Learning Surface Reconstruction from Point Clouds in the Wild

Raphael Sulzer

Université Gustave Eiffel

Learning Surface Reconstruction from Point Clouds in the Wild

Thèse de doctorat de l'Université Gustave Eiffel

École doctorale n° 532, Mathématiques, Science, et Technologie de l'Information et de la Communication (MSTIC)
Spécialité de doctorat : Signal, Image, et Automatique
Unité de recherche : Laboratoire des Sciences et Technologies de l'Information Geographique (LASTIG), IGN.

Thèse présentée et soutenue à l'Université Gustave Eiffel, le 17/10/2022, par

Raphael SULZER

Composition du Jury

Pierre ALLIEZ Directeur de recherche, TITANE, INRIA, France	Président, Rapporteur
Julie DIGNE Chargée de recherche, LIRIS, CNRS, France	Rapportrice
Michael WIMMER Professeur, Faculty of Informatics, TU WIEN, Autriche	Examinateur
Renaud MARLET Directeur de recherche, LIGM, Ecole des Ponts, France	Co-Directeur de thèse
Bruno VALLET Directeur de recherche, LASTIG, IGN-ENSG, France	Directeur de thèse
Loïc LANDRIEU Chargé de recherche, LASTIG, IGN-ENSG, France	Encadrant

Learning Surface Reconstruction from Point Clouds in the Wild

Abstract

Modern 3D acquisition technology unlocks the possibility to represent the world under the form of 3D point clouds. However, point clouds are usually not sufficient to model complex physical processes. Instead, a variety of applications in science and engineering require a representation of objects or scenes under the form of a continuous surface.

In this thesis, we consider the problem of surface reconstruction from point clouds using supervised deep learning techniques. In particular, we are interested in point clouds in the wild, *i.e.* generated from measurements outside of the laboratory, either directly with 3D scanners or indirectly through multi-view stereo. Such point clouds often depict large scenes with multiple different objects and clutter, and include defects such as noise, outliers, non-uniform sampling or missing data. These characteristics complicate the reconstruction of a topologically and geometrically accurate surface from point clouds in the wild.

After having been successfully deployed on many related computer vision tasks, supervised deep learning has recently been used to address the surface reconstruction problem. Deep surface reconstruction (DSR) can learn point cloud defects or surface patterns from a given training set, and use the learned knowledge during reconstruction. However, current DSR methods exhibit two main limitations. First, supervised deep learning often requires a lot of data to train. However, point clouds in the wild typically depict complex objects or scenes, making it costly, ambiguous, or intractable to gather true surfaces. Second, existing DSR algorithms are often too computation- and memory-intensive to process millions of points. In this thesis, we address both issues by introducing novel supervised deep learning methods to handle large-scale point clouds with real-world characteristics while only training on small synthetic datasets.

The thesis includes three main contributions. First, we survey and benchmark several surface reconstruction methods, including learning and traditional approaches proposed over the last three decades. To make the problem tractable and produce geometrically and topologically accurate results even under challenging conditions, non-learning methods often rely on priors on the input point cloud or output surface. In contrast, DSR algorithms learn these priors directly from the training set of point clouds and corresponding true surfaces. We benchmark different methods on the task of reconstructing

objects from synthetically scanned defect-laden point clouds. Our findings show that DSR methods are able to reconstruct accurate and complete surfaces from point clouds with moderate defects, given that similar defects are also present during training. However, the reconstruction quality from point clouds with unseen defect type is often worse compared to non-learning methods. Traditional methods, on the other hand, show a high robustness to defects, even with constant parametrization for different inputs.

Another shortcoming of most learning based methods is the fact that they ignore sensor poses and only operate on point locations. Sensor visibility holds meaningful information regarding space occupancy and surface orientation. We present two simple strategies to augment point clouds with visibility information, which can directly be integrated with various DSR architectures with minimal adaptation. Our proposed modifications consistently improve the accuracy of generated surfaces, as well as the capability of the networks to generalize to unseen domains. We also release synthetically scanned versions of popular shape datasets to encourage the development of DSR algorithms capable of using visibility information.

Lastly, we introduce Delaunay-Graph Neural Networks (DGNNs), a novel learning-based visibility-aware surface reconstruction method for large-scale point clouds in the wild. DGNN relies on a 3D Delaunay tetrahedralisation of the input point cloud, whose cells are classified as inside or outside the surface by a graph neural network and an energy model solvable with a graph cut. The graph neural network makes use of both local geometric attributes and line-of-sight visibility information to learn a visibility model from a small amount of synthetic training data while generalizing to real-life acquisitions.

Reconstruction de Surfaces à partir de Nuages de Points par Apprentissage Profond

Résumé

Les technologies d'acquisition 3D récentes permettent de représenter le monde sous la forme de nuages de points 3D. Cependant, ces nuages de points ne sont généralement pas suffisants pour modéliser des processus physiques complexes. Au contraire, de nombreuses applications en sciences et en ingénierie nécessitent une représentation sous la forme d'une surface continue.

Dans cette thèse, nous considérons le problème de reconstruction de surface à partir de nuages de points par apprentissage profond supervisé. En particulier, nous nous intéressons à la reconstruction de surface à partir de nuages de points réels, c'est-à-dire générés à partir de mesures effectuées sur le terrain: soit directement avec des scanners 3D, soit indirectement par photogrammetrie. Ces nuages représentent souvent de grandes scènes contenant de multiples objets de formes diverses. Ces nuages peuvent aussi inclure des défauts tels que du bruit d'acquisition, des valeurs aberrantes, un échantillonnage non uniforme ou des données manquantes, ce qui complique la reconstruction d'une surface topologiquement et géométriquement précise.

Après avoir été utilisé avec succès pour de nombreuses tâches de vision par ordinateur, l'app-rentissage profond supervisé a récemment été appliqué au problème de reconstruction de surface. Cependant, les méthodes courantes souffrent encore de deux principales limitations. Tout d'abord, l'apprentissage profond supervisé nécessite souvent un grand nombre de données annotées. Les nuages de points réels décrivent des objets ou des scènes complexes, ce qui rend la collecte de surfaces réelles coûteuse, ambiguë ou mathématiquement difficile. Deuxièmement, les algorithmes d'apprentissage existants sont souvent trop gourmands en calcul et en mémoire pour traiter des millions de points simultanément. Nous abordons ces deux problèmes en introduisant de nouvelles méthodes d'apprentissage profond supervisé pour traiter des nuages de points à grande échelle avec des caractéristiques du monde réel tout en étant entraînées sur de petits ensembles de données synthétiques.

Cette thèse comprend trois contributions principales. Tout d'abord, nous passons en revue et évaluons plusieurs méthodes de reconstruction de surface à partir de nuages de points. En plus des méthodes d'apprentissage, nous évaluons certaines des approches traditionelles proposées au cours des trois dernières décennies. Pour rendre le problème

tractable et produire des résultats géométriquement et topologiquement précis même dans des conditions difficiles, les méthodes sans apprentissage reposent souvent sur des hypothèses sur la structure des nuages de points en entrées ou des surfaces reconstruites. En revanche, les algorithmes de reconstruction de surfaces par apprentissage profond (DSR) apprennent ces hypothèses directement à partir d'un ensemble d'entraînement de nuages de points et des surfaces réelles leur correspondant. Nous évaluons les méthodes d'apprentissage et traditionnellles pour la tâche de reconstruction d'objets à partir de nuages de points avec défauts scannés synthétiquement. Nos résultats montrent que les méthodes DSR sont capables de reconstruire des surfaces précises et complètes à partir de nuages de points présentant un degré modéré de défauts atténués, à condition que ces défauts soient présents pendant l'entraînement. Cependant, la qualité de la reconstruction pour les nuages de points avec défauts non presents dans l'ensemble d'entraînement est souvent moins bonne que celle des méthodes sans apprentissage. Les méthodes sans apprentissage, en revanche, sont d'une grande robustesse aux défauts, même avec une paramétrisation constante pour différentes entrées.

Un autre défaut de la plupart des méthodes DSR est le fait qu'elles ignorent la pose des capteurs et n'opèrent que sur la position des points. La visibilité des capteurs contient pourtant des informations importantes sur l'occupation de l'espace et l'orientation de la surface. Nous présentons deux façons simples d'enrichir les nuages de points avec des informations de visibilité, qui peuvent être directement exploitées par des réseaux de reconstruction de surface en ne nécéssitant qu'une adaptation minimale. Nous montrons que les modifications proposées améliorent systématiquement la précision des surfaces générées ainsi que la capacité des réseaux à généraliser à des nouveaux domaines. Nous publions également les versions scannées synthétiquement de base de données de formes 3D largement utilisées, afin d'encourager le développement d'algorithmes DSR capables d'utiliser les informations de visibilité. Enfin, nous présentons une nouvelle méthode de reconstruction de surface basée sur l'apprentissage et tenant compte de la visibilité pour les nuages de points réels à grande échelle. Notre méthode repose sur une triangulation 3D de Delaunay (3DT) dont les cellules sont classées comme intérieur ou extérieur de la surface recherchée par un réseau de convolution sur graphe (GNN) et un modèle énergétique résolvable avec une coupe de graphe. Le GNN utilise à la fois des attributs géométriques locaux et des informations de visibilité pour apprendre un modèle de visibilité à partir d'une petite quantité de données de formes synthétiques tout en généralisant aux acquisitions réelles.

Contents

List of Figures

List of Tables

Glossary

Acknowledgments

I thank my supervisors Loïc, Bruno and Renaud. They provided a great amount of inspiration and experience during my PhD and were good team mates :). Thank you!

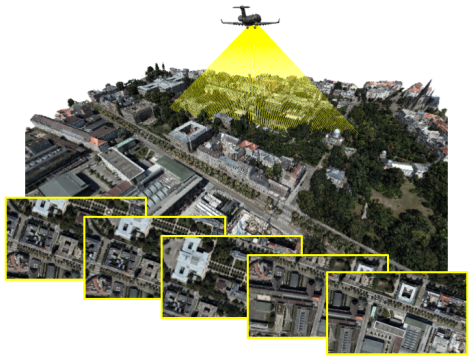
I thank the whole LaSTIG lab for welcoming me with open arms: Alexandre, Ali, Amin, Anatol, Arno, Clément, Damien, Evelyn, Ewelina, Lâmân, Laurent, Mathieu, Mehdi, Mohamed, Nathan, Oussama, Paul, Qasem, Stéphane, Teng, Romain, Yanis, Yilin and especially Vivien and other great friends I met in the past 3,5 years in Paris: Danijela, Adrien, Solène, Diwan, Maëlle, Malo, Iris, Mathilde, Negin, Juliette and Sasha.

Special thanks also go to Arkose Montreuil, la Forêt de Fontainebleau et le Bois de Vincennes for being excellent places to climb and relax.

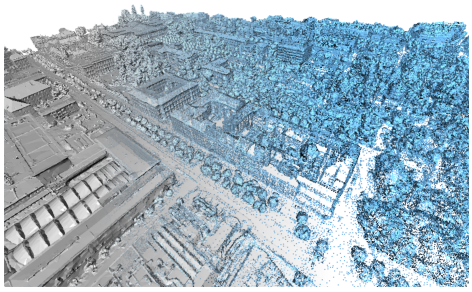
1 Introduction

Modern three-dimensional (3D) acquisition technology, such as Light Detection and Ranging (LiDAR) or multi-view stereo (MVS) brought the ability to record the world under the form of 3D point clouds. However, point clouds are usually not sufficient to model complex physical processes. Instead, a variety of applications in science and engineering require a representation of objects or scenes under the form of a continuous surface. For example, in medicine, the continuous surface of an organ allows for diagnosing and monitoring malformations [111]. In robotics, the surface of a robots environment enables path planning and collision detection [24]. In architecture, building and civil engineering, the surface of a city permits the computation of light, heat or noise propagation [14]; and in environmental engineering, a continuous surface of a terrain permits to model wind or floods [73] (see Figure 1.1). Digitally and continuously modeling an existing surface involves converting physical measurements into mathematical and digital models [1]. This process includes *surface reconstruction from point clouds*, one of the key scientific challenges in digital geometry processing.

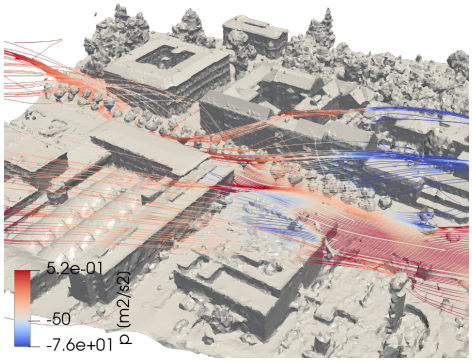
Surface reconstruction from point clouds addresses the problem of producing a continuous representation of a surface of which discrete point samples have been acquired. However, discrete point samples usually do not cover the entire surface geometry and do not contain topological information about the recorded surface (cf. Figure 1.2a and 1.2b). Theoretically, there are infinitely many surfaces that can pass through, or near the point samples. This means there is no unique solution to surface reconstruction from point clouds, making it an ill-posed problem. If no prior information is given, a reconstruction algorithm can only *approximate* the real surface between sampled points.



(a) Data acquisition

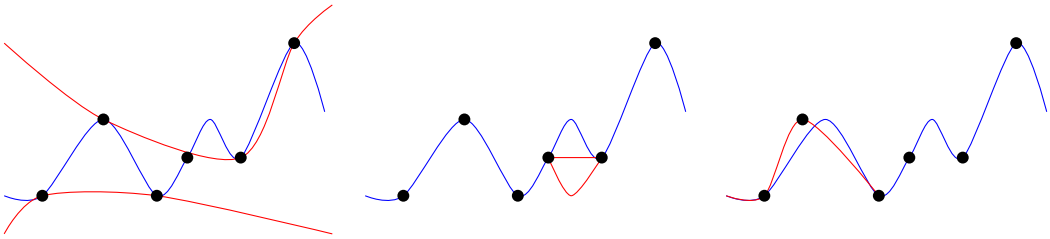


(b) Surface reconstruction



(c) Analysis

Figure 1.1: 3D urban analysis pipeline: 3D point clouds of urban environments are often acquired by aerial laser scanning or aerial imagery (a). The acquired point cloud can be used to generate a digital surface model using a surface reconstruction algorithm (b) — the central topic of this thesis. The surface model can then be used for advanced urban analysis, such as a numerical wind simulation (c). Here, this pipeline is exemplified on a part of the city of Strasbourg, France around the *Musée zoologique*.



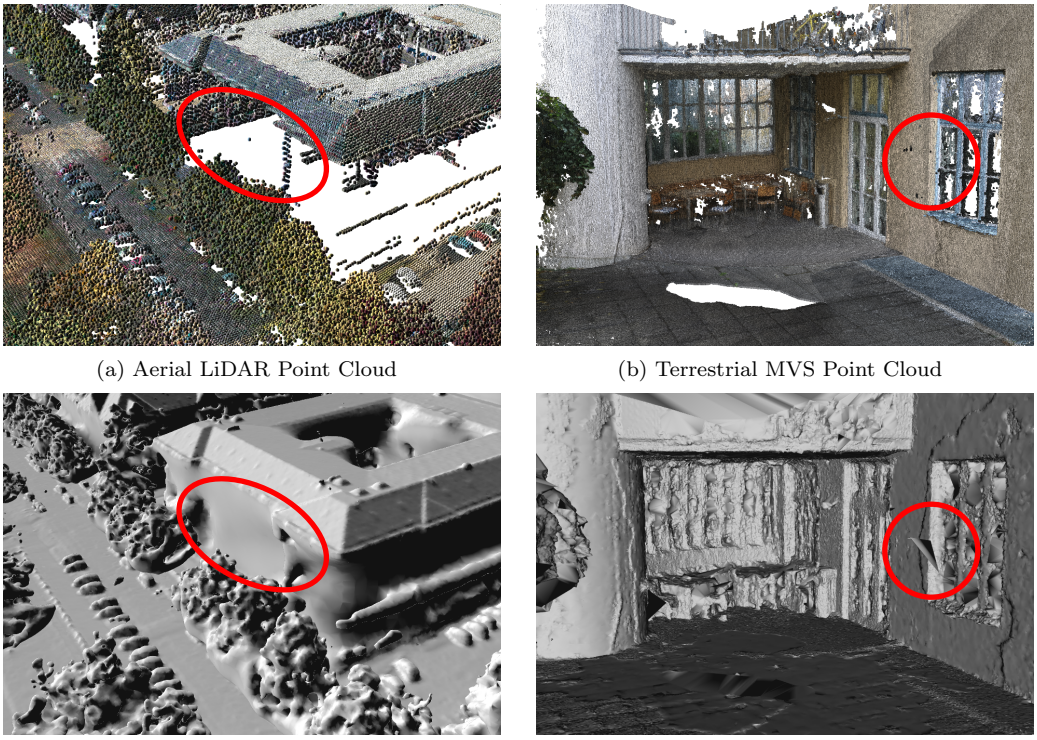
(a) Unknown Topology (b) Unknown Geometry (c) Acquisition Defects

Figure 1.2: Difficulties in surface reconstruction from point clouds: In each plot, we show the real surface —, point samples ●, and possible reconstructions —. The correct topology and geometry of the real surface are not known from the point samples (a,b). The point samples may also include acquisition defects such as noise (c). The goal of any surface reconstruction algorithm is finding a good approximation of the real surface, in terms of its geometry and topology. Learning based surface reconstruction can learn shape patterns or sampling errors such as the one exemplified here, and use the learned knowledge during reconstruction for a better approximation.

Therefore, the goal of surface reconstruction from point clouds is to find a good approximation of the real surface, in terms of its geometry and topology.

Another challenge in surface reconstruction from point clouds lies in the fact that the sampling of a real surface is often not error free (cf. Figure 1.2c). Low quality sensors or difficult acquisition conditions can result in point cloud defects, which complicate the reconstruction of a good surface approximation. For these reasons, surface reconstruction from point clouds has been a long standing problem in digital geometry processing. In this thesis, we revisit the problem of surface reconstruction from point clouds with modern learning-based techniques. We aim to develop deep learning architectures that can learn point cloud defects or surface patterns from a given training set, and use the learned knowledge during reconstruction for a better surface approximation. We thereby focus on surface reconstruction from point clouds in the wild. Point clouds in the wild are recorded in an uncontrolled environment outside of the laboratory. They can depict dynamic environments with multi-scale surfaces, ranging from single objects to entire countries and include defects such as noise, outliers, nonuniform sampling or missing data.

In this first chapter, we briefly introduce the problem of surface reconstruction from 3D point clouds in the wild, the concept of visibility information, and existing deep learning architectures for surface reconstruction. We then discuss the objectives and limitations of this work, as well as its contributions. Finally, we provide a reading guide for the rest of this document.



(a) Aerial LiDAR Point Cloud

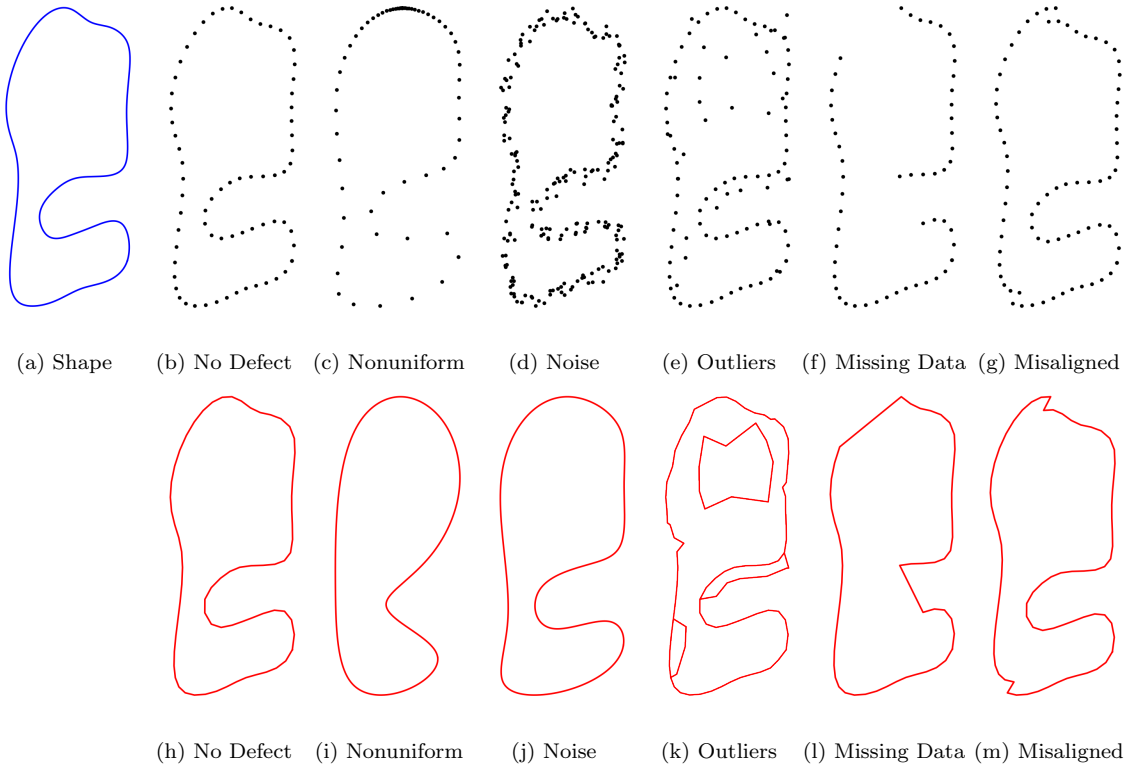
(b) Terrestrial MVS Point Cloud

(c) Aerial LiDAR Surface Reconstruction

(d) Terrestrial MVS Surface Reconstruction

Figure 1.3: Point clouds in the wild and reconstructed surfaces: Real world point clouds acquired under challenging conditions often have defects such as noise, outliers, non-uniform sampling or missing data (a, b). Surfaces reconstructed from this data often inherit some of these defects (c, d). In (a), we show an aerial LiDAR point cloud. Due to the angle of incident rays there are very few points on the façades of the buildings. As a result, the inferred surface in (c) is missing details such as windows on the façades. In (b) we show an MVS point cloud from terrestrial images. The complex shapes and low textured surfaces lead to noise, outliers and missing data in the point cloud, which, in turn, leads to erroneous and noisy surface parts in (d). The input point clouds are from *Strasbourg3D*[1] (a) and *ETH3D* [99] (b). The reconstructions are generated with the traditional Screened Poisson Surface Reconstruction [65] (c) and the method proposed in Vu *et al.* [114] (d).

[1] https://3d.strasbourg.eu/



(a) Shape (b) No Defect (c) Nonuniform (d) Noise (e) Outliers (f) Missing Data (g) Misaligned

(h) No Defect (i) Nonuniform (j) Noise (k) Outliers (l) Missing Data (m) Misaligned

Figure 1.4: Point cloud defects [9]: We show a 2D shape — and samples ●. The samplings include different forms of defects which affect the reconstructed surface — .

1.1 Surface reconstruction from 3D point clouds in the wild

The most commonly used methods for acquiring point clouds outside of the laboratory are active LiDAR and passive MVS (cf. Figure 1.3). LiDAR sensors or cameras for an MVS acquisition are used stationary or mounted on moving platforms such as cars, airplanes or satellites. The sensors itself range from low quality sensors, *e.g.* built into smartphones, to high quality sensors, mounted on dedicated surveying platforms. The diverse set of acquisition subjects and techniques leads to a variety of different properties and imperfections for point clouds in the wild. Most traditional approaches for surface reconstruction rely on handcrafted or data driven priors on the input point cloud or output surface to overcome acquisition defects. This approach allows to reconstruct surfaces sufficient for visualisation and a variety of other applications. However, when input point clouds include heavy defects, or when highly accurate surfaces are required, current surface reconstruction methods may not suffice [13].

1.1.1 Point cloud properties and imperfections

Throughout this document we consider $\mathcal{P} \in \mathbb{R}^{3\times P}$ a 3D point cloud defined by the absolute point positions in space, where P is the number of points p in the cloud.

Point cloud defects can be classified into the five different groups illustrated in Figure 1.4. We briefly discuss the five groups here and refer the reader to the survey of Berger *et al.* [9] for a more complete discussion.

