 / 38.58
Points2Surf [78]	20.99	22.22	23.47 / 25.21 / 29.34	70.12 /114.66 / 138.48	71.47 / 98.03 / 164.85	21.81 / 26.36 / 38.98
DSE [114]	**14.38**	15.40	19.32 / 27.54 / 40.59	145.71 /227.83 / 254.67	43.73 /106.05 / **83.63**	19.81 / 28.73 / 49.27
IMLSNet [115]	22.98	23.37	22.93 / 23.99 / 33.09	124.95 /220.04 / 261.79	46.41 / 69.55 / 107.22	21.88 / **23.97** / 45.91
ParseNet [40]	156.04	144.21	165.10 /158.52 / 157.96	188.92 /221.36 / 238.92	180.37 /210.85 / 289.67	126.48 /126.36 / 154.52

Algorithms	F-score (%) ↑					
	Perfect scanning	Non-uniform distrubution	Point-wise noise *low- /middle- /high-level*	Point outliers *low- /middle- /high-level*	Missing points *low- /middle- /high-level*	Misalignment *low- /middle- /high-level*
GD [33]	**99.64**	98.09	97.85 / 86.53 / 61.66	52.79 / 46.20 / 41.27	**93.14** / **86.66** / **82.51**	97.93 / 82.74 / 53.61
BPA [34]	93.15	89.64	92.90 / 81.79 / 62.05	94.58 / **94.27** / **93.55**	83.20 / 79.38 / 72.94	87.90 / 79.74 / 57.62
SPSR [36]	99.50	**98.79**	**99.54** / **99.36** / **97.26**	**99.67** / **99.57** / **99.45**	82.54 / 77.00 / 65.45	**99.48** / **98.76** / **84.70**
RIMLS [53]	99.18	98.39	**98.88** / 94.80 / 66.18	93.64 / 15.89 / 8.25	91.63 / **86.51** / 81.59	97.69 / 89.09 / 56.96
SALD [73]	99.47	**99.07**	98.47 / 88.80 / 62.65	97.41 / 87.41 / 73.92	**91.69** / 85.37 / 78.47	**98.82** / 88.21 / 55.47
IGR [71]	97.81	98.23	98.26 / **97.25** / **96.53**	**97.69** / 90.70 / 85.29	77.76 / 69.29 / 56.88	97.22 / **96.35** / **90.88**
OccNet [9]	29.69	29.58	30.37 / 27.71 / 25.01	26.74 / 20.61 / 16.64	26.25 / 24.22 / 21.88	29.67 / 28.69 / 22.36
DeepSDF [8]	18.56	17.57	16.37 / 15.23 / 12.35	6.32 / 1.69 / 1.56	14.54 / 14.95 / 13.37	17.10 / 13.78 / 10.39
LIG [10]	93.94	92.69	92.37 / 92.18 / 88.73	86.32 / 68.05 / 42.68	78.97 / 75.23 / 67.96	92.31 / 90.28 / 77.47
Points2Surf [78]	95.02	94.31	93.94 / 93.77 / 91.17	86.08 / 71.77 / 59.91	81.36 / 75.14 / 64.84	94.70 / 93.40 / 79.40
DSE [114]	**99.59**	98.43	98.87 / 92.28 / 68.84	57.21 / 25.77 / 16.87	91.52 / 86.17 / **82.10**	98.76 / 89.74 / 58.59
IMLSNet [115]	95.47	94.34	95.30 / 94.44 / 85.38	75.40 / 42.20 / 30.26	85.89 / 79.90 / 74.89	96.09 / 94.74 / 66.41
ParseNet [40]	40.99	42.40	41.24 / 43.34 / 40.14	38.23 / 35.00 / 32.44	41.01 / 41.40 / 36.30	47.91 / 48.22 / 40.85

Algorithms	NCS ($\times 10^{-2}$) ↑					
	Perfect scanning	Non-uniform distrubution	Point-wise noise *low- /middle- /high-level*	Point outliers *low- /middle- /high-level*	Missing points *low- /middle- /high-level*	Misalignment *low- /middle- /high-level*
GD [33]	97.97	97.70	80.31 / 69.32 / 57.33	89.44 / 83.99 / 81.95	**96.45** / **95.10** / **92.79**	80.56 / 65.52 / 57.00
BPA [34]	95.10	95.54	82.88 / 73.14 / 60.40	93.38 / 90.07 / 92.08	93.55 / 91.15 / **91.64**	85.24 / 67.24 / 63.88
SPSR [36]	**98.15**	**97.91**	**95.89** / 93.51 / 91.06	**98.14** / **97.11** / **96.98**	91.38 / 89.74 / 85.60	**95.79** / 93.19 / 88.58
RIMLS [53]	97.41	96.66	93.10 / 82.03 / 70.01	94.64 / 71.32 / 65.36	93.69 / 92.39 / 89.56	92.01 / 80.65 / 69.78
SALD [73]	**98.44**	**98.27**	90.35 / 87.75 / 77.57	**97.30** / 94.56 / 90.70	**96.84** / **95.05** / 90.76	92.79 / 87.70 / 78.10
IGR [71]	97.23	97.17	**97.13** / **96.99** / **96.34**	97.13 / **95.00** / **93.13**	90.53 / 88.92 / 85.10	**96.69** / **96.12** / **94.78**
OccNet [9]	79.22	79.60	79.74 / 79.32 / 79.59	78.40 / 76.36 / 73.57	79.25 / 78.71 / 76.77	79.10 / 79.32 / 78.96
DeepSDF [8]	78.14	78.59	78.37 / 78.37 / 78.02	72.72 / 73.01 / 72.32	74.37 / 75.04 / 72.09	78.13 / 78.36 / 78.12
LIG [10]	95.36	95.16	94.67 / 94.25 / **93.59**	93.36 / 88.43 / 83.12	88.84 / 87.37 / 84.52	94.73 / 93.99 / **92.58**
Points2Surf [78]	94.06	93.24	93.11 / 92.84 / 91.54	90.57 / 86.71 / 84.08	88.35 / 85.21 / 80.18	92.98 / 92.38 / 88.55
DSE [114]	98.07	97.15	80.51 / 69.65 / 54.34	75.48 / 66.52 / 65.12	94.90 / 93.58 / 90.60	80.88 / 65.81 / 54.49
IMLSNet [115]	95.64	95.57	95.20 / **95.34** / 93.57	89.09 / 80.42 / 77.37	92.72 / 91.10 / 87.88	95.56 / **95.17** / 89.83
ParseNet [40]	77.50	77.69	78.33 / 79.53 / 79.42	75.94 / 74.43 / 72.79	75.27 / 73.73 / 71.79	79.76 / 79.29 / 79.11