Deficient and nonuniform sampling density. An important relation for surface reconstruction from point clouds can be established between the local feature size of a surface $\mathcal{S} \subset \mathbb{R}^3$ and the density of $\mathcal{P}$. The local feature size $\mathrm{LFS}(x)$ is defined as the minimal distance between a point $x \in \mathcal{S}$ and the medial axis of $\mathcal{S}$ [2]. A point cloud with point samples $p \in \mathcal{P}$ is called an ϵ-sampling of $\mathcal{S}$, if every point x has a point p in Euclidean distance at most $\epsilon\mathrm{LFS}(x)$ [28]. Some traditional surface reconstruction algorithms guarantee topological validity and geometric convergence between the reconstruction and $\mathcal{S}$, given that the input point cloud is noise free and has a small ϵ-value [2, 15, 28]. However, the local feature size can in general not be computed without having access to $\mathcal{S}$. Thus, an ϵ-sampling can not be guaranteed for point clouds in the wild. Uniformly sampling the surface with a high density leads to large redundant amounts of data, while uniformly sampling with a low density can lead to loss of information. One option for point cloud acquisition can thus be to define the minimal feature size that should be recovered and drive the acquisition process accordingly [1]. However, point clouds in the wild often non-uniformly sample the underlying surface, *e.g.* as a result of varying distance and orientation between surface and sensor [9]. Such a nonuniform sampling can pose problems for certain surface reconstruction methods, *e.g.* for defining fixed distances to determine local neighborhoods [9], or for partitioning the point cloud with regular grids.

Noise. LiDAR points can include noise in the form of offsets from the real surface, distributed along the ray coming from the sensor [96]. Points that are randomly distributed near the surface are common noise for MVS acquisitions. The noise is a result of mismatches during dense point cloud reconstruction or erroneous approximations of the camera position and orientation. This happens in particular for low texture surfaces, or in highly variable outdoor environments which include vegetation and clutter. The level of noise can also vary within one MVS point cloud. Estimating a model representing such kind of noise is often not possible [114]. Noisy MVS point clouds are thus challenging for surface reconstruction algorithms, especially if the level of noise is close to the local feature size. The challenge for surface reconstruction algorithms is to preserve small features in the surface while discarding or smoothing noise.

Outliers. Outliers are points randomly distributed in the acquisition space, not close to the surface. In MVS point clouds, single outlier points or groups of outliers are often the result of large mismatches or erroneous orientation between images. Furthermore, they occur in LiDAR acquisitions due to reflecting objects and glass [99]. Structured outliers pose a significant problem for surface reconstruction algorithms, as they are hard to filter and can produce ghost structures in the surface (see Figure 1.4k).

Missing data. The main reason for missing data in point clouds are (self-)occlusions. This happens when an object or surface part lies between subject and sensor, *i.e.* the subject is occluded. Even for surfaces with little complexity this is very common. Missing data is one of the main challenges for surface reconstruction algorithms, because it requires to reconstruct entire surface parts without local input information.

Misaligned scans. Large-scale scenes often require to split the acquisition into several smaller parts, *e.g.* for changing the sensor position during multiple stationary acquisitions. Subsequently, the scans have to be aligned to produce one point cloud depicting the entire scene. If the position and orientation between the scans is unknown, it has to be inferred using an alignment technique called registration [57]. This process may however not be error free and lead to misaligned scans.

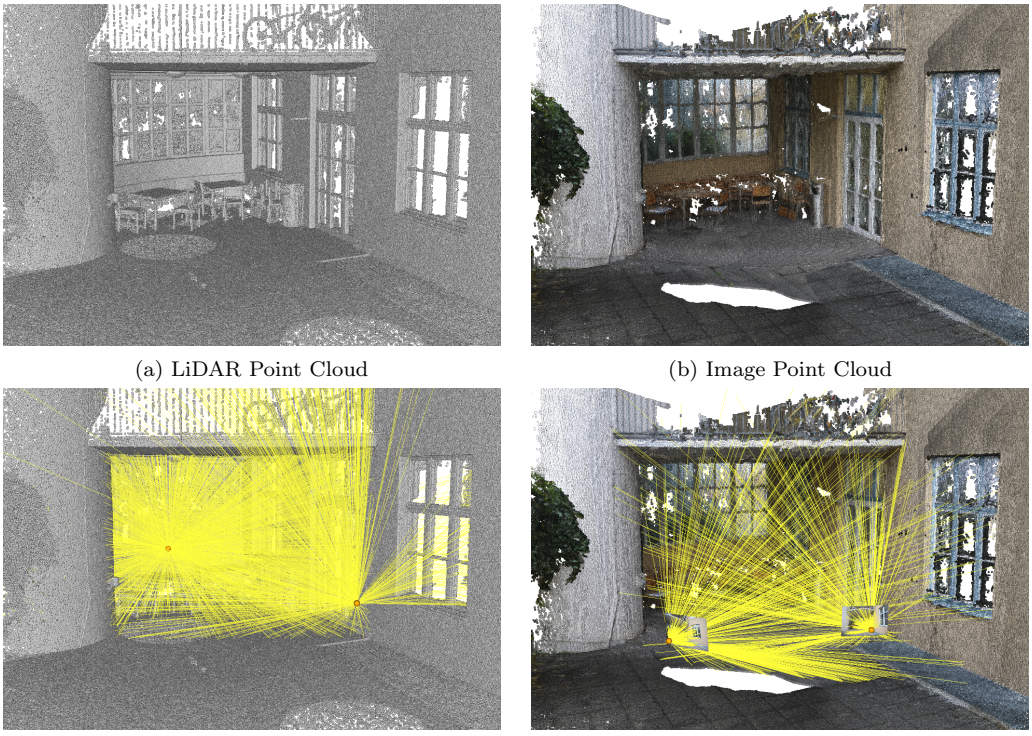
In general, LiDAR points clouds are accurate but their density varies with the acquisition distance and angle. Furthermore, occlusions are more common for some LiDAR sensors, due to a limited field of view. MVS point clouds, acquired in the wild, are usually less accurate, but provide denser information, with less missing data. One way to overcome some of the sensor limitations is to combine different acquisitions [25]. However, this can be more costly and time consuming.

1.1.2 Visibility information

Both LiDAR and image-based point cloud acquisitions allow to directly measure or recover the position of the sensors relative to $\mathcal{P}$. We consider $\mathcal{C} \in \mathbb{R}^{3\times C}$ the positions of a set of sensors, where C is the number of sensors c. C can very from a few (for stationary acquisitions) to hundreds of different positions (for moving sensors).

Given a point p and its corresponding sensor position c, we call the half open segment between a point and its sensor the line-of-sight $\overline{cp}$ (cf. Figure 1.5). It is half open, because it includes c but not p. Up to uncertainties in the acquisition, $\overline{cp}$ must lie outside the scanned subject, because it is the path of reflected light from surface point to sensor. We call this additional information *visibility information*.

While visibility information is key for point cloud reconstruction techniques such as structure from motion (SfM) and MVS, it is only explored by a small subset of surface



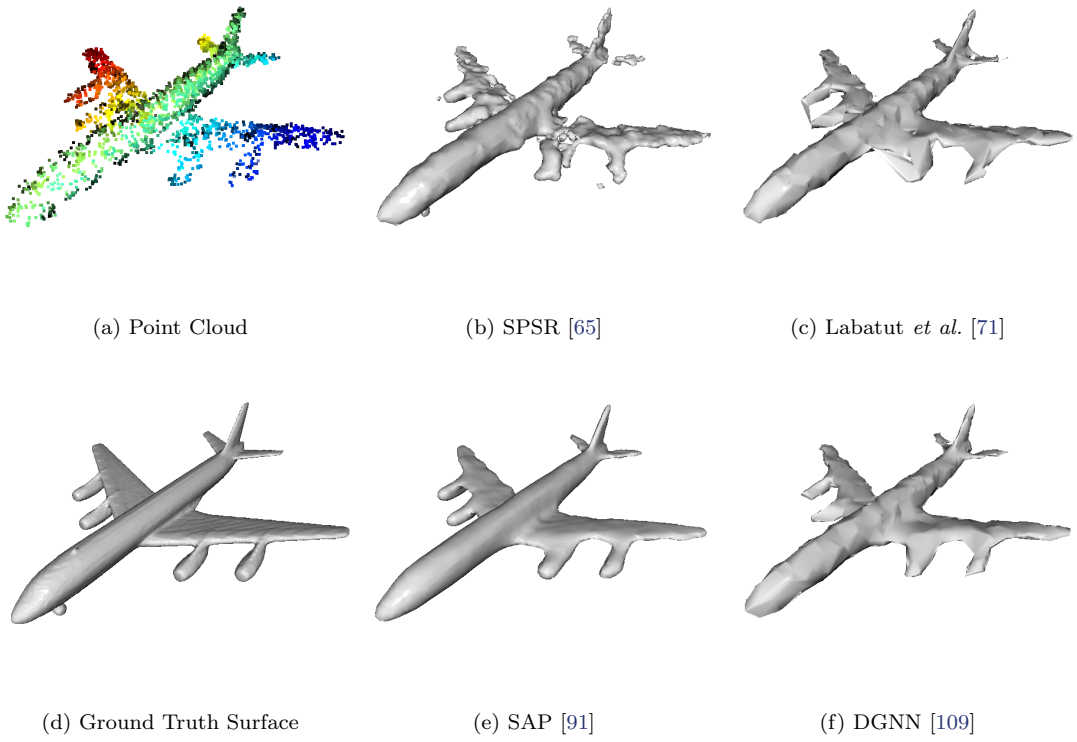
(a) LiDAR Point Cloud

(b) Image Point Cloud

(c) LiDAR Point Cloud With Visibility

(d) Image Point Cloud With Visibility

Figure 1.5: Point clouds with visibility information: Terrestrial point clouds (a, b) from the *terrace* scene of ETH3D [99]. We visualize some of the sensor positions ● and lines-of-sight —— (c, d). This so-called visibility information can help surface reconstruction algorithms in identifying free space and in correctly orienting the surface towards the sensor.



(a) Point Cloud (b) SPSR [65] (c) Labatut *et al.* [71]

(d) Ground Truth Surface (e) SAP [91] (f) DGNN [109]

Figure 1.6: Learning and non-learning based surface reconstruction: We show a (synthetic) point cloud (a), its corresponding ground truth surface (d) and corresponding non-learning (b,c) and learning based (e,f) surface reconstructions. The learning based methods are related to the non-learning methods, however, they have the ability to incorporate learned knowledge from a training set during reconstruction. In this case, the two learning methods were trained on a large shape database including similar (airplane) models as the one shown here. The training point clouds had the same density and level of noise as the one shown here, which makes the learning methods more robust to these defects.

reconstruction algorithms. However, as shown by one of the most widely used surface reconstruction algorithms [71], incorporating visibility information into the surface reconstruction problem can be beneficial by providing a prior that lowers the solution space, and helps in correctly orienting the surface.

1.1.3 Deep Learning for Surface Reconstruction

In recent years the deep learning community started to address the surface reconstruction problem [49, 79, 82]. Most deep surface reconstruction (DSR) approaches use su-

pervised learning to learn point cloud characteristics or shape patterns directly from a training set. However, supervised deep learning often requires large amounts of training data. To this end, DSR methods train on large synthetic shape dataset with continuous true surfaces, sampled to generate point clouds. For mimicking point cloud defects, *e.g.* noise is added to the point clouds following a simple Gaussian noise model. However, such trained models may not be useful to reconstruct unseen shapes from point clouds in the wild [86]. On the other hand, directly training DSR methods with point clouds in the wild is often not feasible, as gathering and modeling true surfaces for such point clouds is either too complex, costly or mathematically intractable. Furthermore, a variety of methods work on regular 3D grids, *i.e.* voxels, which are processed with large fully connected network (FCN) architectures. As a result, existing architectures are often too compute- and memory-intensive to process large-scale point clouds. Lastly, existing learning-based DSR algorithms only operate on point locations and ignore visibility information. Neural radiance fields (NeRFs) and other differentiable volumetric rendering techniques [84, 86, 87] use visibility information to reconstruct a surface directly from images. However, they often require a slow optimization process and leverage little or no shape priors.

1.2 Problem statement and objectives

In this thesis, we aim to overcome some of the main limitations of existing traditional and learning based surface reconstruction methods, namely:

- Traditional methods can not incorporate learned priors, *e.g.* to fill large parts of missing data or to filter and smoothen defects sufficiently
- Learning methods do not scale to large point clouds and their generalization capability to unseen point cloud defects and shapes has not yet been studied systematically
- Learning methods do not yet incorporate visibility information

We investigate the requirements of deep learning based architectures for surface reconstruction from point clouds in the wild. We aim to develop supervised deep learning architectures that can handle large-scale point clouds, while being trained on small synthetic datasets. The methods should be robust to common defects for point clouds in the wild and produce topologically valid and geometrically accurate surfaces. We consider point clouds with visibility information and aim to develop algorithms that can benefit by using this additional information.

1.2.1 Scope and limitations

A major advantage of deep learning methods is that they can be trained end-to-end for a specific task without the necessity of intermediate representations. For this reason we focus on surface reconstruction methods that take a defect-laden point cloud as input and export a continuous surface. We neither discuss pre-processing algorithms, such as point cleaning [96] or resampling methods [80], nor post-processing algorithms, such as mesh repair, surface smoothing, or sharp feature recovery. Ideally, we would like to directly export surfaces which can be visualised and are valid input for surface analysis algorithms. To this end, we focus on the reconstruction of a watertight, non-self intersecting surface, a useful requirement for a variety of downstream engineering applications. This entails that the discussed and developed algorithms aim to fill all wholes in the point set and provide a watertight reconstruction, even if there is no guarantee.

LiDAR and image based sensors are the two main sensors for acquiring point clouds in the wild. We study both, unprocessed LiDAR and MVS point clouds. However, albeit the end-to-end paradigm, we do not investigate methods that can directly use images in an end-to-end fashion to reconstruct surfaces. We rely on dedicated pipelines for this step. This has the benefit that our developed architectures can handle both LiDAR and image based point clouds without modification, and can even input a combination of both.

During the course of this thesis, a variety of surface reconstruction methods based on deep neural networks have emerged. We call all of these methods *deep surface reconstruction*. However, not all deep surface reconstruction methods incorporate shape priors learned from a training set. While we also discuss such non-learning (optimization based) methods in Chapter 2, they are not the main focus of our work.

Recent research also suggest to use end-to-end learning for directly performing surface analysis, such as fluid dynamics from scanned point clouds. This approach removes the need to explicitly compute a continuous surface as an intermediate representation. While an interesting field of research, it can lead to more computation. Each new analysis requires a new optimization from a raw point cloud instead of a richer information surface. Additionally, such methods do not allow surface visualisation. We do not cover these approaches.

1.3 Reading guide and contributions

We assume that the reader of this thesis is familiar with sensors and methods to acquire 3D point clouds such as LiDAR and MVS [72] and with basic principles of polygon mesh processing [18], computer vision [105] and deep learning [47]. In the digital PDF version of this document all abbreviations and acronyms are linked to the Glossary. In

turn, the Glossary links to the first occurrence of the abbreviation or acronym, where it is spelled out, explained and, if applicable, a citation is given.

In Chapter 2, we formally introduce the problem of surface reconstruction from point clouds. This chapter can been seen as an introduction for readers unfamiliar with the problem of surface reconstruction from point clouds and popular methods that address it. We summarise traditional test-of-time as well as recent learning-based surface reconstruction methods. We benchmark all methods on the same datasets to assess the influence of learning in surface reconstruction.

In Chapter 3, we augment existing deep surface reconstruction methods to enable the processing of point clouds with visibility information. We show that visibility information consistently improves the accuracy of generated surfaces as well as the generalization capability of the networks to unseen domains. The chapter is published in Sulzer *et al.* [108].

In Chapter 4, we present a novel deep surface reconstruction method that can process large-scale point clouds in the wild, and reconstruct surfaces including objects not present during training. The chapter is published in Sulzer *et al.* [109].

In Chapter 5, we summarise and conclude the thesis and provide an outlook into future work.

1.3.1 Open Source Software and Datasets

Several open source software packages published on GitHub are also a result of this thesis and provide means for reproducing the presented results.

MESH-TOOLS

mesh-tools[1] is a collection of tools for reconstructing and processing meshes. The programs are mainly written using CGAL[2] and are based on several scientific publications. Included is, *e.g.* our implementation of the surface reconstruction algorithm published by Labatut *et al.* [71], which is extensively used in this thesis; and methods for surface normal estimation and orientation.

DSR-BENCHMARK

dsr-benchmark[3] includes a collection of datasets and an evaluation pipeline for surface reconstruction algorithms introduced in Chapter 2.

[1] https://github.com/raphaelsulzer/mesh-tools
[2] https://www.cgal.org/
[3] https://github.com/raphaelsulzer/dsr-benchmark

DSRV-DATA

dsrv-data[4] is a collection of several deep surface reconstruction algorithms modified to be able to process point clouds with visibility information, as introduced in Chapter 3.

DGNN

dgnn[5] is a surface mesh reconstruction algorithm presented in Chapter 4 mainly written using pytorch[6] and pytorch-geometric[7].

[4] https://github.com/raphaelsulzer/dsrv-data
[5] https://github.com/raphaelsulzer/dgnn
[6] https://pytorch.org/
[7] https://pytorch-geometric.readthedocs.io/en/latest/

2

A Survey and Benchmark of Automatic Surface Reconstruction from Point Clouds

We survey and benchmark traditional and novel learning-based algorithms that address the problem of surface reconstruction from point clouds. Surface reconstruction from point clouds is particularly challenging when applied to real-world acquisitions, due to noise, outliers, non-uniform sampling and missing data. Traditionally, different handcrafted priors of the input points or the output surface have been proposed to make the problem more tractable. However, hyperparameter tuning for adjusting priors to different acquisition defects can be a tedious task. To this end, the deep learning community has recently addressed the surface reconstruction problem. In contrast to traditional approaches, deep surface reconstruction methods can learn priors directly from a training set of point clouds and corresponding true surfaces. In our survey, we detail how different handcrafted and learned priors affect the robustness of methods to defect-laden input and their capability to generate geometric and topologically accurate reconstructions. In our benchmark, we evaluate the reconstructions of several traditional and learning-based methods on the same grounds. We show that learning-based methods can generalize to unseen shape categories, but their training and test sets must share the same point cloud characteristics. We also provide the code and data to compete in our benchmark and to further stimulate learning-based surface reconstruction.

2.1 Introduction

Surface reconstruction from point clouds is a key step between acquisition and analysis of surface models and is a long-standing problem in digital geometry processing. In this chapter, we survey and benchmark several traditional and learning-based methods that address the problem of surface reconstruction from point clouds.

Traditionally, surface reconstruction methods made the problem more tractable by using handcrafted priors, imposed on the input, such as point density, level of noise or outliers, and on the output, such as smoothness, topological properties or the shape category. In contrast, recent methods introduced by the deep learning community can learn point cloud defects or shape patterns directly from training data and therefore promise to reconstruct more accurate surfaces without the need for manual parameter tuning. However, so far DSR methods have mostly been applied on datasets with a small number of different object categories. Such datasets are not representative for real-world applications, where algorithms have to reconstruct surfaces containing a large variety of shapes unseen during training.

Furthermore, DSR methods are often applied on uniformly sampled point clouds. Likewise, such point clouds are not representative for real-world acquisitions, as they do not model non-uniformity or missing data stemming *e.g.* from occlusions, or transparent and low texture areas. The ability to reconstruct shapes, either from unseen shape classes or from point clouds with unseen defects is rarely studied in a systematic manner for DSR methods.

To this end, we propose several experiments to benchmark algorithms for surface reconstruction from point clouds. We make use of a variety of publicly available shape datasets with object surfaces of different complexities. The objects are represented by a true surface $\mathcal{S}$, which is a boundary-free 2-manifold, *i.e.* each point on the surface has a neighborhood that is homeomorphic to an open subset of the Euclidean plane. We synthetically scan the objects to produce point clouds with real characteristics. Having access to the true surfaces allows us to measure the geometric and topological reconstruction quality of the benchmarked methods. We also verify our findings on real-world point clouds.

We compare novel learning-based algorithms to traditional test-of-time methods to specifically study the influence of learned priors incorporated into the surface reconstruction process. We thereby pay special attention to the generalization capability of methods to unseen domains. Our main contributions are as follows:

- We review methods for surface reconstruction from point clouds from over three decades up to recent learning-based methods. We contrast popular test-of-time with novel DSR methods.

- We benchmark traditional and learning-based methods on the same ground across

several experiments, using openly available shape datasets and point clouds generated with synthetic and real scanning.

2.2 Related work

2.2.1 Surveys

There exists only few works that survey the broad field of surface reconstruction from point clouds [9, 17, 28, 120], most of them predating the advance of learning-based surface reconstruction [9, 17, 28]. Surface reconstruction methods are often grouped into interpolating or approximating methods [18]. Interpolating methods "connect" all points of the input point cloud, or a subset thereof, usually by linearly interpolating between pairs of points. Approximating methods often define one or several smooth functions approximating the point cloud globally or locally. See Figure 2.1 for an illustration. Berger *et al.* [9] and Cazals & Giesen [28] provide detailed reviews for approximating and interpolating surface reconstruction methods, respectively.

To the best of our knowledge, only one survey includes learning-based methods [120]. However, this survey predates important developments for learning-based methods, such as the incorporation of local information [21, 31, 45, 62, 92, 95, 109]. In this work, we review both interpolating and approximating methods and focus on novel ideas in learning-based surface reconstruction. While many reconstruction methods can be distinguished by the prior assumptions they impose [9], we argue that a variety of successful methods combine different priors. This makes grouping by priors difficult. We thus organize methods into two groups: surface-based and volume-based approaches. This breakdown closely relates to the two main classes of mathematical representations of a surface: parametric and implicit.

2.2.2 Benchmarks

To date, benchmarks for surface reconstruction from point clouds are rare. Many methods use custom datasets to evaluate their approach, usually generated by uniformly sampling point clouds from ground truth shapes of existing shape collections [21, 31, 62, 89, 92, 95]. However, the characteristics of the sampled point clouds often differ across publications, which hampers the ability to fairly compare the results of different works. Furthermore, the point clouds often lack common defects of real acquisitions, such as missing data or outliers. One notable exception is the benchmark of Berger *et al.* [8]. The authors develop a synthetic range scanning procedure to produce scans with realistic artifacts, such as noise, non-uniformity and misaligned scans and create point clouds from shapes with non-trivial topology and details of various feature sizes. While providing interesting results, the benchmark predates learning-based surface reconstruction and only considers traditional approximating methods. In

the benchmarks proposed in this paper, we reuse their synthetic range scanning procedure and their five test shapes, as they provide realistic and challenging input for both learning-based and traditional algorithms. We also implement our own synthetic scanning procedure for MVS-like point clouds. We use the synthetic scanning to scan existing large shape datasets to create training datasets with true surfaces and point clouds with realistic characteristics.

A problem related to surface reconstruction is the generation of point clouds from 2D information such as overlapping images. There exists a variety of benchmarks using data captured in a laboratory environment [61, 100] or in the wild [68, 99, 107]. These benchmarks often use a low quality image acquisitions as reconstruction input. Simultaneously, a higher quality acquisition, *e.g.* from LiDAR scans, serves as reference.

One problem with this approach is that, even for high quality acquisition techniques, it is difficult to produce complete ground truth point clouds. This issue is sometimes addressed by decreasing the ground truth domain to specific evaluation areas, in which reliable information is available either from recorded points or sightlines between points and sensors [61, 99]. However, in contrast to true surfaces, reference point clouds, do not allow to calculate topologic metrics such as the number of components or differential metrics such as surface normals. Furthermore, most learning-based methods require closed reference surfaces instead of reference point clouds for training.

2.3 Surface definition, representations, properties and reconstruction

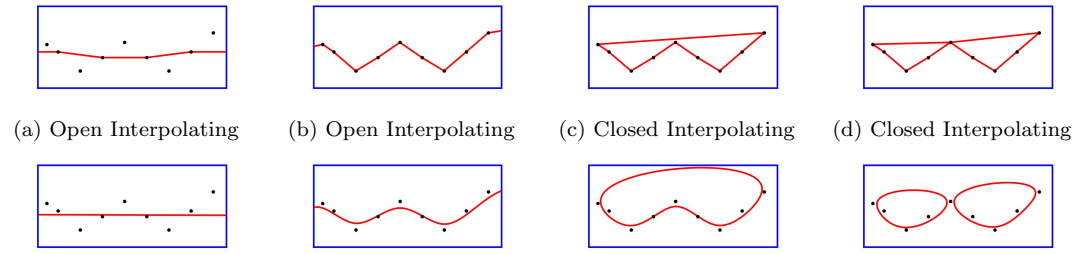
In this section, we first provide a definition of a surface and its mathematical and digital representations. We then discuss important surface properties. Finally, we establish the connection between mathematical surface representations, and the grouping of surface reconstruction algorithms used in our survey.

2.3.1 Definition

A surface can be defined as an orientable, continuous 2-manifold in $\mathbb{R}^3$, with or without boundaries [18, 54, 88]. These properties are important for surface visualisation and processing, and we will discuss them further down. Mathematically, there are two main classes of surface representations: *parametric* and *implicit*.

2.3.2 Representations

Parametric surfaces are defined by a function $\mathbf{f} : \Omega \mapsto \mathcal{S}$ that maps a parameter domain $\Omega \in \mathbb{R}^2$ to the surface $\mathcal{S} = \mathbf{f}(\Omega) \in \mathbb{R}^3$. However, for complex surfaces it is not feasible to find a single function that can parameterise $\mathcal{S}$. Therefore, the parameter domain Ω is usually split into sub-regions for which an individual function is defined [18]. The most



(a) Open Interpolating (b) Open Interpolating (c) Closed Interpolating (d) Closed Interpolating

(e) Open Approximating (f) Open Approximating (g) Closed Approximating (h) Closed Approximating

Figure 2.1: Approximating and interpolating surfaces from point clouds: A surface generated from point samples can either interpolate (top row) or approximate (bottom row) the samples. Theoretically, there exist an infinite number of surfaces with different geometry and topology that can pass through, or near, the samples. We show eight different surfaces — reconstructed from the same point cloud • • • • in (a) - (h). The point cloud can be seen as a sampling of a part of a real surface. All reconstructed surfaces are watertight, as they are either closed and boundary-free, or their only boundary is the intersection with the domain boundary □. The surface in (d) is non-manifold in the center-vertex. All other surfaces are manifold. Except for (h), all surfaces are comprised of only one component. In contrast to the point cloud depicted here, in our benchmark, we mainly consider point clouds sampled from closed surfaces.

common way is to segment Ω into triangles, which are planar by definition. A set of triangles approximating $\mathcal{S}$ can be efficiently stored and processed as a triangle surface mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, with triangle facets $\mathcal{F}$, edges $\mathcal{E}$ and vertices $\mathcal{V}$.

Implicit surfaces are defined by the level-set c of a scalar valued function $F : \mathbb{R}^3 \mapsto \mathbb{R}$:

$$\mathcal{S}_c = \{\mathbf{x} \in \mathbb{R}^3 \mid F(\mathbf{x}) = c\}. \tag{2.1}$$

The most common choice of the implicit function F is either a signed distance or an occupancy function. A signed distance function (SDF) gives the distance from a 3D point $\mathbf{x}$ in space to the surface; with points in the interior signed a negative value, and points on the exterior signed a positive value. An indicator or occupancy function (OF) usually has a value of 1 inside the surface and 0 outside. The c-level-set of F then yields the surface $\mathcal{S}$, where $c = 0$ in the case of a signed distance function and $c = 0.5$ in the case of an occupancy function. Similar to the parametric case, the implicit function domain is often split into sub-regions, such as voxels, octree-nodes or tetrahedra, and constant functions are defined in each sub-region.

2.3.3 Properties

The reconstructed surface $\mathcal{S}^r$ should be close in terms of geometry and topology to the real surface $\mathcal{S}$ from which the point cloud $\mathcal{P}$ is sampled. To facilitate subsequent geometric operations on $\mathcal{S}^r$, such as sampling or deforming the surface, a mesh reconstruction $\mathcal{M}$ is also desirable. $\mathcal{S}^r$ and $\mathcal{M}$, respectively, should have the following properties (see Figure 2.1 for illustrations):

- **Watertight:** A geometric surface is closed if it is boundary-free. A mesh $\mathcal{M}$ is closed—or boundary-free—if no edge is incident to exactly one facet. However, a reconstructed surface of a real scene necessarily has a border defined *e.g.* by the limit of the scan coverage. One may still reconstruct a closed surface by intersecting it with the boundary of the domain in which $\mathbf{f}$ or F is defined: *e.g.* the convex hull or bounding box of $\mathcal{P}$. However, this procedure may not be desirable, as it can hinder simple geometric analysis such as the calculation of surface area. Instead, we define a surface as watertight if it is boundary free, *except* for a possible intersection with the domain boundary.

- **Manifold:** We consider real and geometric surfaces to be 2-manifolds, *i.e.* each point on the surface has a neighborhood that is homeomorphic to an open subset of the Euclidean plane. A mesh $\mathcal{M}$ is manifold if it is edge- and vertex-manifold, and intersection-free.

  - *Edge-manifold:* For each edge $\mathcal{E}$, the set of facets $\mathcal{F}$ sharing this edge form a topological (half-)disk. This means that no edge can be incident to more than two facets.

  - *Vertex-manifold:* For each vertex $\mathcal{V}$, the set of facets sharing this vertex form a topological (half-)disk. This means that facets with a common vertex form an open or closed fan, *i.e.* there are no dangling facets.

- **Intersection-free:** $\mathcal{M}$ is intersection free if all pairs of facets not sharing an edge or vertex do not intersect.

- **Orientable:** $\mathcal{M}$ is orientable if one can define a consistent continuous orientation of each facet. This means that the order of the vertices of all facets is either clockwise or counter-clockwise and a common edge of two adjacent facets has opposite orders on the two sides.

The watertight property is useful for simulations such as fluid dynamics. Manifoldness and orientability are often required for mesh storing and processing, in particular because they are a prerequisite for the widely-used half-edge data structure [66, 78]. Furthermore, intersection-free and orientable surfaces lead to a well-defined notion of inside and outside, which is important for mesh visualization and a variety of geometric opertations.

2.3.4 Reconstruction

Surface reconstruction from point clouds is the process of constructing a continuous surface of which discrete point samples have been acquired. In our survey, we group methods for surface reconstruction from point clouds into two groups: surface- and volume-based. *Surface-based* reconstruction methods consists in finding (a set of) parameterised surfaces $\mathcal{S}^r$ that approximate the point cloud $\mathcal{P}$, either in the form of triangles or larger two-dimensional (2D) patches, or by deforming parameterised enclosing envelops such as meshed spheres. The main challenge for surface-based methods using a single function $\mathbf{f}$ is that the topology of Ω has to be equivalent to the topology of S, which is usually unknown. The main challenge for surface-based methods with individual functions for sub-regions of S, on the other hand, is to guarantee a consistent transition between each region. Hence, these methods often struggle to produce an intersection-free, manifold and watertight surface.

Volume-based methods, on the other hand, segment a subset of $\mathbb{R}^3$ into interior (inside) and exterior (outside) subspaces. The surface is implicitly defined as the interface between the two subspaces. Most, but not all algorithms in this class formulate the problem as finding an implicit function. Surfaces from volume-based methods are guaranteed to be watertight and intersection-free, but not necessarily manifold [28].

While surface-based methods can directly yield a mesh, *e.g.* by triangulating Ω, volume-based methods usually require an additional processing step. If the implicit field is discretized with tetrahedra, one can simply use a process which is sometimes called triangle-from-tetrahedra (TFT). TFT builds a triangle mesh from all triangles

that are adjacent to one inside- and one outside-tetrahedra. Another option is the algorithm of Boissonnat and Oudot [16] that iteratively samples F along lines from inside to outside to find points that lie on S and builds a triangle mesh from these points. One of the most popular methods for mesh extraction from an implicit field is Marching Cubes [76], which (i) discretizes the implicit function into voxels, (ii) constructs triangles inside each voxel that have at least one inside and one outside vertex and (iii) extracts a triangulation as the union of all triangles. Recently, mesh extraction has also been addressed by the deep learning community. Neural meshing [113] specifically addresses the case where an implicit function is represented by a neural network, and aims to extract meshes with fewer triangles compared to Marching Cubes from such a function.

In both, surface- and volume-based groups, there are methods that come with theoretical guarantees about the topology and geometry of the reconstruction in the absence of noise and when the point sampling is dense enough [28]. However, in this paper, we are mostly interested in the robustness of methods to defect-laden input point clouds from 3D scanning.

2.4 Survey

In this section, we review important surface- and volume-based surface reconstruction methods and discuss their robustness against different point cloud defects. We also show that learning-based approaches are often related to more traditional methods.

2.4.1 Surface-based reconstruction

Interpolating approaches

Advancing-front techniques. Most traditional surface-based approaches linearly interpolate between the point samples $\mathcal{P}$, or a subset thereof. This can be done efficiently by triangulating triplets of points which respect the empty ball property *i.e.* no other point lies within their circumsphere. Triangulating all triplets of $\mathcal{P}$ that have this property leads to the 3D Delaunay tetrahedralisation (3DT) of $\mathcal{P}$. The Ball Pivoting algorithm [11] is a greedy approach to find local triplets of points that form a triangle which is part of the surface. The first step is to (i) define a ball with constant radius, related to the density of $\mathcal{P}$ and to (ii) select a seed triplet of points. The ball must touch all three points and have no other point in its interior. The points then form the first surface triangle. Then, (iii) the ball pivots around an edge of the triangle until it touches a new point, forming a new surface triangle. Once all possible edges have been processed the algorithm starts with a (iv) new seed triangle until all points of $\mathcal{P}$ have been considered. The algorithm has later been refined to be more robust to non-uniform sampling [44, 93]. The Ball Pivoting algorithm and its related variations are often called

Table 2.1: Overview of surface- and volume-based surface reconstruction methods: We show an overview of surface- and volume-based surface reconstruction methods, both non-learning and learning-based, together with their input requirements (*normals*, *sensor pose*) and output type (*triangle mesh* or *implicit field*). Attributes denoted in brackets are optional. Methods with a *local* receptive field divide the point cloud into smaller sub-regions and define individual functions or surface patches for each sub-region. Methods with a *global* receptive field consider the entire point cloud at once. Methods denoted with *both* combine local and global receptive fields. We test methods in **bold** in our benchmark.

Method		learning	normals	sensor pose	receptive field	output
Surface-based						
BPA	[11]				local	triangle mesh
Sharf *et al.*	[101]				both	triangle mesh
AtlasNet	[49]	✓			local	triangle mesh
IER	[74]	✓			both	triangle mesh
PointTriNet	[102]	✓			local	triangle mesh
DSE	[95]	✓			local	triangle mesh
P2M	[52]				both	triangle mesh
Volume-based						
SPSR	[65]		✓		both	implicit field
Labatut *et al.*	[71]			✓	global	triangle mesh
ONet	[79]	✓			global	implicit field
DeepSDF	[89]	✓			global	implicit field
IM-Net	[35]	✓			global	implicit field
ConvONet	[92]	✓			both	implicit field
IGR	[48]	(✓)	(✓)		global	implicit field
LIG	[62]	✓	✓		local	implicit field
DGNN	[109]	✓		✓	both	triangle mesh
SAP	[91]	✓			both	implicit field
P2S	[45]	✓			both	implicit field
SAP	[91]	(✓)			both	implicit field
POCO	[21]	✓	(✓)		local	implicit field

advancing-front techniques. Their main drawback is that they are not robust to point cloud defects such as noise or point clouds with large missing parts.

Selection-based Similar to advancing-front techniques, the idea to iteratively build the triangulation from initial candidate triangles has also been explored in learning-based methods [74, 102]. PointTriNet [102] (i) starts with an initial set of seed triangles from a k-nearest neighbor graph of $\mathcal{P}$. Then, (ii) a first network takes in neighboring points and triangles of each seed triangle, and estimates its probability to be part of the surface. (iii) Triangles with high probability are selected to be part of the final surface and (iv) a second network proposes new candidate triangles constructed from two points of already selected surface triangles and neighboring points. The proposed new candidates are, again, processed by the first network and the algorithm continues for n user-defined iterations. The loss function is based on Chamfer distance between input points and the reconstructed surface, which allows the method to be trained without the need for ground truth meshes. IER-meshing [74] also (i) starts with a large set of seed triangles from a k-nearest neighbor graph. It then defines a so-called intrinsic-extrinsic ratio (IER), as the quotient of geodesic and Euclidean distance between points of a triangle. (ii) This ratio is estimated by an multilayer perceptron (MLP) from learned point features per triangle and supervised with IER's from a ground truth mesh. (iii) Only triangles with an IER close to 1 (*i.e.* Euclidean distance $\approx$ geodesic distance) are considered to be part of the surface and (iv) selected based on handcrafted heuristics. Both aforementioned methods have shown to be robust against small amounts of noise in the input point cloud. However, their reconstructed surfaces are neither manifold nor watertight.

Tangent plane and other projection methods Another class of surface-based interpolating approaches are tangent plane methods. This class includes the algorithm of Boissonnat [15], which is according to Cazals and Giesen [28] probably the first algorithm to address the surface reconstruction problem. The basic idea is to (i) find a tangent plane for each sample point, (ii) project the points local neighborhood on the tangent plane, (iii) construct 2D Delaunay triangulations of the projected points and (iv) merge the local reconstructions. A shortcoming of such an approach is that tangent planes are difficult to use in areas with high curvature or thin structures [95]. To this end, the idea of using local 2D Delaunay triangulations of projected points has been refined in a recent learning-based approach [95]. Instead of tangent planes, DSE-meshing [95] uses *logarithmic maps*, local surface parametrizations around a point p, based on geodesics emanating from p. This method (i) classifies geodesic neighbors of each point in $\mathcal{P}$ from a set of k-nearest neighbors. Then, (ii) an MLP approximates a logarithmic map parametrization to gain a 2D embedding of the geodesic neighbors. Lastly, (iii) neighboring logarithmic maps are mutually aligned and triangulated. This step allows

the method to reconstruct surfaces with fewer non-manifold edges, compared to methods that process triangles independently. However, the surface is still not watertight and the method has not been tested for reconstruction from noisy point clouds.

Patch-fitting

Patch-fitting methods are related to tangent plane approaches. Instead of interpolating the initial point set, a new triangulation patch is formed. AtlasNet [49] is based on this idea and was one of the first learning-based surface reconstruction methods. Small 2D triangulated patches are transformed to fit $\mathcal{P}$ based on transformations predicted by an MLP. Similar to interpolating approaches, this method cannot guarantee to fill all gaps between patches, which results in a non-watertight and potentially self-intersecting surface.

Surface deformation

One of the only classes of surface-based approaches that can guarantee a watertight surface are deformation-based methods. Sharf *et al.* [101] introduced a method that (i) iteratively expands an intial mesh contained within the input point cloud along the face normal directions, and (ii) moves the mesh vertices to fit the input point cloud using moving least squares. The method is shown to be robust against missing data, but requires careful parameter tuning to be robust against noise or outliers. Point2Mesh (P2M) [52] is also based on the aforementioned idea, but avoids the need for tuning parameters by hand. The method takes as input a convex hull or a low resolution Poisson reconstruction [65] of $\mathcal{P}$, and shrink-wraps this initial surface to best fit the point cloud. The process is guided by multiple local convolutional neural networks (CNNs) that share weights. The idea is that the weight sharing between the CNNs acts as a prior that identifies symmetric features in the shape while being able to ignore unsystematic, random defects in the point cloud. One problem with this approach is that the topology of the initial surface stays constant during reconstruction. If the correct topology of the surface is not known, it cannot be recovered. For example, if the sought surface has holes, they cannot be reconstructed from a convex hull initialisation. This poses a limitation for reconstructing arbitrary objects in the wild.

2.4.2 Volume-based reconstruction

Interpolating approaches

Volume-based interpolating approaches commonly start by constructing a 3DT of $\mathcal{P}$. In $\mathbb{R}^3$ a Delaunay triangulation (or tetrahedralization) subdivides the convex hull of $\mathcal{P}$ with tetrahedra. The 3DT is created in such a way that no point of $\mathcal{P}$ is contained in the

circumspheres of any tetrahedra. For well distributed point clouds it can be constructed in $O(n \log n)$ [3]. The Delaunay triangulation does not directly generate the surface, as it connects points in any direction. However, if the sampling $\mathcal{P}$ of $\mathcal{S}$ is dense enough a subcomplex of the 3DT is guaranteed to include a surface $\mathcal{S}^r$ closely approximating the geometry and topology of $\mathcal{S}$ [28]. One of the simplest ways to recover this subcomplex from a 3DT is to (i) prune all tetrahedra with circumspheres larger than a user specified constant radius α and then (ii) keeping only the boundary triangles. This leads to a so-called α-shape [10]. Similar to the Ball Pivoting algorithm the radius of the ball (here α) depends on the point density. For error free and dense samplings, alpha-shapes and some other interpolation methods [2, 15, 28] provide provable guarantees that the reconstructed surface is topologically correct [28]. Another way to recover a surface from a 3DT is inside-outside labelling [26, 53, 59, 60, 69, 71, 85, 104, 109, 109, 114, 122]. Here, all tetrahedra of a 3DT of $\mathcal{P}$ are (i) labelled as either *inside* or *outside* with respect to $\mathcal{S}^r$, and (ii) the surface is defined as the interface between tetrahedra with different labels. This guarantees to produce intersecting-free and watertight surfaces. The inside-outside labelling is usually implemented through a global energy minimized with graph-cuts. Inside-outside potentials are computed using visibility information and spatial regularization is achieved through surface smoothness or low area priors in the energy. This approach has been shown to be robust against most kinds of acquisition defects of moderate levels [59, 71, 114] and is capable of reconstructing (very) large scale scenes [85]. Delaunay-Graph Neural Network (DGNN) [109] is a learning-based method that replaces the handcrafted potentials in the aforementioned energy with a graph neural network (GNN). The GNN takes local geometric attributes and visibility information as input and operates locally on small subgraphs of the 3DT. The locality makes the method scale to large scenes. The method of Luo *et al.* [77] proceeds similarly, but without the use of visibility information and a global energy formulation. Instead, the GNN processes the 3DT of entire objects at once, which can hamper scalability.

Implicit functions

Arguably the largest class of surface reconstruction algorithms represent the surface with an implicit function (cf. Equation 2.1). One of the first methods that used implicit functions for surface reconstruction was presented in Hoppe *et al.* [54]. Hoppe *et al.* (i) calculate tangent planes at each input point of $\mathcal{P}$, using principal component analysis (PCA) of the local neighborhood. They then (ii) approximate an SDF by mapping an arbitrary point $x \in \mathbb{R}^3$ to its signed distance to the closest tangent plane. (iii) The surface is defined as the 0-level-set of the SDF. The local tangent plane estimation makes the process sensitive to low density sampling and noise, and computationally expensive.

Poisson surface reconstruction. The most popular approach for surface reconstruction based on implicit functions is Poisson Surface Reconstruction (PSR) [63]. The

idea is that the Laplacian of an indicator function χ, whose c-level-set approximates the unknown surface $\mathcal{S}$, should equate the divergence of a vector field $\vec{N}$ associated with $\mathcal{P}$:

$$\Delta\chi = \nabla \cdot \vec{N} \ . \tag{2.2}$$

The vector field $\vec{N}$ is defined by the oriented normals of $\mathcal{P}$. To define χ the algorithm (i) builds an octree on $\mathcal{P}$ and (ii) sets up a system of hierarchical functions, locally supported in each octree node, and (iii) globally solved by using a sparse linear system, which makes the method time and memory efficient. Dirichlet conditions can be imposed on the bounding box of the surface with $\chi = 0$ to ensure that the surface is closed. The approach is known to inherently produce smooth surfaces, but also over-smooth the surface in parts. The later introduced Screened Poisson Surface Reconstruction (SPSR) [65] can reconstruct much sharper surfaces by constraining Equation 2.2 to pass through $\mathcal{P}$. Additionally, it introduces the choice of Neumann boundary conditions which allows the surface to intersect the boundary of the domain in which F is defined. This is useful for open scene reconstruction. Recently the method has been revisited again, to impose Dirichlet constraints on a tight envelope around $\mathcal{P}$, enabling better reconstructions in areas of missing data [64]. Poisson surface reconstruction produces watertight meshes and has shown to be robust against almost all kinds of acquisition defects of moderate levels. However, all Poisson-based approaches require well oriented normals as input, which can pose a significant limitation in practice.

Neural implicit functions The most common approach to surface reconstruction with deep networks is to model F in Equation 2.1 with a neural network. This was first done in the pioneering works of Mescheder *et al.* [79], Park *et al.* [89], and Chen & Zhang [35].

In the case of Occupancy Networks (ONet) [79], F is modelled with a simple FCN architecture. The network takes as input a point cloud $\mathcal{P}$ and one or several test points $\mathbf{x}$ and outputs the occupancy of the test points in relation to the surface from which $\mathcal{P}$ was sampled. The conditioning on the input point cloud slightly changes the formulation of Equation 2.1 to:

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \mid F_\theta(\mathbf{x}, \mathcal{P}) = c\} \ . \tag{2.3}$$

To estimate the network weights θ, the network is trained with batches $\mathcal{B}$ of K objects using a simple binary cross entropy (BCE) loss:

$$\mathcal{L}_\mathcal{B}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \sum_{j=1}^{K} \mathrm{BCE}\left(F_\theta\left(\mathbf{x}_{ij}, \mathcal{P}_i\right), o_{ij}\right) \ , \tag{2.4}$$

where o_{ij} is the ground truth occupancy of test point $\mathbf{x}_{ij}$. To compute the ground truth occupancy o_{ij}, the training objects have to be available in the form of watertight surfaces. A common approach is to use large shape collections, such as ShapeNet [32] for training. Similar ideas have been introduced in IM-Net [35] and DeepSDF [89] to model an occupancy or signed distance function with a neural network. Instead of an encoder-decoder architecture as in ONet, the authors of DeepSDF [89] introduce an auto-decoder which is trained to find a shape code z that best explains an objects shape. This slightly changes Equation 2.3 and Equation 2.4, where the point cloud input $\mathcal{P}$ is replaced by a shape code z in the form of a 256-dimensional vector. The DeepSDF architecture then allows to reconstruct a complete signed distance field (and thus the shape), given a shape code z. However, to find the shape code for a specific shape during inference, at least a few ground truth signed distance values are necessary. This can be a significant limitation in practice. A common downside of the first DSR networks based on neural implicit fields is their simple fully connected network architecture. This architecture does not allow the incorporation of local point cloud information [92] and often leads to oversmoothing or inaccuracies of the inferred surface.

To this end, occupancy networks have later been refined by prepending 2D or 3D U-Nets [39, 97] before the fully connected occupancy network, to better incorporate local information. The idea is to (i) extract point features from local neighborhoods and (ii) aggregate these features in 2D or 3D grid cells. The U-Nets are then used to (iii) integrate local and global information using multiple down- and upsamplings. (iv) Finally, the fully connected ONet is used to compute test point occupancies. The approach is called Convolutional Occupancy Networks (ConvONet) [92]. Just as for the fully connected architectures, the network can be trained with test points $\mathbf{x}$ with known occupancy values o. In the same work, the authors also introduce an overlapping sliding-window approach in which a single trained ConvONet can be used to reconstruct entire indoor scenes. However, this approach requires to carefully scale the scene, such that the sliding window captures parts of the scene with comparable surface features during training and inference. Furthermore, for large-scale scenes, a sliding-window approach can be very time-consuming.

Local Implicit Grids (LIG) and DeepLS [31] also split input point clouds into overlapping subregions, and treat each subregion separately. The methods infer local shape codes z for parts of objects or scenes. These local shape codes have the additional benefit that they can represent parts from several different object classes. For example, a flat part-surface may belong to a table top or to a TV screen. This makes the methods less prone to overfit on specific shape categories used during training. However, the methods are largely based on IM-Net and DeepSDF. This means they also require a sort of ground truth test point during inference to optimize for the shape codes. Additionally, similar to the sliding window method of ConvONet, the region size (*i.e.* part size) has to be tuned.

Using the same encoder architecture as ConvONet, Shape As Points (SAP) [91] intro-

duces the combination of neural implicit fields with a differentiable Poisson solver. The method estimates (i) oriented normals as well as k point offsets for each input point, to correct and densify the point cloud $\mathcal{P}$. (ii) The resulting point cloud of size $k|\mathcal{P}|$ is fed to a differentiable Poisson solver [65] that computes an indicator grid, *i.e.* $\hat{\chi}$ evaluated on all nodes of a regular voxel grid. (iii) This indicator grid is supervised with a ground truth indicator grid χ. The ground truth indicator grid is created prior to training, from a Poisson reconstruction of a dense and error free point cloud, sampled from a ground truth mesh. A simple mean square error (MSE) loss is used for training the network:

$$\mathcal{L} = |\hat{\chi} - \chi|^2 \tag{2.5}$$

The entire pipeline is differentiable which allows to update point offsets, oriented normals and the network parameters during training (with batches of shapes). During inference, the computed indicator grid can simply be converted to a mesh using marching cubes. In contrast to the original Poisson Surface Reconstruction, SAP allows to incorporate learned priors and does not need $\mathcal{P}$ to be equipped with oriented normals.

In general, all of the methods based on voxel grids in this paragraph require the size of the initial voxels to be constant during training, because the resolution of the convolution layers depends on the voxel grid. This poses problems for training on point clouds with different densities. A dense voxel grid can be memory intensive and long to train, while a coarse voxel grid can oversmooth the input and lead to loss of information.