Algorithms	NFS ($\times 10^{-2}$) ↑					
	Perfect scanning	Non-uniform distrubution	Point-wise noise *low- /middle- /high-level*	Point outliers *low- /middle- /high-level*	Missing points *low- /middle- /high-level*	Misalignment *low- /middle- /high-level*
GD [33]	94.72	94.56	78.07 / 49.98 / 40.95	26.75 / 25.46 / 21.15	**90.93** / **81.35** / **77.08**	69.98 / 47.13 / 38.67
BPA [34]	89.93	86.55	77.09 / 56.07 / 43.78	88.92 / 84.62 / **82.52**	78.29 / 72.80 / 61.47	70.53 / 52.23 / 48.60
SPSR [36]	**95.90**	**95.31**	**94.84** / **90.69** / 85.53	**95.81** / **95.61** / **94.90**	75.09 / 69.26 / 55.82	**93.90** / **92.91** / 80.91
RIMLS [53]	93.49	92.67	91.92 / 75.15 / 56.07	86.43 / 25.17 / 14.58	84.01 / 79.01 / 69.45	86.27 / 73.19 / 51.89
SALD [73]	**95.08**	**96.52**	72.18 / 65.01 / 53.61	61.66 / 39.89 / 30.76	**90.92** / **84.03** / **72.99**	82.22 / 64.24 / 50.80
IGR [71]	94.46	94.61	**94.13** / **93.96** / **91.60**	**93.66** / **85.83** / 76.25	71.72 / 64.63 / 52.78	**93.17** / **91.31** / **88.68**
OccNet [9]	46.30	47.22	47.29 / 45.31 / 44.84	45.55 / 39.08 / 32.95	45.53 / 42.96 / 38.41	46.00 / 45.48 / 41.75
DeepSDF [8]	39.54	39.18	38.20 / 37.29 / 35.34	17.18 / 5.37 / 4.03	33.24 / 34.14 / 28.85	38.74 / 36.44 / 32.99
LIG [10]	89.82	88.38	89.19 / 87.15 / 86.05	77.04 / 55.26 / 29.19	70.92 / 68.13 / 60.68	87.83 / 86.15 / **81.80**
Points2Surf [78]	91.02	90.35	90.28 / 88.29 / **86.35**	72.88 / 52.99 / 38.05	76.78 / 69.20 / 55.50	90.01 / 87.84 / 79.58
DSE [114]	94.10	93.07	81.08 / 54.83 / 40.44	42.61 / 21.45 / 16.33	84.40 / 79.08 / 68.31	70.44 / 52.63 / 40.18
IMLSNet [115]	90.61	90.66	90.08 / 90.17 / 79.69	52.75 / 25.59 / 14.72	81.60 / 76.77 / 67.82	89.98 / 87.68 / 71.86
ParseNet [40]	37.24	39.70	40.71 / 46.73 / 45.38	37.68 / 34.04 / 30.64	37.56 / 35.37 / 30.72	46.88 / 44.80 / 43.49

TABLE 14: Quantitative results on the testing synthetic data of scene surfaces when the input point clouds are Not preprocessed (cf. Section 5.2 for details). Results of the **best** and **second best** methods are highlighted in each column. "-" indicates that the method cannot produce reasonable results due to their limited generalization.

Prior	Algorithm	CD ($\times 10^{-3}$) ↓	F-score (%) ↑	NCS ($\times 10^{-2}$) ↑	NFS ($\times 10^{-2}$) ↑
Triangulation-based	GD [33]	34.50	71.33	56.78	38.77
	BPA [34]	49.01	55.45	53.52	39.87
Smoothness	SPSR [36]	**30.63**	**83.03**	83.50	**62.46**
	RIMLS [53]	42.53	69.74	64.83	33.82
Modeling	SALD [73]	32.87	75.15	**91.29**	51.36
	IGR [71]	**32.82**	**81.37**	**90.32**	**66.71**
Learning Semantic	OccNet [9]	102.75	34.60	85.01	45.95
	DeepSDF [8]	-	-	-	-
Local Learning	LIG [10]	42.03	67.42	74.84	51.86
	Points2Surf [78]	39.60	75.00	82.25	59.33
Hybird	DSE [114]	33.46	72.35	52.70	38.37
	IMLSNet [115]	36.22	75.04	86.45	59.40
	ParseNet [40]	-	-	-	-