Another way to combine local and global information, that avoids the use of grids was introduced in Points2Surf (P2S). P2S uses both a local test point neighborhood sampling, and a global point cloud sampling which are both processed using MLPs and combined to predicted a signed distance for the test point. The k-nearest neighbor sampling makes this method less sensitive to point density, at the cost of increasing computational complexity, since the local neighborhood sampling has to be performed for each test point during inference.

Point Convolution for Surface Reconstruction (POCO) only relies on local neighborhoods and computes a latent vector per point using a point convolution backbone. The occupancy of a test point x is then predicted using attention-based weighing of neighboring latent vectors. This approach can focus the parameters of the learned implicit function to be used close to the surface. However, it also requires neighborhood sampling during inference. Similar to most other DSR methods, POCO is trained on object point clouds with a fixed number of points for easy mini-batching. However, to make the method more robust to point clouds with higher density during inference, the authors use a procedure called test-time augmentation. During inference, the latent vectors of each input point p are computed several times, from different local subsamples and then averaged.

Another approach to use neural implicit surface representations is to "train" (or optimize) the weights of a deep neural network per shape [48, 91]. The idea is to leverage

inherent symmetries of deep neural networks to act as priors in the reconstruction process, similar to the surface deformation based Point2Mesh discussed above. To this end, Gropp *et al.* [48] designed a simple fully connected network representing a signed distance function. To encourage the reconstruction of a smooth 0-level-set, given an input point cloud $\mathcal{P}$, they design a loss function which (i) should vanish on $\mathcal{P}$ and (ii) which gradients $\Delta_{\mathcal{P}} F$ should be of unit 2-norm and similar to the normals of $\mathcal{P}$. The method is called Implicit Geometric Regularisation (IGR). SAP also has an optimization-based variant where (i) the indicator grid, computed with the differential Poisson solver from the input point cloud $\mathcal{P}$ is used to compute a mesh. (ii) The mesh is then sampled, which allows to calculate a Chamfer loss between the sampled and input point cloud and, again, update the network weights, point offsets and oriented normals. (iii) This process is repeated until a user defined stopping criterion. The optimization-based variants of SAP and IGR can be trained per shape, without the need for ground truth meshes for supervision. However, in this optimization-based setting, they cannot learn and incorporate shape priors from a training set.

An upside of all DSR methods based on neural implicit representations is that they can store an implicit function, potentially conditioned on a point cloud, in the weights of a neural network. Especially DSR architectures that are entirely grid-less can directly relate their degrees of freedom to represent the surface. This can be more flexible compared to voxel, octree, or tetrahedral representations. Being a relatively new discovery, the full potential of neural network-based surface representations has probably yet to be explored.

2.5 Benchmark setup

In this section, we describe our set up of a series of experiments for benchmarking several surface reconstruction algorithms discussed in the previous section. We first describe how we generate realistic point clouds by using synthetic range and MVS scanning procedures. We then describe the datasets we used and several experiments to evaluate the performance of reconstruction methods. Finally, we provide an overview of the competing methods.

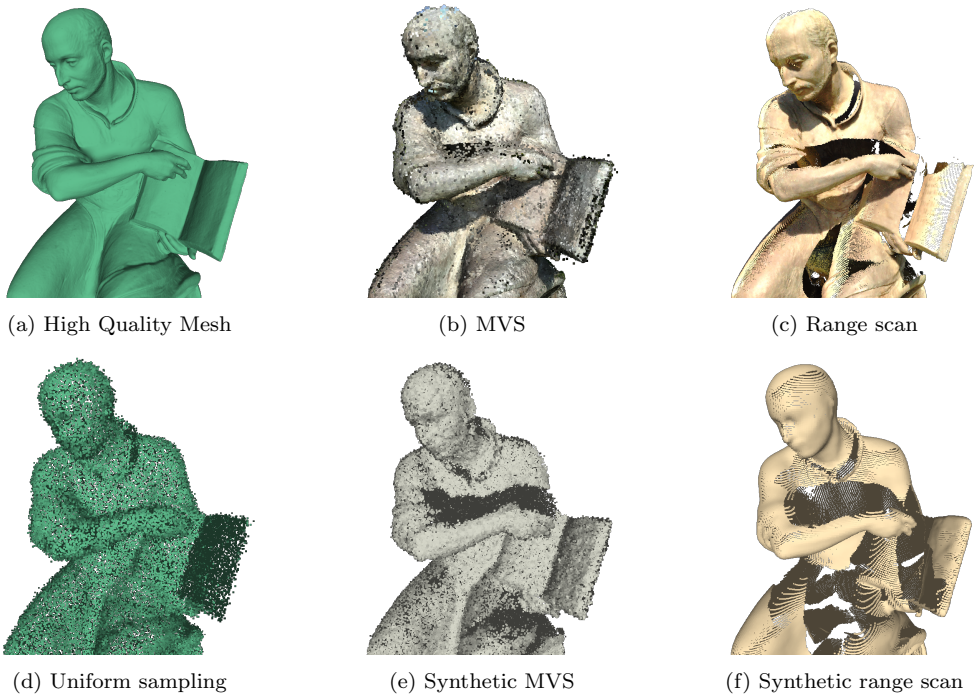
Synthetic scanning for point cloud generation In an ideal setting, we would evaluate methods on real point cloud acquisitions together with their true surfaces. However, generating true surfaces of real objects requires error free and dense input point clouds or substantial manual intervention. Therefore, such a dataset is difficult to produce. MVS benchmarks [61, 68, 99, 100, 107] commonly use image acquisitions for the reconstruction input and a highly complete and precise acquisition, *e.g.* from multiple stationary LiDAR scans as reference. We make use of such datasets for evaluation. Using such a dataset for training surface reconstruction networks requires reconstructing

Table 2.2: Scanning configurations for Berger et al. benchmark: We show the five different scanner configurations used in our modified version of the Berger et al.'s scanning procedure. We use the resulting scans to evaluate object-level reconstruction with varying point-cloud defects and for training data generation. For the low resolution (LR) scans the scanning process results in 1000 to 3000 points per shape, and for the high resolution (HR), the scanning process yields around 10 000 to 30 000 points.

	Low res. (LR)	**High res. (HR)**	**HR + noise (HRN)**	**HR + outliers (HRO)**	**HR + noise + outliers (HRNO)**
Camera resolution x, y	50, 50	100, 100	100, 100	100, 100	100, 100
Scanner positions	5	10	10	10	10
Min/max range	70/300	70/300	70/300	70/300	70/300
Additive noise	0	0	0.5	0	0.5
Outliers (%)	0	0	0	0.1	0.1

a watertight surface from the high-quality acquisition. However, even with high-quality acquisitions, parts of the object or scene may be missing due to occlusions, for example. These issues ultimately lead to inconsistencies in the ground truth and make this source of data unreliable to train DSR networks. Additionally, existing datasets of point cloud acquisitions and reliable ground truth surface information only consist of a handful of objects or scenes. Instead, training and evaluation of learning-based surface reconstruction is often done on point clouds sampled from synthetic surfaces stemming from large shape collections. However, such point clouds are not representative for real-world acquisitions, as they do not model non-uniformity or missing data stemming *e.g.* from occlusions, or transparent and low texture areas. To this end, we resort to synthetic scanning to produce point clouds from synthetic surfaces in our benchmark. In contrast to directly sampling the surfaces, synthetic scanning can produce point clouds with realistic defects, such as anisotropy and missing data from (self-)occlusion, see Figure 2.2. At the same time, the synthetic surfaces provide reliable information for training and evaluation.

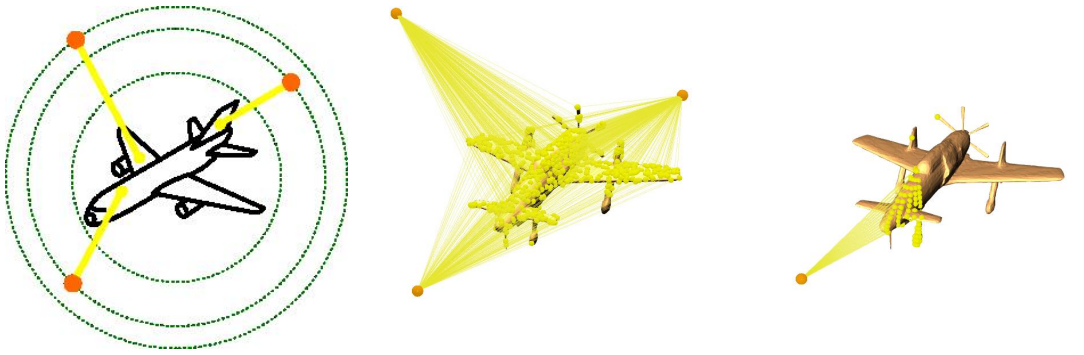
Synthetic range scanning We use the range scanning procedure from the surface reconstruction benchmark of Berger *et al.* [8]. To this end, we modified their provided code to export the camera positions of the scanning process along with the point cloud. We also add outliers to the produced point clouds by uniformly sampling the bounding box of the object. The scanning procedure produces uniform, evenly spaced point clouds. We choose five different scanner settings to scan each test shape: (i) a low resolution setting replicates point clouds obtained from long range scanning and (ii) a high resolution setting produces point clouds with close to no defects. Three further settings produce high resolution point clouds with challenging defects such as (iii) noise, (iv) outliers or (v) noise and outlier defects combined. See Table 2.2 for details. Because Berger *et al.*'s provided code pipeline is too time and memory extensive, we cannot generate a dataset sufficiently large for training DSR methods. Thus, we only use this dataset for testing. We refer the reader to the original benchmark paper [8] for further



(a) High Quality Mesh (b) MVS (c) Range scan

(d) Uniform sampling (e) Synthetic MVS (f) Synthetic range scan

Figure 2.2: Synthetic and real point clouds: Surface reconstruction methods are often tested on uniform surface samplings (d). Instead, we test methods on synthetic MVS (e) and synthetic range scans (f). In contrast to uniform surface sampling, synthetic scanning can produce realistic point cloud defects, such as missing data from occlusion, often present in real scans (b,c).



(a) Synthetic scanning setup (b) Synthetic MVS (c) Synthetic range scanning

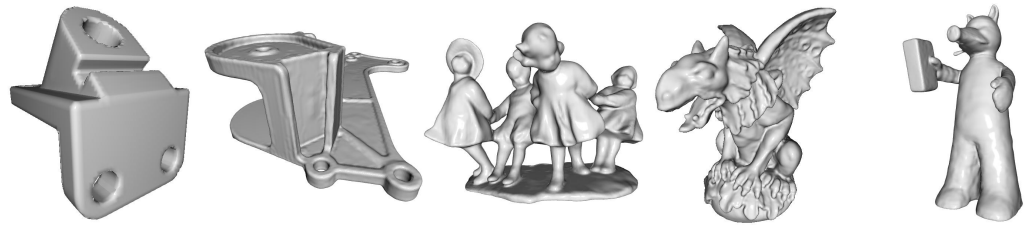
Figure 2.3: Synthetic scanning procedure: We randomly place sensors on bounding spheres with multiple radii around the object (a). To produce MVS like point clouds, we consider rays aiming at uniformly sampled points on the circumsphere of the object (b). This produces non-uniform point clouds with missing data similar to real MVS point clouds. For synthetic range scanning, we use Berger *et al.*'s [8] pipeline, which considers ray targets arranged on a uniform grid aiming at the object (c). This produces uniform point clouds with missing data similar to real range scanning point clouds.

details about the scanning pipeline.

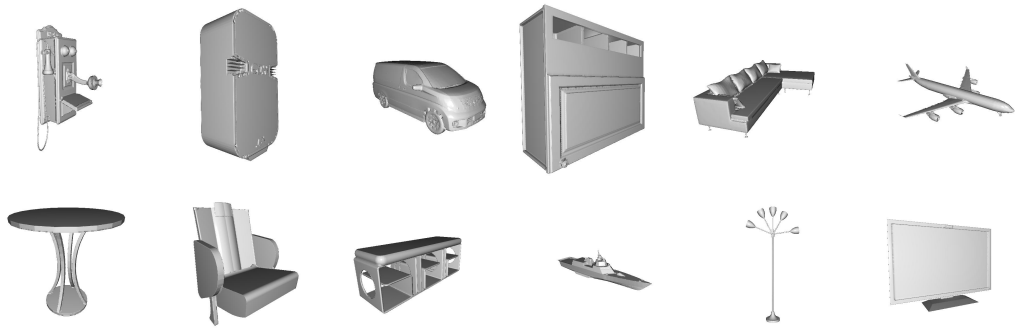
Synthetic MVS To mimic MVS acquisitions, we synthetically scan objects by placing virtual sensors on two bounding spheres around an object and shooting rays to the circumsphere of the object. Sensor positions (ray origin) and ray target points are uniformly sampled on the surface of the spheres. A 3D point is then given as the intersection of the ray and the objects surface. Our goal is not to mimic an MVS pipeline but rather produce point clouds with similar characteristics. We depict our scanning procedure in Figure 2.3. We produce two different scans with our approach: (i) sparse point clouds with $3,000$ points per object and Gaussian noise on the point position with zero mean and standard deviation 0.005 as in [92], and (ii) dense point clouds with $10,000$ points per object of which 10% are outliers and Gaussian noise on the point position with zero mean and standard deviation 0.005. For both versions we scan from 10 different sensor positions.

2.5.1 Datasets

We consider a variety of datasets to evaluate the versatility and precision of different reconstruction methods. We use closed surfaces from ShapeNet, ModelNet and Berger *et al.*, as they are widely available. ShapeNet and ModelNet are sufficiently big to train surface reconstruction networks. Most learning-based methods require reliable



(a) Berger *et al.*



(b) ShapeNet

(c) ModelNet

Figure 2.4: Ground truth shapes of the benchmark datasets: We show an example shape of each class of ModelNet in (c) and of ShapeNet in (b) and the five shapes of Berger *et al.* in (a).

inside/outside querying of the models for training. To this end, we make the models watertight using ManifoldPlus [56]. Note that we also use the train sets to tune the parameters of learning-free methods. The watertight surfaces of the test sets allow for a reliable quantitative evaluation of the reconstructions. For qualitative evaluation, we also test on real scans [61, 68, 100] which further allows us to evaluate the reconstruction of open surfaces. All surfaces are scaled to be contained inside the unit cube. In the following we give additional details for each dataset used in our benchmark. See Figure 2.4 for example shapes.

ShapeNet As is common practice in related studies, we use Choy *et al.*'s [38] 13 class subset of ShapeNet as well as its train/val/test split. We generate point clouds with $3,000$ and $10,000$ points using our synthetic MVS-like scanning.

ModelNet10 We use ModelNet10 shapes as a second object shape dataset. Its shapes are less complex than ShapeNet's, with more flat surfaces and fewer details. Additionally, the number of training shapes is smaller ($4k$ vs $30k$ objects). We use the full train set and the test sets for the 6 out of 10 classes which are not represented in ShapeNet. We generate point clouds with $3,000$ points with our synthetic MVS-like scanning.

Berger *et al.* We select five shapes from the benchmark of Berger *et al.*. These shapes include challenging characteristics such as details of various sizes or a non-trivial topology, which makes them more difficult to reconstruct than ModelNet shapes. We generate point clouds between $3,000$ and $10,000$ points using our synthetic MVS and range scanning procedures.

Real MVS and range scans We select a range scan from Tanks and Temples [68], and two MVS point clouds from DTU [61] and from Middlebury [100]. We subsample these point clouds to $50,000$ points.

2.5.2 Experimental Setup

We show a summary of our experimental setup on Table 2.3 and Table 2.4. In the following, we provide details for each experiment.

In-distribution (E1) First, we train and evaluate methods on ShapeNet using all 13 categories and sparse point clouds with $3,000$ points and Gaussian noise with zero mean and standard deviation 0.005. With this experiment, we evaluate the capacity of learning methods to complete missing data of sparse point clouds and eliminate noise.

Table 2.3: Benchmark setup: In E1 to E4, we train surface reconstruction methods on noisy point clouds of ShapeNet objects. In E1, we test on ShapeNet. In E2, we test on ShapeNet, but with denser point clouds and 10% outliers. In E3, we test on the simpler ModelNet objects. In E4, we test on the Berger *et al.* shapes. In E5, we train the methods on the simpler ModelNet dataset and test on ShapeNet. In E6, we test optimization-based methods on synthetic range scans of the Berger *et al.* dataset. In E7 and E8, we directly compare learning- and optimization-based methods on synthetic and real point clouds.

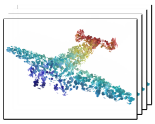
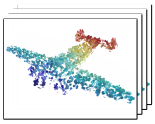
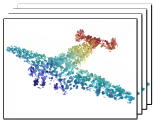
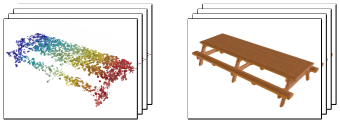
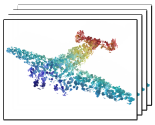
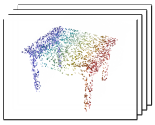
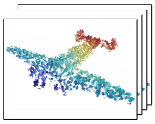
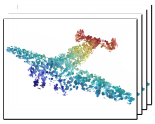
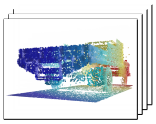
Experiment	Training set	Test set
1 In-distribution	 ShapeNet (synthetic MVS)	 ShapeNet (synthetic MVS)
2 Out-of-distribution *unseen point cloud characteristics*	 ShapeNet (synthetic MVS)	 ShapeNet (synthetic MVS)
3 Out-of-distribution *unseen shape categories, less complex*	 ShapeNet (synthetic MVS)	 ModelNet (synthetic MVS)
4 Out-of-distribution *unseen shape categories, similar complexity*	 ShapeNet (synthetic MVS)	 Berger *et al.* (synthetic MVS)
5 Out-of-distribution *unseen shape categories, more complex*	 ModelNet (synthetic MVS)	 ShapeNet (synthetic MVS)
6 Optimization	—	—	 Berger *et al.* (synthetic range scan)
7 Out-of-distribution vs. optimization *unseen shape categories vs. optimization*	 ShapeNet (synthetic MVS)	 Berger *et al.* (synthetic MVS)
8 Out-of-distribution vs. optimization *unseen point cloud characteristics and shape categories vs. optimization*	 ShapeNet (synthetic MVS)	 Real MVS & range scan	—

Table 2.4: Detailed benchmark setup: We present the four different experiments of our benchmark for surface reconstruction algorithms. For E1 to E4, we train methods to reconstruct ShapeNet objects from noisy point clouds. In E1, we test on the ShapeNet test set. In E2, we test on ShapeNet, but reconstruct from denser point clouds with noise and outliers. In E3, we test on ModelNet with the same sampling as in E1. In E4, we test on five Berger *et al.* shapes with the same sampling as in E1. Finally, in E5, we train the methods on ModelNet and test on ShapeNet, with the same sampling as in E1.

	Training set						Test set					
Experiment	Name	# shapes	complexity	# points	σ noise	% outliers	Name	# shapes	complexity	# points	σ noise	% outliers
1	ShapeNet	30,661	⋆⋆	3,000	0.005	0	ShapeNet	1,300	⋆⋆	3,000	0.005	0
2	ShapeNet	30,661	⋆⋆	3,000	0.005	0	ShapeNet	1,300	⋆⋆	10k	0.005	10
3	ShapeNet	30,661	⋆⋆	3,000	0.005	0	ModelNet	506	⋆	3,000	0.005	0
4	ShapeNet	30,661	⋆⋆	3,000	0.005	0	Berger *et al.*	5	⋆⋆	3,000	0.005	0
5	ModelNet	3,979	⋆	3,000	0.005	0	ShapeNet	1,300	⋆⋆	3,000	0.005	0
6	–						Berger *et al.*	5	⋆⋆	see Table 2.2		
7	see Section 2.5.2						Berger *et al.*	5	⋆⋆	3,000	0.005	0
8	ShapeNet	30,661	⋆⋆	3,000	0.005	0	Real	3	⋆⋆⋆	10,000	*variable*	*variable*

Out-of-distribution (unseen point cloud characteristics) (E2) We evaluate the models trained in E1 on test shapes scanned with a different setting than the train shapes. We use dense point clouds with 10,000 points of which 10% are outliers. We add the same noise as in E1. Here, we investigate whether learning methods are able to generalize to different point cloud characteristics.

Out-of-distribution (unseen shape categories, less complex) (E3) We evaluate the models trained in E1 on shapes from unseen categories but with the same point cloud characteristics. We use six categories of ModelNet which are not present in the ShapeNet training set. In this experiment, we investigate whether learning methods generalize to unseen categories.

Out-of-distribution (unseen shape categories, similar complexity) (E4) This experiment is similar to E3, but the test set is comprised of five shapes from Berger *et al.* which do not correspond to ShapeNet's categories, but have similar complexity.

Out-of-distribution (unseen shape categories, more complex (E5) This experiment is similar to E3 and E4, but we retrain all methods on the simpler shapes from ModelNet10. Here, we assess whether learning methods can generalize from simple shapes to more complex ones, a difficult out-of-distribution setting.

Optimization (E6) We evaluate several recently developed optimization-based methods, and two traditional test-of-time optimization-based methods. We use the Berger *et al.* dataset for this experiment.

Out-of-category vs. optimization (E7) We compare learning- and optimization-based methods on the same dataset. For this we run optimization-based methods on MVS scans of the Berger *et al.* shapes and compare the results to experiment E4.

Out-of-distribution vs. optimization (E8) Finally, we compare learning- and optimization-based methods on real MVS and range scanning point clouds. For learning-based methods we use the models from E1.

2.5.3 Surface reconstruction methods

We briefly describe the optimization- and learning-based methods that we will benchmark below. For a more complete description of these methods and their related concepts we refer the reader to our survey in Section 2.4. Note that while some of the optimization-based methods are based on deep networks, and we call them DSR methods, they do not learn shape priors from a training set. Instead, the networks are "trained" (or optimized) for each new point cloud to reconstruct a surface and rely on novel regularization techniques to increase their robustness to noise, outliers and missing data. Conversely, while some traditional methods are not based on a deep network architecture, we tune their (hyper)parameters on the training set by using a grid search over different parameter combinations. When we need to extract a surface from an implicit field, we use marching cubes [76] with a resolution of 128^3.

Optimization-based methods

IGR [48] Implicit Geometric Regularisation (IGR) is a DSR method, operating directly on the point cloud using a simple fully connected network architecture that estimates an indicator function from point positions and normals. We optimize the network weights for $100,000$ iterations for each scan/shape.

LIG [62] Local Implicit Grids (LIG) trains an autoencoder to encode crops of a signed distance function gained from ground truth shapes. For inference, only the decoder part of the autoencoder is retained. Then, crops of the input point cloud with oriented normals are augmented with 10 new points along each normal, representing ground truth signed distance information. An initial latent vector is then decoded to produce an SDF and iteratively optimized so that the augmented point cloud crop best matches the SDF. A post-processing removes falsely-enclosed volumes. As code for training is unavailable, we only use the optimization part, with a pretrained model on ShapeNet (without noise). We use the sensor position to orient jet-estimated normals [29].

P2M [52] Point2Mesh (P2M) is an optimization-based method which iteratively moves vertices of an initial mesh to fit a point cloud.

SAP [91] Shape As Points (SAP) has a supervised learning- and an optimization-based variant. In the learning variant, the method estimates the oriented normals as well as k point offsets for each input point, to adjust and densify the point cloud. The resulting point cloud of size $k \mid \mathcal{P} \mid$ is then used by a differentiable Poisson solver [65] to compute an indicator grid, which is supervised with a ground truth indicator grid computed prior to training. The entire pipeline is differentiable which allows for updating point offsets, oriented normals and the network parameters.

SPSR [65] Screened Poisson Surface Reconstruction (SPSR) is a classic non learning-based method which approximates the surface as a level-set of an implicit function estimated from point positions and normal information. We use the sensor position to orient jet-estimated normals [29]. We chose an octree of depth 10 and Dirichlet boundary condition. We also use the provided surface trimming tool for post-processing, but could not find parameters that consistently improve the reconstructed surface.

Labatut *et al.* [71] Labatut *et al.* is a graph-cut-based method for range scans that makes use of visibility information. Because there is no official implementation of the algorithm, we reimplemented it ourselves. To compare with optimization-based methods, we use the parametrization suggested by the authors: point weights $\alpha_{vis} = 32$ and $\sigma = 0.01$; regularization strength $\lambda = 5$.

Learning-based methods

ConvONet [92] Convolutional Occupancy Networks (ConvONet) is a DSR method that first extracts point features and averages them on cells of three 2D grids, or one 3D grid (variant). 2D or 3D grid convolutions then create features capturing the local geometry. Last, the occupancy of a query-point is estimated with a fully connected network from interpolated features stored on each node of the 2D or 3D grid.

SAP [91] In the optimization variant, the method starts as the learning-based variant described above. Then, the estimated indicator grid is used to compute a mesh and points are sampled on the mesh to calculate a Chamfer loss between the mesh and input point cloud.

DGNN [109] This method uses a graph neural network to estimate the occupancy of Delaunay cells in a point cloud tetrahedralization from cell geometry and visibility features. A graph-cut-based optimization then reinforces global consistency.

POCO [21] Point Convolution for Surface Reconstruction (POCO) extracts point features using point cloud convolution [19], then estimates the occupancy of a query point with a learning-based interpolation from nearest neighbors.

SPRS [65] See method description above. For the learning-based experiments, we perform a grid search over octree depth $d = \{6, 8, 10, 12\}$ and boundary conditions $b = \{\text{dirichlet, neumann, free}\}$. We use the parametrization with the best mean volumetric IoU for reconstructions of the training set.

Labatut *et al.* [71] See method description above. For the learning-based experiments, we perform a grid search over regularization strength $\lambda = \{1.5, 2.5, 5, 10\}$, and point weights $\alpha = \{16, 32, 48\}$ and $\sigma = \{0.001, 0.01, 0.1, 1\}$. We use the parametrization with the best mean volumetric IoU for reconstructions of the training set.

2.5.4 Evaluation metrics

We want the reconstructed surface $\mathcal{S}^r$ to be as close as possible to the real (or ground truth) surface $\mathcal{S}$ in terms of geometry and topology. To measure this "closeness" we use several metrics.

Geometric metrics

We evaluate the geometric quality of reconstructions with the volumetric intersection over Union (IoU), symmetric Chamfer distance (CD) and normal consistency (NC).

Volumetric IoU In the following, let $\mathcal{S}^g$ and $\mathcal{S}^r$ be the set of all points that are inside or on the ground truth and reconstructed surface, respectively. The volumetric IoU is defined as:

$$\text{IoU}\left(\mathcal{S}^g, \mathcal{S}^r\right) = \frac{|\mathcal{S}^g \cap \mathcal{S}^r|}{|\mathcal{S}^g \cup \mathcal{S}^r|} \ .$$

We approximate volumetric IoU by randomly sampling $100,000$ points in the union of the bounding boxes of $\mathcal{S}^g$ and $\mathcal{S}^r$.

Chamfer distance To compute the Chamfer distance and normal consistency, we sample a set of points $\mathcal{P}^g$ and $\mathcal{P}^r$ on the facets of the ground truth mesh and the reconstructed mesh, respectively, with $|\mathcal{P}^g| = |\mathcal{P}^r| = 100,000$. We approximate the

symmetric Chamfer distance between $\mathcal{S}^g$ and $\mathcal{S}^r$ as follows:

$$\begin{aligned}\mathrm{CD}(\mathcal{S}^g, \mathcal{S}^r) =& \frac{1}{2|\mathcal{P}^g|} \sum_{x \in \mathcal{P}^g} \min_{y \in \mathcal{P}^r} ||x - y||_2 \\ &+ \frac{1}{2|\mathcal{P}^r|} \sum_{y \in \mathcal{P}^r} \min_{x \in \mathcal{P}^g} ||y - x||_2 \ .\end{aligned}$$

Normal consistency Let $n(x)$ be the unit normal of a point x. We set this normal to be the normal of the facet from which x was sampled. Let $\langle \cdot, \cdot \rangle$ the Euclidean scalar product in $\mathbb{R}^3$. Normal consistency is defined as:

$$\begin{aligned}\mathrm{NC}(\mathcal{S}^g, \mathcal{S}^r) =& \frac{1}{2|\mathcal{P}^g|} \sum_{x \in \mathcal{P}^g} \left\langle n(x), n\left(\operatorname*{argmin}_{y \in \mathcal{P}^r} ||x - y||_2 \right) \right\rangle \\ &+ \frac{1}{2|\mathcal{P}^r|} \sum_{y \in \mathcal{P}^r} \left\langle n(y), n\left(\operatorname*{argmin}_{x \in \mathcal{P}^g} ||y - x||_2 \right) \right\rangle \ .\end{aligned}$$

Topological metrics

We evaluate the topological quality of reconstructions through the number of components, the number of non-manifold edges and the number of boundary edges.

Number of components If not stated otherwise, the ground truth surfaces of our datasets have exactly one component. In consequence, the reconstructed surfaces should also have one component.

Number of boundary edges The surfaces of all ground truth objects in our datasets are closed. We verify this by measuring the number of boundary edges of the reconstructed meshed surface which should be zero. Note that if boundary edges only appear on the intersection of the reconstruction with its bounding box we still classify the reconstruction as watertight, according to the definition in Section 2.3.3.

Number of non-manifold edges The surfaces of all ground truth objects in our datasets are 2-manifolds. We verify this by measuring the number of non-manifold edges of the reconstructed meshed surface which should be zero.

Runtimes

To evaluate the scalability of methods, we measure the average time it takes to reconstruct a surface of ShapeNet from 3,000 points.

2.6 Experiments

2.6.1 Learning-based surface reconstruction from synthetic MVS point clouds (E1 - E5)

We examine the precision and versatility of novel supervised-learning methods and two traditional methods for which training sets were used for tuning parameters. All evaluated methods perform well when reconstructing shapes from known categories and known point cloud characteristics (E1). The learning-based methods show a significantly superior performance of at least 5% over SPSR and Labatut *et al.* (see Table 2.5). The methods based on neural implicit fields (POCO, SAP and ConvONet) produce visually and quantitatively the best reconstructions (see Figure 2.5, first column). DGNN does not perform as well as most other learning methods in this experiment. The sparse point clouds used in this experiment do not contain point samples on all details. However, due to the interpolating nature of DGNN surface details cannot be reconstructed without input points.

In E2, domain shifts results in worse performance, both quantitatively and qualitatively for all methods except SPSR. SPSR shows robustness against outliers and benefits from the higher point density. Most learning methods do not produce satisfying results (see Figure 2.5, second column). The reconstruction of SAP is too smooth and lacks details, but does not show as severe defects as the reconstructions of other learning-based methods. Labatut *et al.* suffers from the low regularization weight tuned for the outlier free point clouds and could benefit from higher regularization to remove erroneous floating components from outliers.

When reconstructing out-of-category ModelNet shapes (E3), the neural implicit field methods exhibit visually the best reconstructions. SAP and POCO produce quantitatively the best reconstructions (see Table 2.5). The interpolating method DGNN performs better than ConvONet.

In E4, we reconstruct shapes from Berger *et al.* which have similar complexity than the shapes from ShapeNet used for training. The only learning methods able to leverage information from the common point cloud characteristics to improve the test results are DGNN and POCO.

In E5, most methods overfit the simpler ModelNet shapes when retrained and used to reconstruct the more complex ShapeNet shapes. Even SPSR slightly suffers from tuning parameters on ModelNet. The best reconstructions on ModelNet are achieved with an octree depth of $d = 8$ (instead of $d = 10$ on ShapeNet) leading to worse results on ShapeNet: 77.1 vIoU in E1 vs. 74.6 vIoU in E5. The parameter tuning of Labatut *et al.* stays unchanged. DGNN is the only method that does not overfit on ModelNet and yields the best results, both quantitatively and qualitatively. In fact, it performs as well as when trained on ShapeNet directly.

ConvONet is only able to outperform traditional methods when the training and

Table 2.5: Numerical results for learning-based reconstructions (E1 to E5): SPSR [65] is the only method that reconstructs surfaces with a high volumetric intersection over Union (IoU) and a low Chamfer distance (CD) in each experiment. Therefore, its reconstructions have the highest mean volumetric IoU and the lowest mean CD. However, SPSR also reconstructs the least compact surfaces on average (*i.e.* surfaces with the highest number of components). Labatut *et al.* [71] reconstructs the most compact surfaces. The reconstructions of DGNN [109] have the highest mean volumetric IoU of the tested learning methods. The reconstructions of SAP [91] have the lowest mean CD of the tested learning methods and the highest normal consistency. ConvONet and SPSR are the only methods that reconstruct surfaces without boundary and non-manifold edges.

		Volumetric IoU (%) [↑]						**Normal consistency** (%) [↑]					
Method		E1	E2	E3	E4	E5	Mean	E1	E2	E3	E4	E5	Mean
ConvONet2D	[92]	85	47.3	79.3	65.1	68.3	69	92.7	76.4	90	78	87.8	85
ConvONet3D	[92]	84.8	15.1	83.6	76.4	51	62.2	93	71.8	93.1	87.2	82.5	85.5
SAP	[91]	88.7	59.8	89.2	78.3	54.9	74.2	93.5	**86.7**	94.1	89	87.1	**90.1**
DGNN	[109]	84.5	38.1	87	82.9	**84.4**	75.4	85.4	68.8	88.5	85.2	85.5	82.7
POCO	[21]	**89.5**	8.74	**90.6**	**83.9**	40.9	62.7	**93.6**	75.6	**94.2**	**89.5**	82.9	87.1
SPSR	[65]	77.1	**80.7**	80.7	77.6	74.6	**78.1**	87.7	83.2	89.1	86.3	**88**	86.9
Labatut *et al.*	[71]	80.3	60.4	83.9	79.4	80.3	76.9	81	73	84.6	80.8	81	80.1
		Chamfer distance (per-point ave. %) [↓]						**Number of components** [↓]					
Method		E1	E2	E3	E4	E5	Mean	E1	E2	E3	E4	E5	Mean
ConvONet2D	[92]	0.553	7.51	0.997	1.43	0.979	2.29	1.6	34.8	2.55	3.6	3.2	9.16
ConvONet3D	[92]	0.546	10.9	0.76	0.887	2.44	3.1	1.37	13.6	1.6	2.6	1.5	4.13
SAP	[91]	0.437	2.09	0.547	0.734	0.924	0.946	2.71	86	3.45	5.6	10.5	21.7
DGNN	[109]	0.549	2.54	0.635	0.586	**0.55**	0.973	1.31	16.1	1.13	**1**	1.31	4.16
POCO	[21]	**0.416**	10.5	**0.516**	**0.579**	1.32	2.67	2.32	178	2.82	2	16.3	40.2
SPSR	[65]	0.801	**0.659**	0.873	0.786	0.886	**0.801**	9.26	185	11.1	8	3.24	43.3
Labatut *et al.*	[71]	0.665	6.97	0.747	0.671	0.665	1.94	**1.22**	**9.02**	**1.05**	**1**	**1.22**	**2.7**
		Number of boundary edges [↓]						**Number of non-manifold edges** [↓]					
Method		E1	E2	E3	E4	E5	Mean	E1	E2	E3	E4	E5	Mean
ConvONet2D	[92]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ConvONet3D	[92]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
SAP	[91]	**0**	0.00923	**0**	**0**	8.44	1.69	**0**	**0**	**0**	**0**	**0**	**0**
DGNN	[109]	**0**	**0**	**0**	**0**	**0**	**0**	1.35	2.24	0.646	0.4	1.69	1.26
POCO	[21]	**0**	121	**0**	**0**	41.7	32.5	**0**	0.00154	**0**	**0**	**0**	0.000308
SPSR	[65]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Labatut *et al.*	[71]	**0**	**0**	**0**	**0**	**0**	**0**	9.35	28.5	8.47	9.6	9.35	13.1

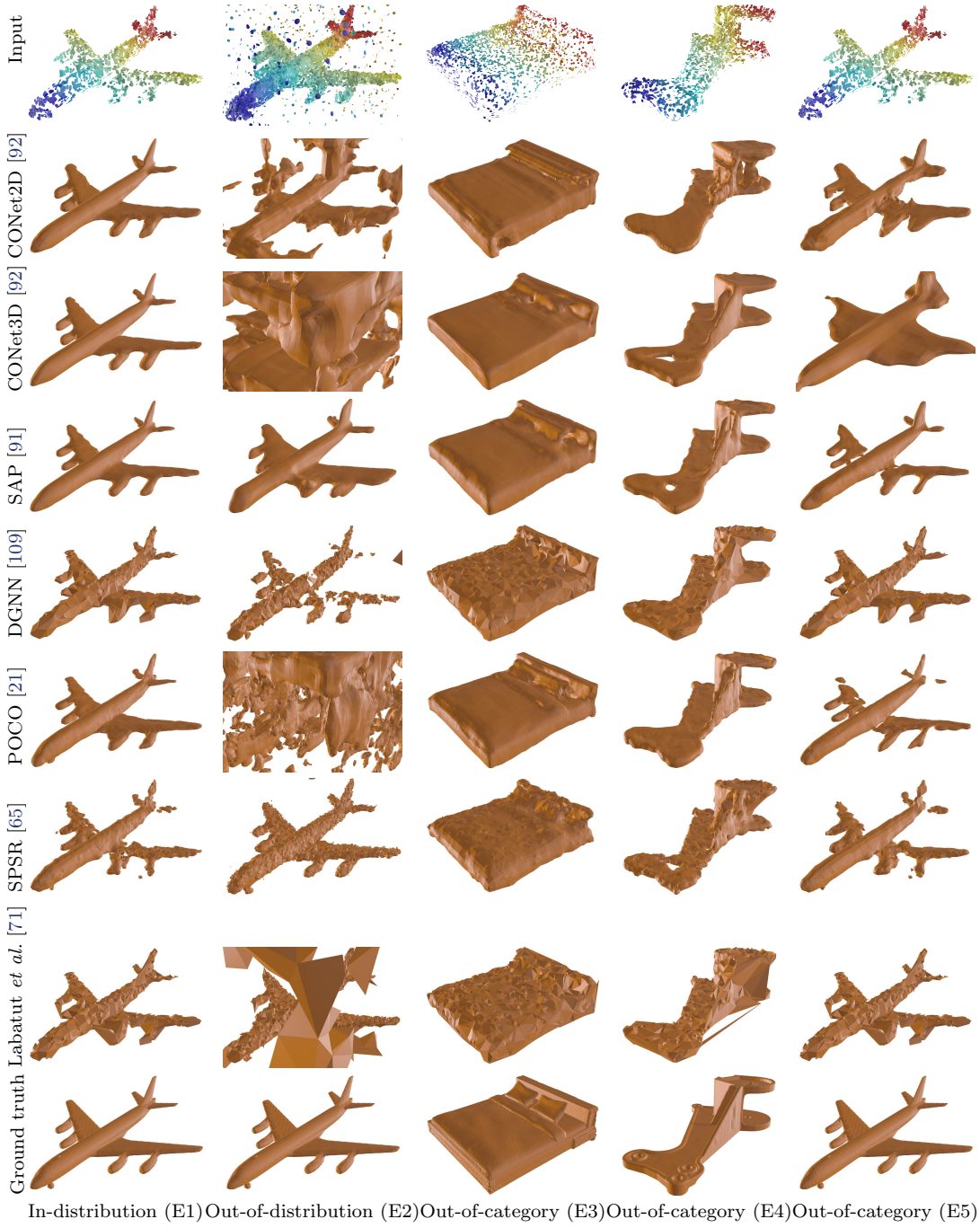


Figure 2.5: Learning-based reconstructions (E1 to E5): DGNN [109], SAP [91] and SPSR [65] provide visually the best reconstructions without prevalent defects.

test sets share the same point cloud characteristics *and* shape categories. SAP produces much better reconstructions and is the learning-based method with the highest robustness against outliers. It is also the only method explicitly predicting normals. As a result SAP reconstructs surfaces with the highest mean normal consistency over all experiments. The local learning and global regularisation approach of DGNN produces competitive results in all experiments, except for the outlier setting of E2. DGNN is the learning-based method producing surfaces with highest mean IoU over all experiments. The local attention-based learning mechanism of POCO leads to the best results when the task does not involve reconstruction from unseen domains. It provides the most faithful reconstructions in three experiments in which point cloud characteristics are identical in train and test set (E1, E3, E4). However, POCO is heavily affected by outliers (E2), which can be explained by its purely local approach. POCO also tends to overfit on simple training shapes (E5). The reconstructions of POCO, as well as the ones of SAP contain boundary edges only in areas where the reconstructions intersect the bounding box *i.e.* they are still watertight. SPSR proves robust to various defects and shape characteristics, providing fair results, with the highest mean IoU and Chamfer distance across the board. However, its reconstructions are the least compact, *i.e.* they have the highest number of components. Labatut *et al.*'s parametrization proves slightly less robust, as the method is affected by outliers. Its mean IoU is higher than that of any learning method, and its reconstructions are the most compact surfaces with an average number of components of 2.7. However, it is also the only method that produces a significant amount of non-manifold edges.

2.6.2 Optimization-based surface reconstruction from synthetic range scanning point clouds (E6)

This experiment evaluates the precision and versatility of non-learning methods. The benchmarked approaches consist in neural network based methods optimizing a function to fit an input point cloud and rely on novel regularization techniques to increase their robustness to noise, outliers and missing data. Furthermore, we benchmark the two traditional methods SPSR and Labatut *et al.* with standard parameter settings. We reconstruct surfaces of Berger *et al.* from synthetic range scanning point clouds with various different defects. We show numerical results in Table 2.6 and visualisations in Figure 2.6. Almost all reconstructions provided by the two traditional methods are much more truthful than the DSR methods, with a mean volumetric IoU almost 10 points higher across all point cloud defects. IGR does visually not provide a good result on the exemplary shape, especially on thin surface parts. Quantitatively, the method provides the best reconstruction for the neural networks based methods in the absence of outliers, and even the best overall reconstruction for the noisy high resolution scans. LIG does not provide good reconstructions for any of the settings. This can be explained by its pretrained model on defect-free uniform high density point clouds. Furthermore,

Table 2.6: Numerical results for optimization-based surface reconstruction (E6): Optimization-based reconstruction of the Berger *et al.* shapes from synthetic range scans. LR is a low resolution scan, HR a high resolution scan, HRN a high resolution scan with noise, HRO a high resolution scan with outliers, and HRNO a high resolution scan with noise and outliers. The methods are optimized per shape and per scan using standard settings as mentioned in the corresponding publications.

Method		**Volumetric IoU** (%) [↑]						**Normal consistency** (%) [↑]					
		LR	HR	HRN	HRO	HRNO	Mean	LR	HR	HRN	HRO	HRNO	Mean
IGR	[48]	80.8	92.5	**83.6**	63.7	62.7	76.7	88	**96.3**	83.9	77.8	71.5	83.5
LIG	[62]	46.9	50.3	63.9	66	63.8	58.2	**88.7**	92.2	**89**	77	75.2	84.4
P2M	[52]	75.2	83.3	75.5	71.3	67.8	74.6	86.3	92.2	88.1	84.5	82.1	86.6
SAP	[91]	75.6	89.1	72.4	55.3	34.9	65.4	83.4	94.8	61.6	74.5	55.3	73.9
SPSR	[65]	77.7	90.2	82.8	90.3	**82.1**	84.6	88.1	96	88.1	**96.2**	**85.8**	**90.9**
Labatut *et al.*	[71]	**81.3**	**93.4**	80.1	**93.4**	79.1	**85.5**	87.6	96	66.3	94.9	66.5	82.3
Method		**Chamfer distance** (per-point ave. %) [↓]						**Number of components** [↓]					
		LR	HR	HRN	HRO	HRNO	Mean	LR	HR	HRN	HRO	HRNO	Mean
IGR	[48]	0.674	0.322	**0.554**	7.96	7.72	3.45	6.8	**1.2**	35.2	44	97.4	36.9
LIG	[62]	0.745	0.581	0.781	7.89	7.8	3.56	**1**	**1**	**1**	1.6	**1**	1.12
P2M	[52]	0.817	0.473	0.729	1.53	2.13	1.13	**1.2**	**1**	**1.2**	1.4	1.6	1.28
SAP	[91]	0.852	0.32	0.701	3.99	3.93	1.96	73.2	85.6	937	1.8e+03	1.96e+03	971
SPSR	[65]	0.794	0.369	0.572	0.362	**0.607**	0.541	**1.2**	1.6	3.6	3.8	20.2	6.08
Labatut *et al.*	[71]	**0.635**	**0.314**	0.608	**0.339**	0.641	**0.507**	**1**	**1**	**1.2**	**1.2**	**1**	**1.08**
Method		**Number of boundary edges** [↓]						**Number of non-manifold edges** [↓]					
		LR	HR	HRN	HRO	HRNO	Mean	LR	HR	HRN	HRO	HRNO	Mean
IGR	[48]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0.8	0.8	5.2	4.2	2.2
LIG	[62]	69	42.8	17.2	**0**	**0**	25.8	**0**	**0**	**0**	**0**	**0**	**0**
P2M	[52]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
SAP	[91]	**0**	**0**	**0**	**0**	449	89.8	**0**	**0**	**0**	**0**	**0**	**0**
SPSR	[65]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Labatut *et al.*	[71]	**0**	**0**	**0**	**0**	**0**	**0**	1	5.8	24.4	3.8	22	11.4

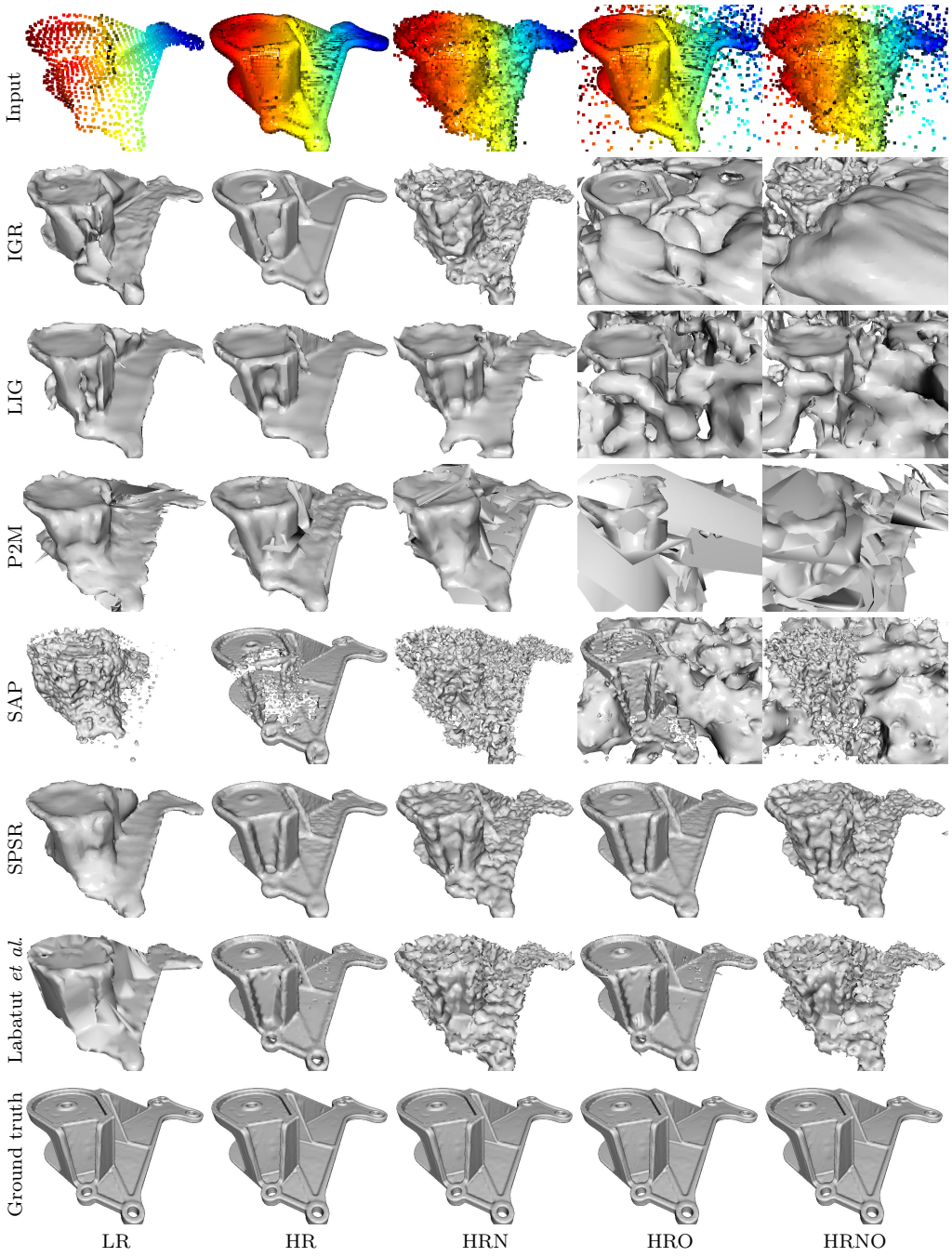


Figure 2.6: Optimization-based experiments: In each column we show the results of different methods of one of the five learning-based experiments.

Table 2.7: Numerical results for learning- and optimization-based reconstructions (E7): Learning- and optimization-based reconstruction of the Berger *et al.* test shapes from synthetic MVS scans. The learning methods were trained on synthetic MVS scans from ShapeNet. Optimization-based methods are optimized per scan using standard settings. BE stands for boundary edges and NME for non-manifold edges. Only the learning-based methods POCO and DGNN reconstruct surfaces with a higher mean volumetric intersection over Union and lower Chamfer distance than all optimization-based methods.

Method	Vol. IoU [↑]	Normal consist. [↑]	Chamfer dist. [↓]	Components [↓]	BE [↓]	NME [↓]
Learning						
ConvONet2D [92]	65.1	78	1.43	3.6	**0**	**0**
ConvONet3D [92]	76.4	87.2	0.887	2.6	**0**	**0**
SAP [91]	78.3	89	0.734	5.6	**0**	**0**
DGNN [109]	82.9	85.2	0.586	**1**	**0**	0.4
POCO [21]	**83.9**	**89.5**	**0.579**	2	**0**	**0**
Optimization						
IGR [48]	78.3	83.8	0.775	15.4	**0**	0.4
LIG [62]	45.7	86.6	0.831	**1**	65.6	**0**
P2M [52]	74.5	85	0.768	2	**0**	**0**
SAP [91]	71.9	77	0.811	133	**0**	**0**
SPSR [65]	77.6	86.4	0.785	8	**0**	**0**
Labatut *et al.* [71]	79.4	80.8	0.671	**1**	**0**	9.6

its post-processing makes the reconstructions non-watertight. P2M provides geometrically fair reconstructions and the topologically best reconstructions with a low number of components, and watertight and manifold surfaces for all reconstructions. SAP provides fair reconstructions in the absence of outliers. None of the neural network based methods is robust against outliers. As in the learning-based experiments, SPSR generates high quality reconstructions for all input defects, and achieves the best mean normal consistency. Labatut *et al.* achieves the best mean IoU and mean Chamfer distance while providing the reconstructions with the lowest number of components. However, the reconstructions of Labatut *et al.* are the only ones with a significant number of non-manifold edges.

2.6.3 Learning- and optimization-based surface reconstruction from synthetic MVS point clouds (E7)

To directly compare learning- and optimization-based reconstructions on the same dataset, we also reconstruct the Berger *et al.* shapes from synthetic MVS scans (cf. E4) with the optimization-based methods. Thus, for learning-based methods, we use the models trained on synthetic MVS scans from ShapeNet (cf. E4) and we optimize non-learning methods per shape using standard settings. We show the numerical results in Table 2.7 and visualisations in Figure 2.6. The learning-based methods DGNN and POCO benefit from the training on point clouds with the same characteristics as in the

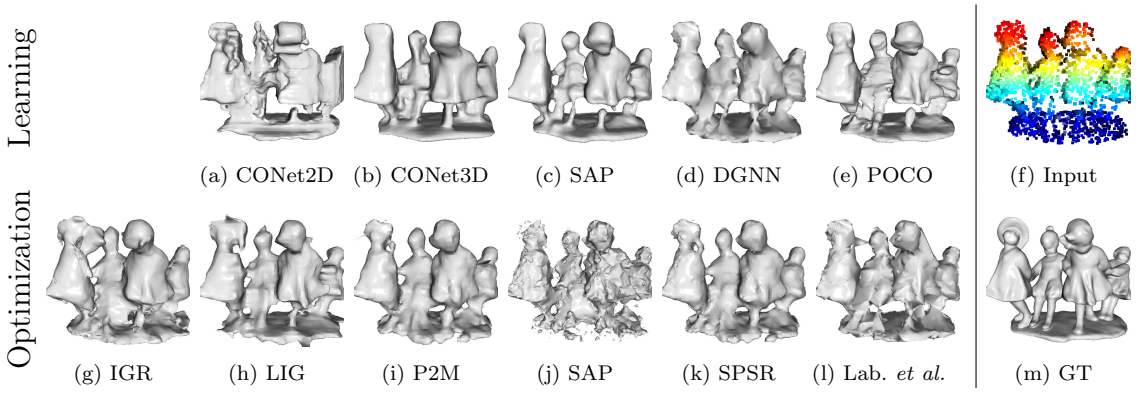


Figure 2.7: Learning- and optimization-based reconstructions: We show reconstructions of *dancing children* from Berger *et al.* from a low resolution input point cloud with noise (f) and the ground truth shape (m). The methods used in the first row (a - e) were trained on ShapeNet, on point clouds with the same sampling as the one used here. In the second row (g - k) we show the reconstructions for the same shape (from the same sampling) from optimization-based methods.

test set and reconstruct more truthful surfaces than the optimization-based methods.

Similar to E6, Labatut *et al.* produces the best results among the optimization-based methods.

2.6.4 Learning- and optimization-based surface reconstruction from real point clouds (E8)

Finally, we reconstruct surfaces from real MVS and range scanning point clouds. Again, for learning-based methods, we use the models trained on synthetic MVS scans from ShapeNet (cf. E4) and we optimize non-learning methods per point cloud. We show the reconstructions in Figure 2.8. The MVS point cloud from Middlebury (Figure 2.8a) is contaminated with a large amount of varying noise. SAP is the only learning method which reconstructs a smooth surface without missing details (Figure 2.8d). However, it suffers from small amounts of topological noise in the form of holes. The optimization-based method P2M provides a visually good reconstruction with few defects (Figure 2.8i). In Figures 2.8m and 2.8y, optimization-based methods handle the additional domain shift to an open scene better compared to learning-based methods. The two traditional methods SPSR and Labatut *et al.* provide the visually best results on average.

This experiment also shows that our findings on synthetic point clouds coincide with those on real-world point clouds, validating our experimental setup.

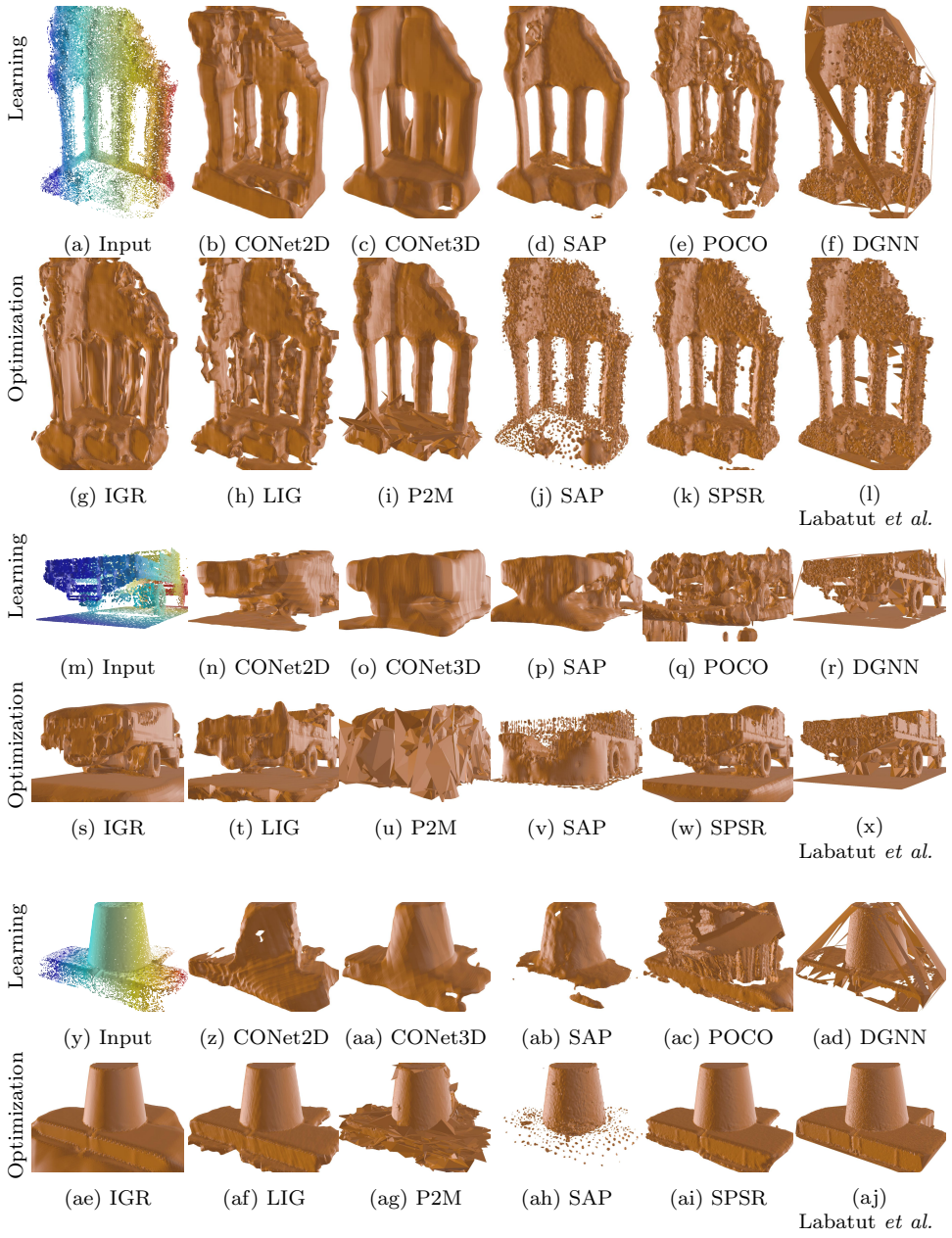


Figure 2.8: Learning- and optimization-based reconstructions of real-world point clouds (E8): Learning methods are trained on synthetic MVS scans from ShapeNet. Optimization-based methods are optimized per shape using standard settings. The traditional methods SPSR [65] and Labatut *et al.* [71] provide visually the best reconstructions with noticeable defects only the noisy *Temple Ring* MVS point cloud ((k) and (l)).

Table 2.8: Runtimes for learning-based reconstruction: Times (in seconds) for reconstructing one object from a point cloud of 3,000 points averaged over the ShapeNet test set. GC stand for Graph-cut; SE stands for surface extraction, such as marching cubes or triangle-from-tetrahedron. Note that different variants and implementations of marching cubes are used by different methods, which also influences the runtimes.

Model	Feature extraction	Decoding/GC	SE	Total
ConvONet2D [92]	0.016	0.32	0.17	0.51
ConvONet3D [92]	0.008	0.21	0.17	0.40
SAP [91]	0.022	0.017	0.047	**0.088**
DGNN [109]	0.11	0.28	0.01	0.39
POCO [21]	0.088	13.72	0.33	15.74
P2S [45]	69.06		11.51	80.57
SPSR [65]				1.25
Labatut *et al.* [71]	0.1	0.07	0.01	0.18

2.6.5 Runtimes

On Table 2.8, we report detailed runtimes for the methods tested in the learning-based experiments. SAP is the fastest of all reconstruction methods. DGNN also shows fast runtimes, while POCO is slow, due to its extensive use of neighborhood sampling. We also compare runtimes of P2S. We were not able to include this method in experiments E1 to E5 due to its long runtime for training and inference.

2.6.6 Summary and analysis

In the right circumstances, learning-based methods can produce highly detailed surfaces while remaining robust to noise and missing data. However, this requires training on large sets (30k shapes in our experiments) of sufficiently complex surfaces and associated point clouds. Even if the tested learning methods can generalize to unseen shape categories to some extent, the training and test sets must share the same point cloud characteristics. This suggests that these methods mainly learn priors related to the acquisition characteristics of the input point clouds, and less on the shapes themselves. However, learning-based methods do not produce satisfying results when the training shapes are too simple, or when the point clouds include unknown defects, such as outliers (seeTable 2.9). Mixing traditional and learning-based methods, as in SAP or DGNN, results in higher robustness to domain shifts and leads to short reconstruction times. Except for IGR, the tested novel optimization-based methods are not robust to acquisition defects and they rarely provide better results compared to the two traditional methods SPSR and Labatut *et al.*.

Table 2.9: Summary of benchmark results: Each method is rated with one to three stars per attribute, determined by the qualitative and quantitative results of our benchmark.

		Robustness to out-of-distribution				**Mesh quality**			**Runtime**
Learning		noise	outliers	density	category	watertightness	manifoldness	compactness	w/o training
ConvONet2D	[92]	*	*	*	*	***	***	*	***
ConvONet3D	[92]	*	*	*	*	***	***	***	***
SAP	[91]	**	**	**	**	***	***	*	***
DGNN	[109]	*	*	*	***	***	**	***	***
POCO	[21]	**	*	*	**	***	***	*	**
		Robustness to				**Mesh quality**			**Runtime**
Optimization		noise	outliers	low density	category	watertightness	manifoldness	compactness	w/ training
IGR	[48]	***	**	**	***	***	**	*	*
LIG	[62]	*	*	*	***	**	***	***	*
P2M	[52]	*	**	*	***	***	***	***	*
SAP	[91]	*	*	*	***	***	***	*	*
SPSR	[65]	***	***	***	***	***	***	*	***
Labatut *et al.*	[71]	***	***	***	***	***	*	***	***

2.7 Conclusion

Surface reconstruction from point clouds is a well studied subject in the field of digital geometry processing. However, constant developments in acquisition techniques and novel ideas for surface reconstruction and analysis bring forward new challenges. In this paper, we survey the field of surface reconstruction from point clouds and benchmark several related methods. We revisit traditional test-of-time approaches for surface reconstruction and detail how they inspired novel approaches. We evaluate traditional and novel optimization and learning-based methods on various tasks and datasets. We show that novel optimization-based methods are not as robust against defects as traditional methods. For in-distribution point clouds with characteristics similar to the ones of the training set, learning methods provide more accurate reconstructions than traditional approaches. However, real-world scenes often include a multitude of different and highly complex objects, and their acquisitions may contain a variety of defects. Most learning methods require shapes of similar complexity in training and test sets and they are not robust to out-of-distribution acquisition defects. These limitations of learning-based methods hinder the reconstruction of point clouds in the wild. Generating or finding adequate training data that includes a large variety of complex shapes scanned with realistic defects is a difficult task. Future work in learning-based surface reconstruction should focus on training on point clouds with realistic acquisition defects, *e.g.* from common sensors and acquisition settings, or on increasing the methods' robustness to unseen defects.

We will address the limited generalization capability of current DSR methods in the next chapter, where we show that training learning-based methods on point clouds with visibility information substantially increases their generalization capability to unseen

domains. We show that the added visibility information allows to reconstruct complex objects and scenes from real-world scans even when the methods are trained on simple training shapes.

In Chapter 4, we also show how DGNN can be trained on point clouds with multiple defects and densities at the same time. This allows to produce superior results compared to traditional test-of-time methods, even when reconstructing complex outdoor scenes from highly defect-laden MVS point clouds.

"What you see is what you get!"
Geraldine Jones

3

Deep Surface Reconstruction from Point Clouds with Visibility Information

Most current neural networks for reconstructing surfaces from point clouds ignore sensor poses and only operate on point locations. Sensor visibility, however, holds meaningful information regarding space occupancy and surface orientation. In this chapter, we present two simple ways to augment raw point clouds with visibility information, so it can directly be leveraged by surface reconstruction networks with minimal adaptation. Our proposed modifications consistently improve the accuracy of generated surfaces as well as the generalization capability of the networks to unseen domains.

3.1 Introduction

The problem of reconstructing a watertight surface from a point cloud has recently been addressed by a variety of deep learning based methods. Compared to traditional approaches, deep surface reconstruction (DSR) can learn shape priors [79, 89] and leverage shape similarities [52] to complete missing parts [41], filter outliers, or smoothen noise in defect-laden point clouds. DSR methods, however, often derive priors from training datasets with few shape classes, generalizing poorly to unseen categories or datasets. Learning more local priors improves consistency across different objects or scenes [62, 109] but may result in higher sensitivity to noise or other defects. Besides, lack of global context complicates surface orientation.

For real world point clouds, usually acquired via active or passive methods such as LiDAR scanning or MVS, the sensor position can be known and used to relate each observed point with a line-of-sight. Such visibility information can then help to orient surface normals [98] or predict occupancy [59, 71, 114]. While visibility is key for MVS, it has largely been ignored by DSR methods. In fact, sensor positions are usually not given in reconstruction benchmarks from point clouds.

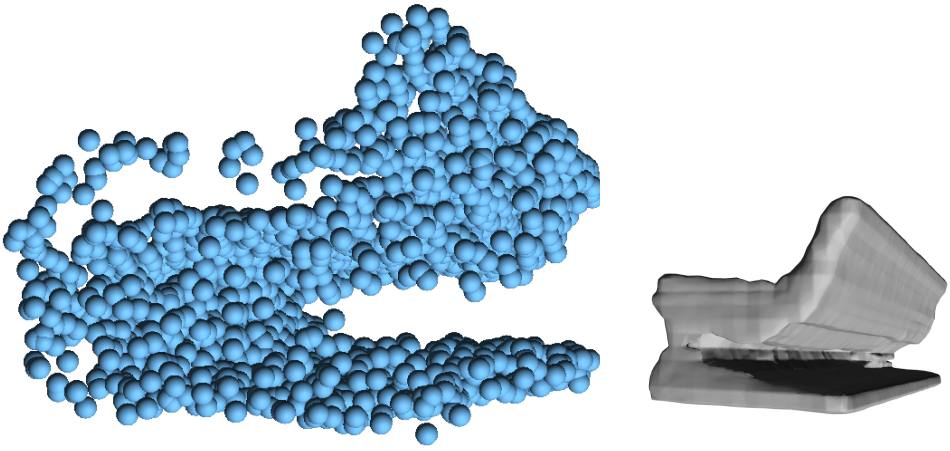
To remedy this, (i) we consider real data containing visibility information and, (ii) for synthetic shape benchmarks, we use a virtual scanning procedure to pair 3D points with the position of their sensor. We then, show that many DSR methods can easily be adapted to benefit from visibility information (cf. Figure 3.1). Our main contributions are as follows:

- We propose two simple ways to add visibility information to 3D point clouds, and we detail how to adapt DSR methods to utilize them, with very little changes.
- Using synthetic and real data, at object and scene level, we show for a wide range of state-of-the-art DSR methods that models leveraging visibility reconstruct higher-quality surfaces and are more robust to domain shifts.

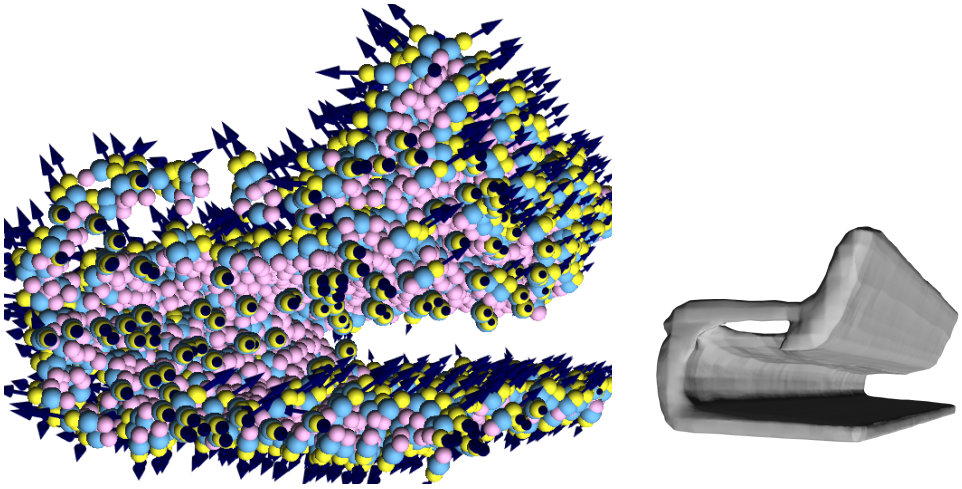
3.2 Related work

Many traditional surface reconstruction methods use visibility information [25, 26, 59, 60, 71, 114, 122]. They are usually based on a 3DT, which is intersected with lines-of-sight to attribute visibility features to Delaunay cells. While such methods can scale to billions of points [27] and are robust to moderate levels of noise and outliers, they do not incorporate learned shape priors.

In contrast, recent DSR methods have shown to produce more accurate surfaces than traditional approaches for shape categories and point cloud defects encountered during training. Many DSR methods use an implicit surface representation, either based on occupancy [79], or on the distance to the surface, whether it is signed [31, 35, 48, 83, 89]



(a) Reconstruction using only the points position.



(b) Reconstruction with visibility augmented point cloud.

Figure 3.1: Surface reconstruction with visibility information: We augment each 3D point ● with a sightline unit vector → pointing towards the sensor observing it. Additionally, two auxiliary points are placed before ● and after ● the observed point along the sightline. This allows DSR networks, with very little modification, to reconstruct a significantly more accurate surface.

or unsigned [4, 5, 36, 121]. To integrate local information, different forms of convolutions are used, either on regular grids [37, 41, 62, 92, 110], directly on points [21, 112] or via an MLP instead [45]. Other methods rather use an explicit surface representation such as a mesh, which is deformed [52] or whose elements are classified [102, 109].

A key issue is to get a sense of surface point orientation, to choose between reconstructing a thin volume (two main opposite orientations) or a thicker one (one main orientation at void-matter interface). Some methods dismiss the orientation issue by requiring oriented normals as input [62, 65, 112, 117], albeit producing such normals is a challenging task in itself [70, 81, 98]. We show that oriented normals can be advantageously replaced by visibility information.

Only few deep-learning methods make use of visibility information, typically from multiple views with camera pose information. RayNet [90] aggregates features from pixels of different views that intersect in the same voxel, but it outputs a dense point cloud, not a watertight surface mesh. Neural radiance fields [84, 86, 87] somehow also model the free space between a point and its sensor. However, they often require a slow optimization process and leverage little or no shape priors. We argue that DGNN [109] (Chapter 4), that classifies Delaunay cells with a graph neural network, currently is the only general DSR method from point clouds with visibility. However, DGNN relies on handcrafted visibility features, while in this chapter, we propose to augment the input point clouds without crafting explicit priors. For point clouds for which visibility information is not available, Vis2Mesh [106] shows that rendering virtual views can significantly improve the reconstruction quality of a traditional method.

3.3 Method

We consider a 3D point cloud $\mathcal{P}$ where each point $p \in \mathcal{P}$ has some coordinates $x_p \in \mathbb{R}^3$ and knows the position $c_p \in \mathbb{R}^3$ of a sensor observing it. Instead of only using the raw point coordinates $(x_p)_{p\in\mathcal{P}}$ as the input $(I_p)_{p\in\mathcal{P}}$ of a DSR network, we propose two simple ways to augment point cloud $\mathcal{P}$ with visibility information, and adapt DSR methods accordingly.

3.3.1 Sightline Vector (SV)

For each $p \in \mathcal{P}$, we define a unit vector $\mathbf{v}_p$ pointing from the observation x_p to the sensor c_p:

$$\mathbf{v}_p = (c_p - x_p)/\|c_p - x_p\|. \tag{3.1}$$

This contains useful information for surface orientation. We normalize the vector as the distance to the sensor is not as relevant as the viewing angle.

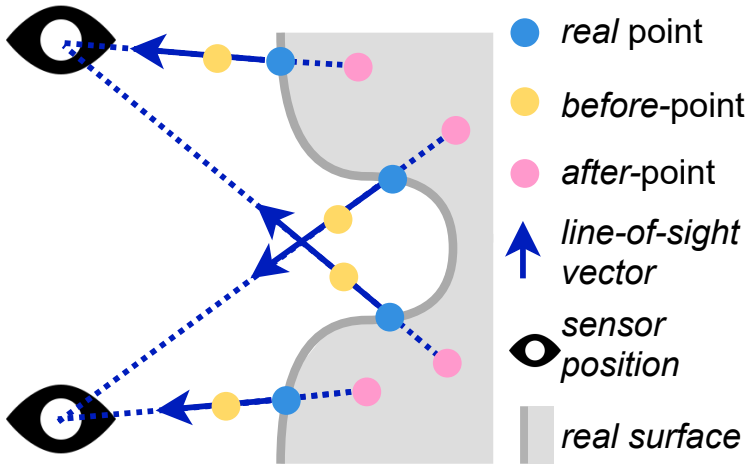


Figure 3.2: Visibility-augmented point cloud: Each observed point is associated to a sightline unit vector pointing towards its sensor. Two new points before and after each point are added. They help to disambiguate occupancy.

3.3.2 Auxiliary Points (AP)

To help the network predict empty and full space immediately in front of and behind the observed surface, we consider two auxiliary points to each point p: a *before-point* p_b and an *after-point* p_a, located along the sightline on each side of p:

$$x_{p_\mathrm{b}} = x_p + d\mathbf{v}_p, \tag{3.2}$$

$$x_{p_\mathrm{a}} = x_p - d\mathbf{v}_p, \tag{3.3}$$

where d is a characteristic distance in the point cloud $\mathcal{P}$, *e.g.*, the average distance from a point to its nearest neighbor. By construction, p_b is likely outside the scanned object or scene (modulo sensing noise and outliers), and p_a, likely inside (modulo object thickness too).

3.3.3 Visibility-Augmented Point Cloud

We use sightline vectors and auxiliary points to add visibility information to an input point cloud, separately or together.

(SV) To use sightline information only, we simply concatenate the sightline vector channelwise to the point coordinates to form the network input: $I_p = (x_p \oplus \mathbf{v}_p) \in \mathbb{R}^6$.

(AP) To use auxiliary points only, we add before-points p_b and after-points p_a to $\mathcal{P}$, with tags $\mathbf{t} \in \mathbb{R}^2$ concatenated to point coordinates to identify the point type, *i.e.*,

$I_q = (x_q \oplus \mathbf{t}_q) \in \mathbb{R}^5$ with $q \in \{p, p_\mathrm{b}, p_\mathrm{a}\}$, where $\mathbf{t}_p = [0\ 0]$ (observed point), $\mathbf{t}_{p_\mathrm{b}} = [1\ 0]$ (before-point), or $\mathbf{t}_{p_\mathrm{a}} = [0\ 1]$ (after-point).

(SV+AP) When combining both kinds of visibility information, before-points p_b and after-points p_a are given the same sightline vector as their reference point, *i.e.*, $\mathbf{v}_{p_\mathrm{b}} = \mathbf{v}_{p_\mathrm{a}} = \mathbf{v}_p$, and we take as input $I_p = (x_p \oplus \mathbf{v}_p \oplus \mathbf{t}_p) \in \mathbb{R}^8$.

While holding a similar kind of information, no augmentation can be reduced to the other one. SVs alone are not enough to place APs without knowing d, and APs alone, as they are not associated to their observed point in $\mathcal{P}$, cannot determine SVs (cf. Figure 3.2). This is also confirmed empirically in Section 3.4.4.

3.3.4 Modifying an Existing Architecture

We can adapt most DSR networks to handle visibility-augmented point clouds with only few modifications:

- We change the input size (number of channels) of the first layer of the network (generally an encoder), increasing it by 2, 3 or 5, depending on the augmentation respectively (AP, SV, SV+AP).
- We directly add auxiliary points to the point cloud, thus tripling the number of input points. For methods based on neighboring point sampling, we add auxiliary points after sampling for more efficiency.

The batch size may need to be adjusted to fit a larger point cloud in memory, but the rest of the network stays unchanged. Its size is mostly unaltered (*e.g.*, +0.005% for ConvONet [92]).

3.4 Experiments

To assess our proposal, we first describe how we generate synthetic point clouds with visibility information, and the real-world datasets we use. Then, we detail our simple adaptation of six different DSR baseline networks to leverage our visibility information, compare the quality of the reconstructed surfaces and analyze the generalization capability of the networks trained on point clouds with and without visibility information.

3.4.1 Datasets

We consider a variety of object and scene datasets, both synthetic and real, to show the versatility of our approach. For generating point clouds from artificial shapes we use our synthetic MVS scanning procedure described in Section 2.5.

ModelNet10. We use the official train/test splits of all 10 object classes of ModelNet10 [118] and hold out 10% of the train set for validation. We make the models watertight using ManifoldPlus [56]. We then scan the models from 10 different sensor positions to produce 3,000 points per object and add Gaussian noise with zero mean and standard deviation 0.005 as in [92].

ShapeNet. We study the generalizability of models trained on ModelNet10 by testing on 1000 shapes per class from the ShapeNet [32] test set of Choy *et al.* [38] (9 out of 13 classes are not represented in ModelNet10). We use the watertight models provided by the authors of Occupancy Networks [79] and scan the models using the same scanning procedure as for ModelNet10. We apply a transformation to the models (and scans) to match their orientation to the orientation of the ModelNet10 objects (except for networks marked with † in Table IV, which were trained with the original orientation).

Synthetic Room. We use the train/val/test splits of Synthetic Rooms and the provided watertight scenes [92]. For scanning, we only place sensors in the upper hemispheres. We scan 10 000 points and add Gaussian noise with zero mean and standard deviation 0.005 as in [92].

SceneNet. We test on a few synthetic scenes of SceneNet [51] using the given virtual scans, voxel-decimated to 1 cm.

ScanNet. We test on a few real scenes of ScanNet [40] using the provided real RGB-D scans, voxel-decimated to 2 cm.

Tanks and Temples. We use the real LiDAR point cloud of the *Ignatius* statue from the Tanks and Temples dataset [68] downsampled to 10,000 points.

Middlebury. We use an MVS point cloud of the *TempleRing* from Middlebury [100], made with OpenMVS [30] and downsampled to 10,000 points.

DTU. We use an MVS point cloud of *scan1* from DTU [61], made with OpenMVS [30] and downsampled to 10,000 points.

3.4.2 Evaluation Metrics

We use the geometric evaluation metrics presented in Section 2.5.4: volumetric intersection over union (IoU), mean Chamfer distance $\times 100$ (CD) and normal consistency (NC).

3.4.3 DSR Baselines

ConvONet [92]. This method first extracts point features and projects them on three 2D grids, or one 3D grid (variant). 2D or 3D grid convolutions then create features capturing local occupancy. Last, the occupancy of a query-point is estimated after interpolating grid features. We consider the 3×64^2 2D-plane encoder and the 64^3 3D-volume variant. To adapt them, we change the input size of the point encoder's first layer.

Points2Surf [45]. This method predicts both the occupancy of a query point and its unsigned distance to the surface. It uses both a local query-point neighborhood sampling and a global point-cloud sampling. We use the best-performing variant (uniform global sampling, no spatial transformer). To adapt it, we increase the input size of the first layer of both the local and global encoders, and when a point is sampled, locally or globally, we add its two auxiliary points on the fly.

Shape As Points [91]. For each input point, the method estimates its normal as well as k point offsets that are used to correct and densify the point cloud. The resulting point cloud of size $k|\mathcal{P}|$ is then fed to a differentiable Poisson solver [65]. To adapt the method, we change the input size of the first layer of the encoder, and of the normal and offset decoders as they also input the point cloud. We directly add auxiliary points as input, whose normal and offsets will thus be computed too.

Local Implicit Grids (LIG) [62]. This method trains an auto-encoder from dense point cloud patches. For inference, a given sparse patch with oriented normals is first augmented, close to our idea, with 10 new points along each normal; then reconstruction uses latent vectors minimizing a decoder-based training loss, and a post-processing removes falsely-enclosed volumes. As training code is unavailable, we use the model pretrained on ShapeNet (without noise). For oriented normals, we use Jets [29] oriented with a minimum spanning tree [98], as in [121]. To exploit visibility, we replace normals with sightline vectors; we do not add (more) auxiliary points.

POCO [21]. This method extracts point features using point cloud convolution [19], then estimates the occupancy of a query point with a learning-based interpolation on nearest neighbors. To adapt it, we increase the input size of the first layer and add auxiliary points on the fly only in the first layer.

Delaunay-Graph Neural Network (DGNN) [109]. This method, introduced in Chapter 4, uses a graph neural network to estimate the occupancy of Delaunay cells in a point cloud tetrahedralization. A graph-cut-based optimization then reinforces global

Table 3.1: Ablation study: The vanilla model of ConvONet [92] trained and tested on ModelNet10 with different ways to add visibility or normal information.

Model	SV	AP	IoU ↑
ConvONet-2D (3×64^2) [92]			0.853
+ sightline vectors (SV) only	✓		0.871
+ auxiliairy points (AP) only		✓	0.881
+ both SV and AP	✓	✓	**0.889**
+ sensor position	c_p		0.870
+ unnormalized SV	$c_p - x_p$		0.870
+ estim. normals / estim. orientation	Jets [29] / MST [98]		0.853
+ estim. normals / sensor orientation	Jets [29] / sensor-base [98]		0.868
+ true normals	GT normals		0.879

consistency. The method, which already uses visibility, outperforms other traditional reconstruction methods that use visibility information. As it already exploits visibility, we do not alter it, but use it as baseline for comparison.

Hyperparameters. For all methods, unless otherwise stated, training and evaluation are unchanged; we keep the value of the hyperparameters used in the original papers. When marching cubes [76] are needed for surface extraction, we use a grid resolution of 128^3.

3.4.4 Ablation Study

To validate our design, we compare in Table 3.1 various ways to add visibility information to the vanilla model of ConvONet.

Independently, SVs and APs significantly improve performance (+1.8 and +2.8 IoU pts). A reason why APs are more profitable could be that the network is tailored for points, not points with sightline features. While SVs and APs capture a similar kind of information, they are, however, complementary: combining them is even more beneficial (+3.5 IoU pts). Our general interpretation is that SVs help to decide whether a locally "thin" point cloud is to be considered as a noisy scan of a single surface, or as a (less noisy) scan on both sides of a thin surface. They thus have an impact on local shape topology, which can bring a notable gain. Auxiliary points convey similar information, but also contribute more directly to refine the surface position. Replacing SVs by the sensor position or by the unnormalized point-sensor vector gives essentially the same performance than our unit vector. This can be explained by the fact that our scanning procedure does not introduce significant variation in terms of distance to the sensor.

Table 3.2: Object-Level Reconstruction: DSR methods trained and tested on ModelNet10, with and without sightline vectors (SV) or auxiliary points (AP). † Trained on ShapeNet.

Model	SV	AP	IoU ↑	CD ↓	NC ↑
ConvONet-2D [92]			0.853	0.618	0.934
ConvONet-2D [92]	✓		0.871	0.557	0.936
ConvONet-2D [92]	✓	✓	**0.889**	**0.508**	**0.944**
ConvONet-3D [92]			0.885	0.493	0.949
ConvONet-3D [92]	✓		0.911	0.424	0.956
ConvONet-3D [92]	✓	✓	**0.923**	**0.393**	**0.959**
Points2Surf [45]			0.842	0.590	0.890
Points2Surf [45]	✓		**0.859**	**0.544**	0.896
Points2Surf [45]	✓	✓	0.856	0.548	**0.897**
Shape As Points [91]			0.903	0.438	0.948
Shape As Points [91]	✓		0.907	0.430	0.950
Shape As Points [91]	✓	✓	**0.914**	**0.410**	**0.954**
POCO [21]			0.907	0.422	0.945
POCO [21]	✓		0.915	0.408	**0.950**
POCO [21]	✓	✓	**0.917**	**0.406**	**0.950**
† LIG [62]			–	0.974	0.849
† LIG [62]	✓		–	**0.880**	**0.882**
DGNN [109]	✓		0.866	0.543	0.884

Yet, for real world acquistions, with a larger range of sensor distances, normalizing the SV ensures more stability.

Adding SVs outperforms estimated normals [29] with estimated orientation [98], and even estimated normals with sensor-based orientation. While using ground-truth normals is slightly more beneficial than SVs, combining SV+AP yields the best overall performance, which highlights the richness of our visibility information.

We also experiment with adding more than two auxiliary points: (i) at distance $0.5d$ or $2d$, (ii) at the midpoint between sensor and point, or (iii) as grazing points, estimated by densely sampling the sightlines with auxiliary points and keeping the ones close to an input point. None of these strategies brought significant improvements over simply adding two points at distance d on both sides of the observed point.

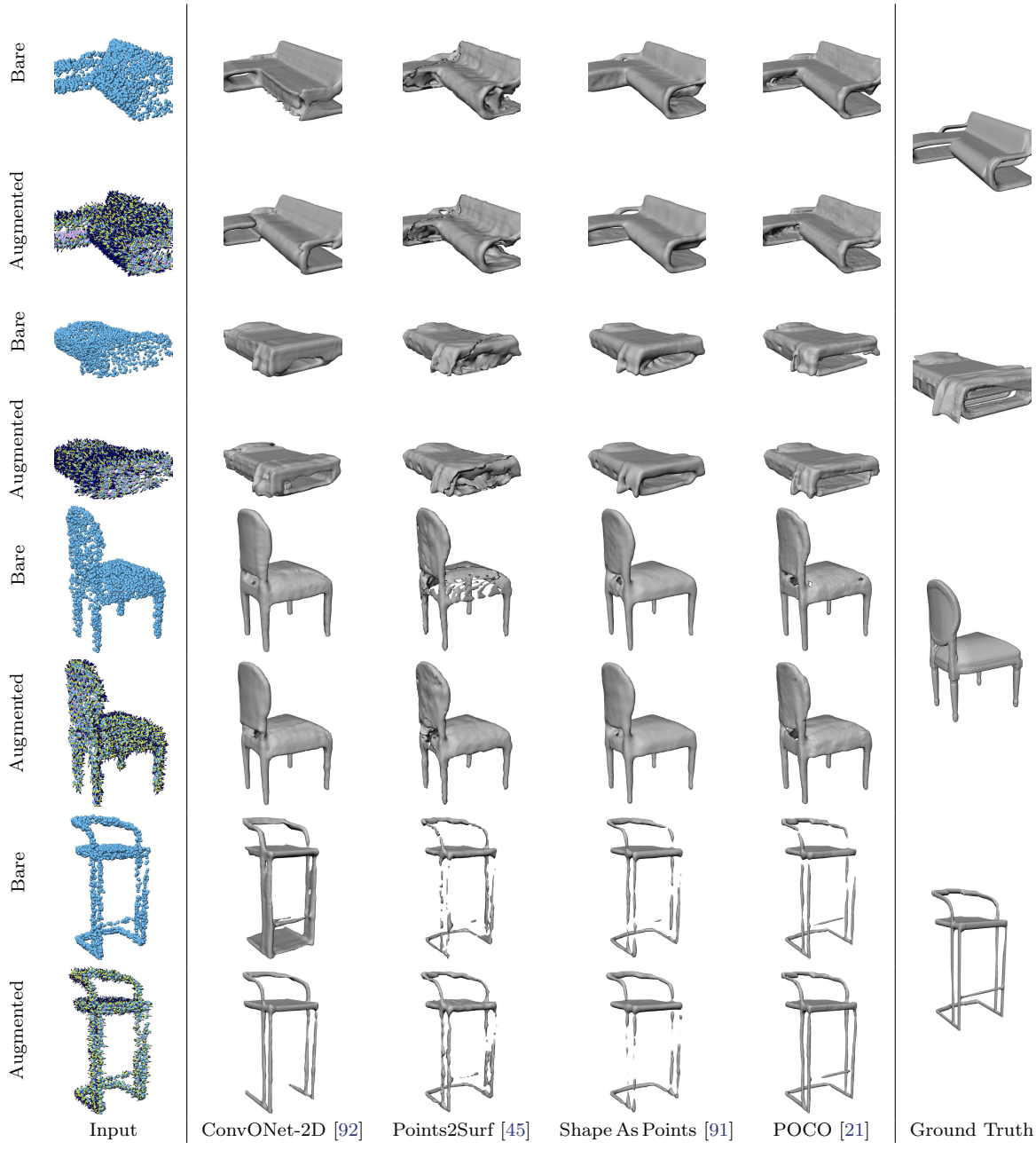


Figure 3.3: Object-level reconstruction on ModelNet10 I: Reconstructed shapes from the ModelNet10 test set using four different DSR methods trained on ModelNet10. Top rows of each object use the bare point cloud as input, and bottom rows use the point cloud augmented with visibility information.

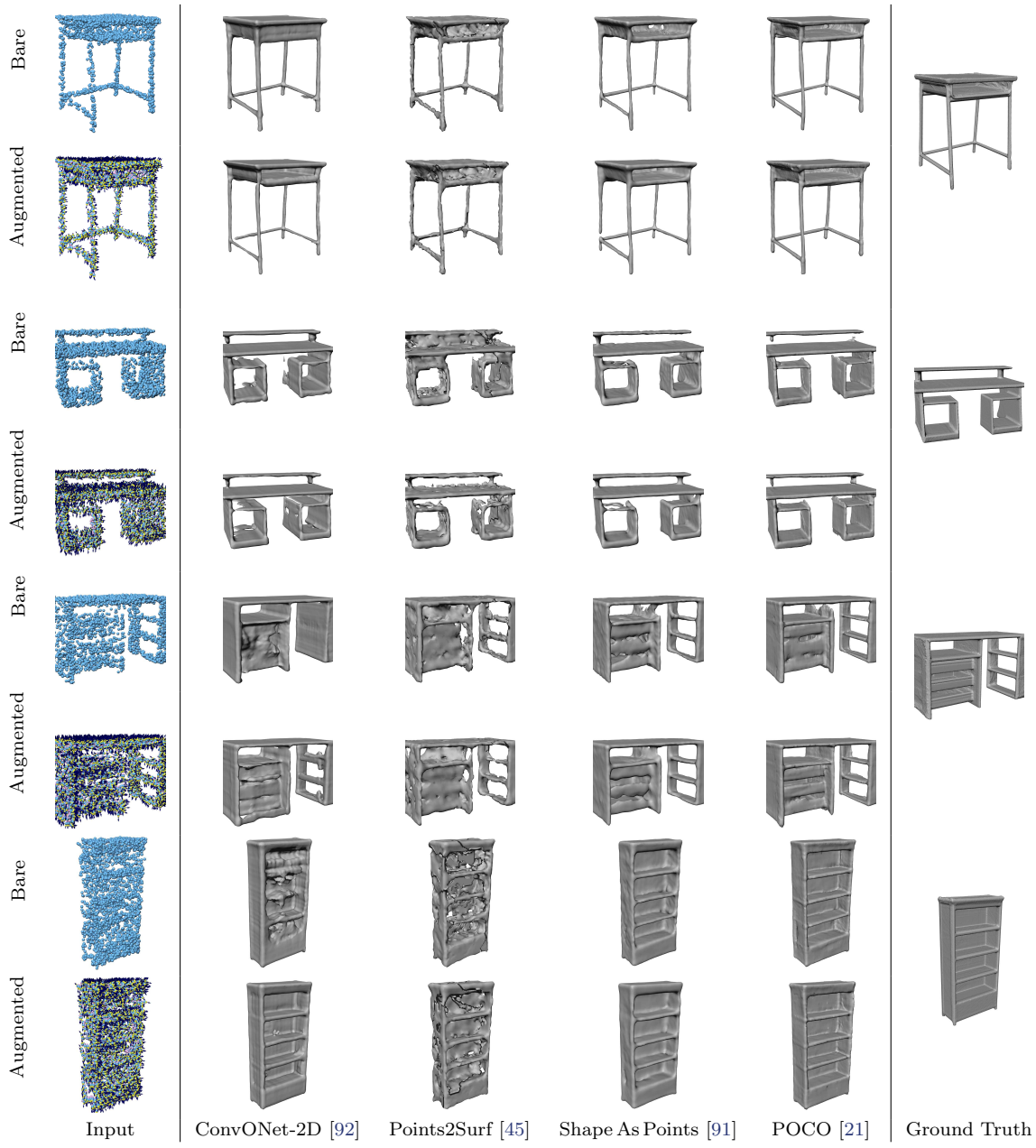


Figure 3.4: Object-level reconstruction on ModelNet10 II: Reconstructed shapes from the ModelNet10 test set using four different DSR methods trained on ModelNet10. Top rows of each object use the bare point cloud as input, and bottom rows use the point cloud augmented with visibility information.

Table 3.3: Numerical results for scene-level reconstruction: ConvONet trained and tested in sliding-window mode on Synthetic Rooms.

Model	SV	AP	IoU ↑	CD ↓	NC ↑
ConvONet-3D [92]			0.805	0.598	0.906
ConvONet-3D [92]	✓	✓	**0.832**	**0.569**	**0.911**

3.4.5 Object-Level Reconstruction

Table 3.2 reports the performance on ModelNet10 of various models, with and without sightline vectors or auxiliary points.

ConvONet, both planar and volumetric variants, gain about +3 IoU pts with visibility information. The resulting surface is more accurate, especially in concave parts, as illustrated in Figure 3.3 and Figure 3.4.

Points2Surf improves with sightline vectors, but auxiliary points do not improve further: the sensor vectors are enough to resolve ambiguities for the occupancy estimation, but distance estimation does not further benefit from auxiliary points.

Shape As Points benefits from sightline vectors, although not as much as other methods, probably because the model also estimates normals which provide a similar information as visibility. Still, adding auxiliary points further gains +0.6 IoU pts, yielding more complete and smoother surfaces.

POCO similarly benefits +1 IoU pts from sightline vectors but not much from the further addition of auxiliary points. While sightline vectors help for surface orientation, POCO is already accurate enough for APs to bring little refinement.

LIG produces poor results, likely because the only available model is trained on ShapeNet, with uniform sampling, little or no noise, and because oriented normals are only estimated. We cannot report IoU because LIG's post-processing creates holes in some objects. Yet, replacing the estimated normals by sightline vectors improves the predicted surface.

DGNN, which already exploits visibility and outperforms ConvONet-2D and Points2Surf, is outdistanced on this dataset by methods that use our augmented point clouds.

3.4.6 Scene-Level Reconstruction

To study the impact of visibility information at scene level, we train and test ConvONet on Synthetic Rooms, in sliding-window mode [92]. We report quantitative results in Table 3.3 and qualitative results in Figure 3.5. The model gains almost +3 IoU pts with visibility information, showing that benefits scale to scenes, not just to objects.

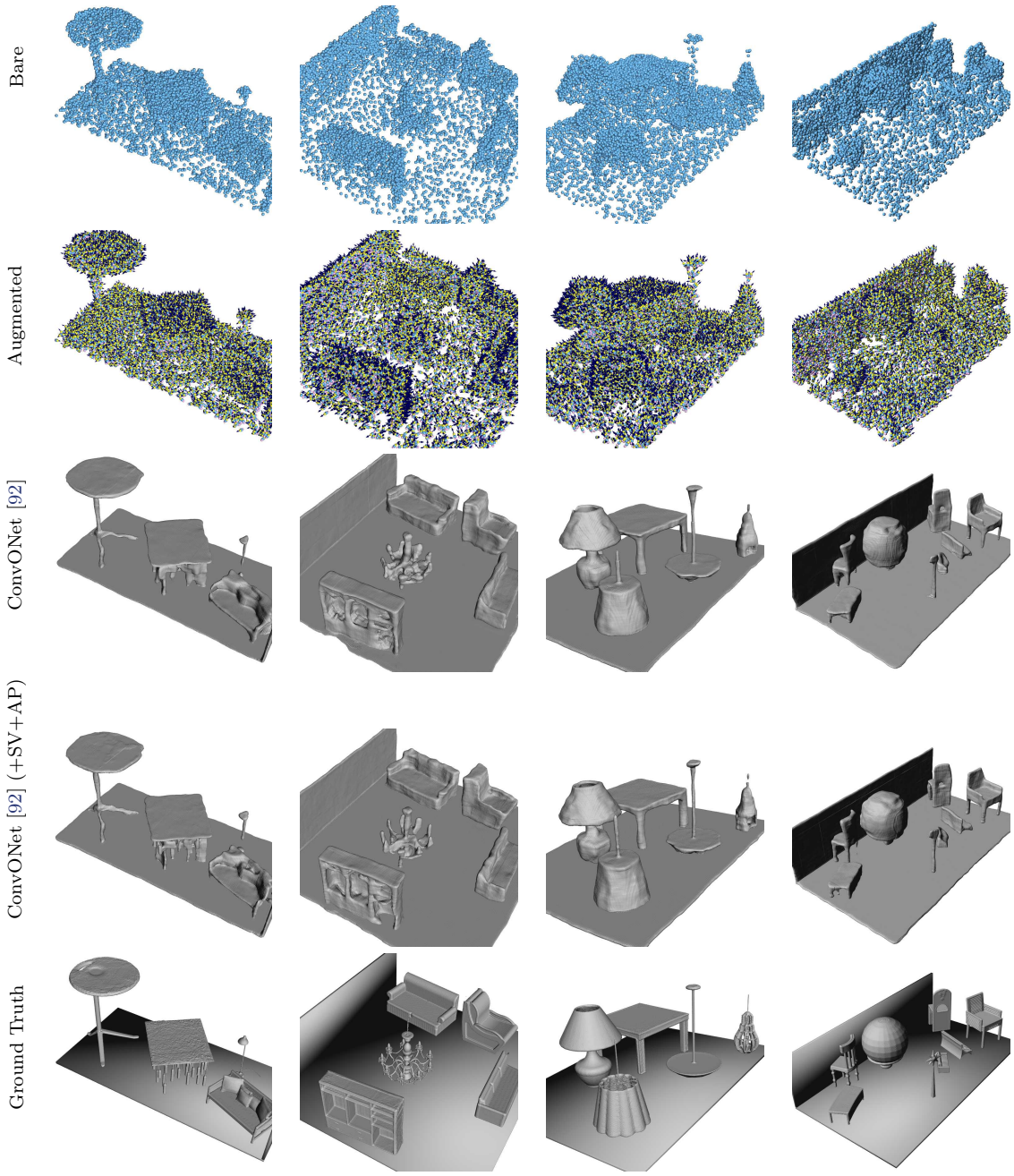


Figure 3.5: Scene-Level Reconstruction on Synthetic Rooms: Reconstructed scenes of the Synthetic Rooms dataset using ConvONet [92] in sliding-window mode, with and without visibility information.

Table 3.4: Out-of-Domain Object-Level Reconstruction: DSR methods trained on ModelNet10 and tested on ShapeNet, with and without sightline vectors (SV) or auxiliary points (AP). † Trained on ShapeNet.

Model	SV	AP	IoU ↑	CD ↓	NC ↑
† ConvONet-2D [92]			0.852	0.560	0.929
ConvONet-2D [92]			0.685	0.979	0.878
ConvONet-2D [92]	✓		0.667	1.042	0.833
ConvONet-2D [92]	✓	✓	**0.780**	**0.847**	**0.883**
ConvONet-3D [92]			0.628	0.972	0.885
ConvONet-3D [92]	✓		0.759	0.724	0.905
ConvONet-3D [92]	✓	✓	**0.823**	**0.685**	**0.912**
Points2Surf [45]			0.807	0.561	0.876
Points2Surf [45]	✓		**0.836**	**0.516**	0.886
Points2Surf [45]	✓	✓	0.833	0.522	**0.887**
† Shape As Points [91]			0.838	0.577	0.923
Shape As Points [91]			0.556	0.923	0.870
Shape As Points [91]	✓		0.749	0.843	0.881
Shape As Points [91]	✓	✓	**0.809**	**0.641**	**0.915**
POCO [21]			0.391	1.119	0.839
POCO [21]	✓		**0.832**	**0.618**	**0.901**
POCO [21]	✓	✓	0.815	0.635	0.887
DGNN [109]	✓		0.844	0.549	0.854

3.4.7 Generalization to New Domains

ShapeNet. To evaluate the impact of added visibility on the generalization capability of DSR methods, we train on ModelNet10 and test on ShapeNet (Table 3.4).

We observe that ConvONet, Shape As Points and POCO trained with visibility information generalize much better on the new objects and classes, with a gain up to +44 IoU pts. For comparison, we also show the scores of official models trained on ShapeNet, although trained on uniformly sampled points rather than virtual scans, which explains the drop of performance compared to the numbers in the papers [91, 92]. Points2Surf also improves by up to +3 IoU pts with added sightline vectors, but not further with APs.

We show the results of object-level reconstruction on ShapeNet in Figure 3.6 and Figure 3.7. All methods also visually benefit from added visibility information. In particular, ConvONet produces very accurate and complete surfaces of the unseen shape classes. The reason for the largely improved volumetric IoU when using visibility infor-

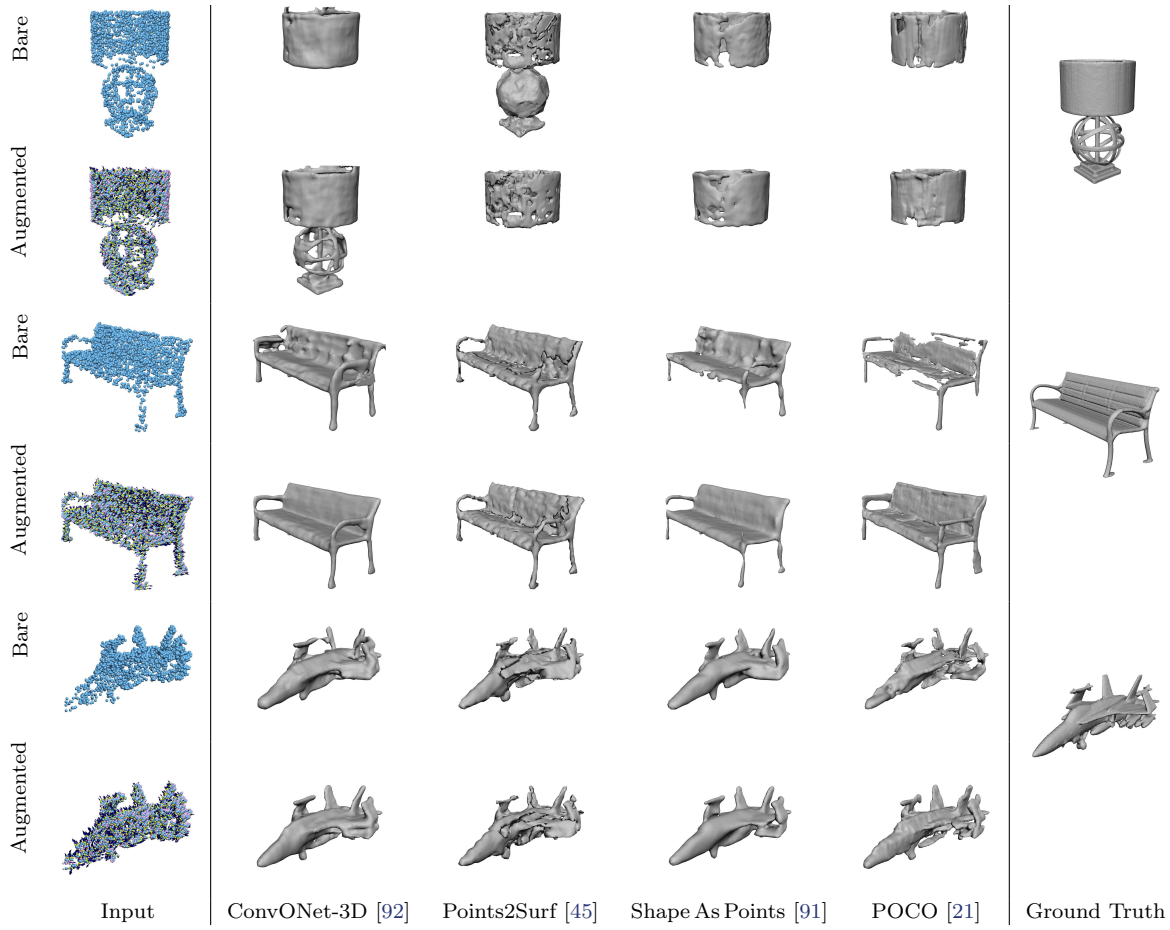


Figure 3.6: Out-of-domain object-level reconstruction on ShapeNet: Reconstructed shapes from the ShapeNet test set using four different DSR methods trained on ModelNet10. Top rows of each object use the bare point cloud as input, and bottom rows use the point cloud augmented with visibility information. The last two rows show a cut of the reconstructions that are shown on the two other rows immediately above.

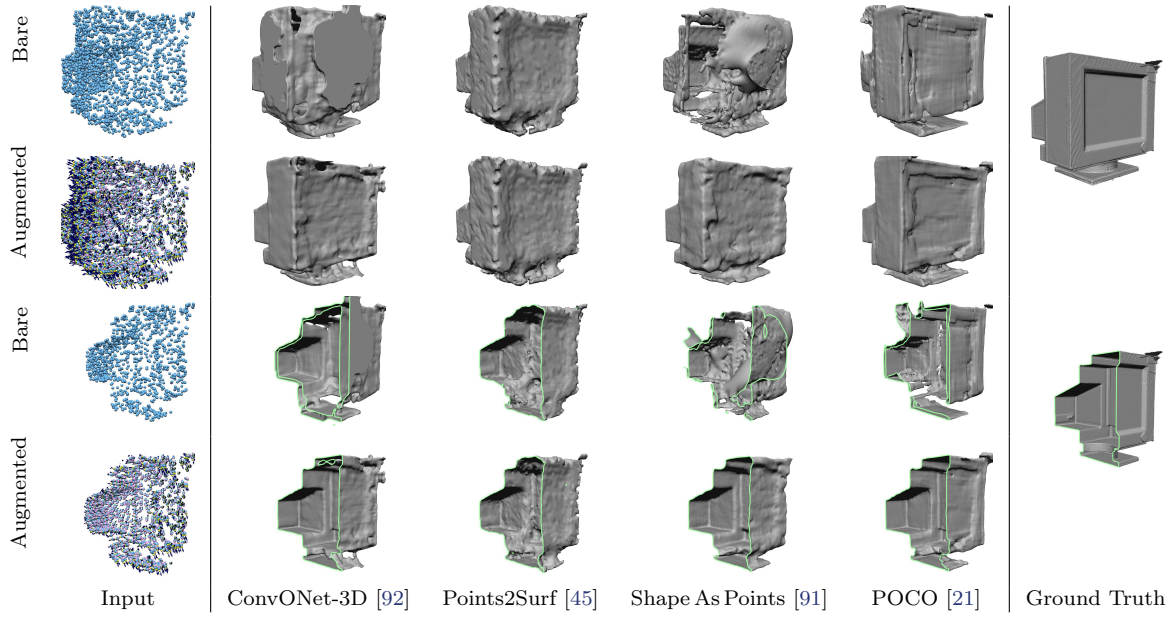


Figure 3.7: Cut of out-of-domain object-level reconstruction on ShapeNet: Reconstructed shapes from the ShapeNet test set using four different DSR methods trained on ModelNet10. Top rows of each object use the bare point cloud as input, and bottom rows use the point cloud augmented with visibility information. The last two rows show a cut of the reconstructions that are shown on the two other rows immediately above.

Table 3.5: Runtimes for object-level reconstruction with visibility information: Average times (in seconds) for reconstructing one object from a point cloud of 3 000 points with and without sightline vectors (SV) or auxiliary points (AP). MC is marching cubes. Times are averaged over the ModelNet10 test set.

Model	SV	AP	Encoding	Decoding	MC	Total
ConvONet-2D [92]			0.016	0.32	0.17	0.51
ConvONet-2D [92]	✓		0.016	0.34	0.17	0.54
ConvONet-2D [92]	✓	✓	0.016	0.33	0.17	0.52
Points2Surf [45]			69.06		11.51	80.57
Points2Surf [45]	✓		71.92		11.35	83.27
Points2Surf [45]	✓	✓	173.2		11.41	184.7
Shape As Points [91]			0.022	0.017	0.047	0.088
Shape As Points [91]	✓		0.023	0.017	0.046	0.086
Shape As Points [91]	✓	✓	0.024	0.041	0.047	0.114
POCO [21]			0.088	13.72	0.33	15.74
POCO [21]	✓		0.091	13.68	0.33	15.66
POCO [21]	✓	✓	0.093	13.70	0.33	15.67

mation is illustrated in Figure 3.7 and Figure 3.9. For out-of-domain reconstructions, the baseline methods often predict hollow shapes, i.e., empty space enclosed inside an object. This leads to backfaces behind the real surface and a poor volumetric IoU. On the contrary, our models, trained on visibility-augmented point clouds, learn to distinguish between empty and full space more reliably and do not produce such artifacts.

Real-World Data. The increased generalization capability of the models is also validated when reconstructing surfaces from real-world scans obtained with LiDAR or MVS. In Figures 3.10 and 3.8, we show that networks using visibility information can reconstruct more accurate and more complete surfaces. The results in Figure 3.10 and in the last row of Figure 3.8 represent open scenes, while all methods used for reconstruction were only trained on the closed ModelNet10 objects. Methods using our augmented point clouds with visibility information cope much better with this additional domain shift.

3.4.8 Runtimes

On Table 3.5, we report detailed runtimes for the tested methods, with and without visibility information.

Adding sightline vectors does not significantly increase the runtime for any of the tested methods. The effect of auxiliary points depends on the method. For ConvONet,

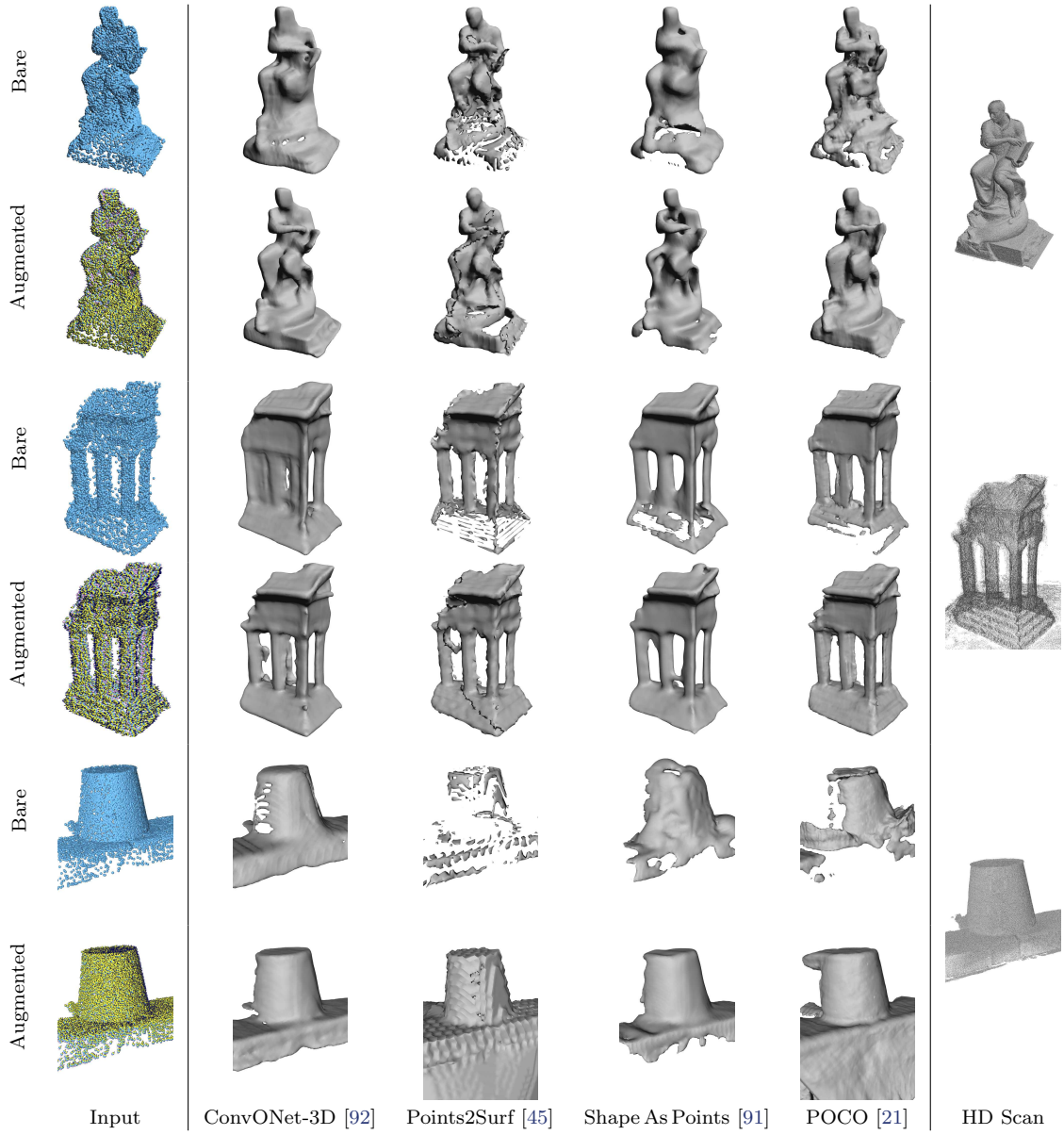


Figure 3.8: Out-of-domain object-level reconstruction from real-world scans: Reconstructed shapes from a LiDAR point cloud (top, *Ignatius* from Tanks And Temples [68]) and MVS point clouds (middle, *TempleRing* from Middlebury [100], bottom *scan1* from DTU [61]) using four different DSR methods trained on ModelNet10. Top rows of each object use the bare point cloud as input, and bottom rows use the point cloud augmented with visibility information. *HD Scan* is a high-density point cloud.

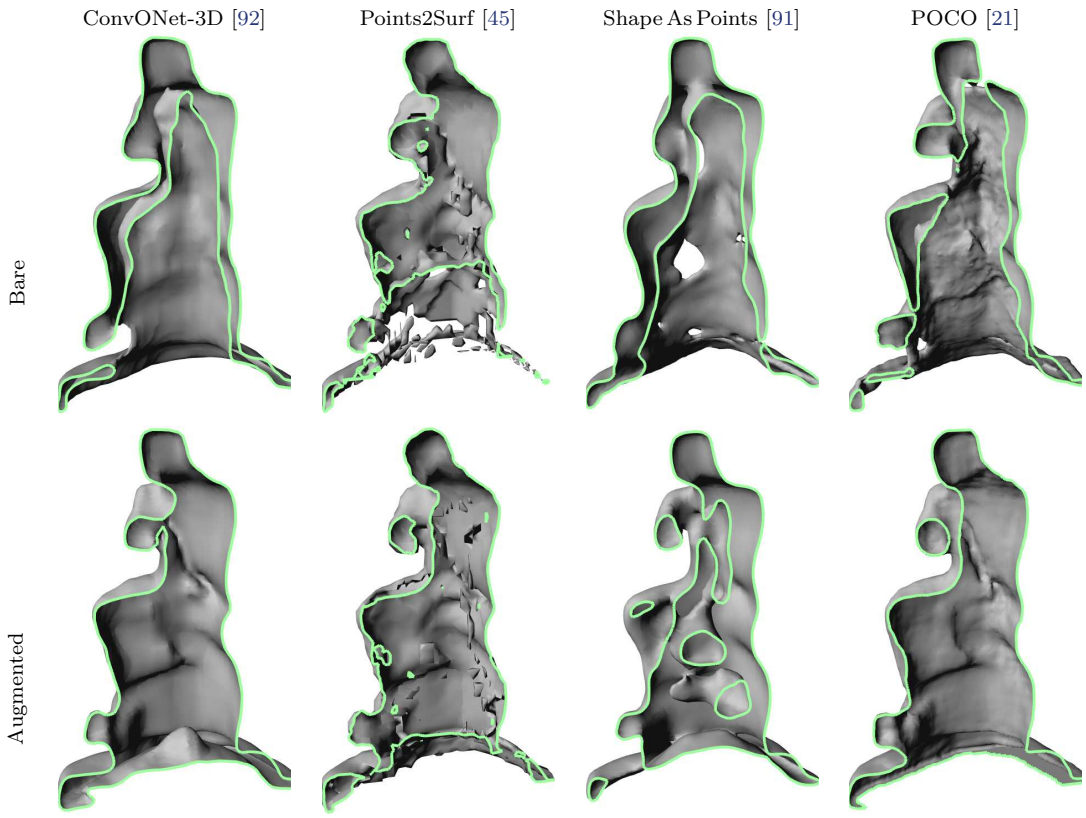


Figure 3.9: Cut of out-of-domain object reconstruction of $Ignatius$: A cut (along the green curve) of the reconstructed surface of *Ignatius*. The reconstructions from the bare point cloud (top row) include empty space enclosed inside the object with backfaces, leading to a poor volumetric IoU. The reconstructions from the point cloud augmented with visibility information (bottom row) include only one surface, close to the input points.

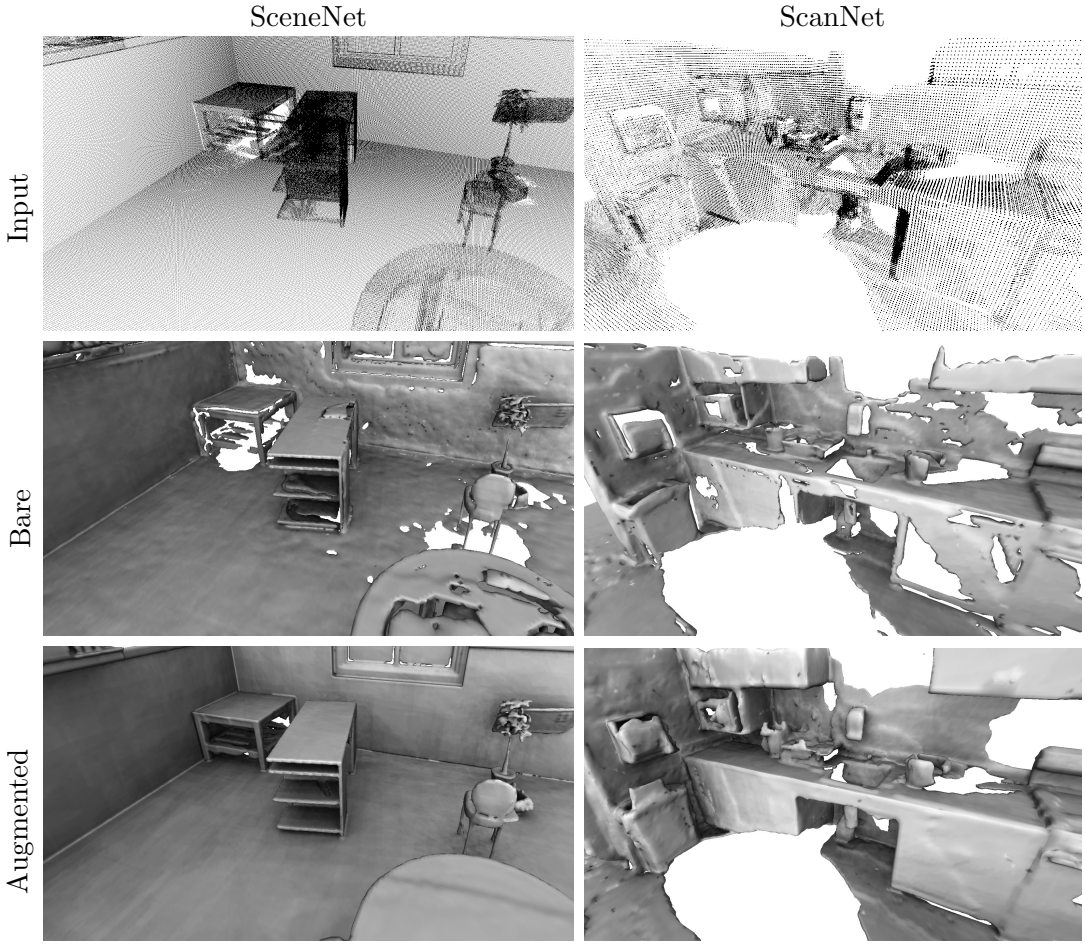


Figure 3.10: Out-of-domain scene-Level reconstruction on SceneNet and ScanNet: POCO [21] trained on ModelNet10, with and without visibility information, is run on scenes from SceneNet (synthetic RGB-D scan) and ScanNet (real RGB-D scan).

most of the processing time is spent computing grid features. The encoding of 3d points is performed by a small PointNet network [94], whose runtime is only a small fraction of the total time. As a consequence, adding APs does not incur significant changes in computation time ($< +2\%$). In contrast, Points2Surf uses a large point encoding network and is 2.2 times slower with APs. Shape As Points is 1.3 times slower due to the fact that we decode 3 times as many points as the baseline method. POCO is also essentially unaffected ($< +2\%$) by the addition of auxiliary points as they only impact the first (small) layer of the point-convolution backbone.

3.5 Limitations and perspectives

The position of auxiliary points depends on parameter d, which is the average distance, across the whole scene, from a point to its nearest neighbor. To better handle point density variations, it could be set locally rather than globally. Besides, as this positioning is also sensitive to sampling noise, d could also be directly adjusted after noise estimation.

Our current approach only associates each point with a single sensor, while MVS points typically have several. A more efficient and versatile approach than simply duplicating sightlines is still an open issue.

Last, we resort to virtual scans because current 3D reconstruction benchmarks do not provide sensor positions. While we show that using our augmented point clouds allows common architectures to successfully generalize from virtual to real scenes, our training set may fail to replicate some challenging configurations encountered using actual sensors.

3.6 Conclusion

The sensor poses are often ignored in point cloud processing, even though available with most acquisition technologies. For acquisitions with stationary sensors, only $3C$ 4 byte floats and P 2 byte unsigned integers are necessary to store sensor poses. For mobile acquisitions, where the sensor position varies for each point p, $3P$ floats are necessary. Image-based pipelines usually already contain sensor poses within the image orientation. We present two straightforward ways to exploit sensor positions in order to augment point clouds with visibility information. Our experiments show that various DSR methods can be adapted with minimal effort to exploit these visibility-augmented point clouds, resulting in improved accuracy and completeness of reconstructed surfaces, as well as a substantial increase in generalization capability.

"Bigger is always better!"

Adam Savage

4

Scalable Surface Reconstruction with Delaunay-Graph Neural Networks

In this chapter, we introduce a novel learning-based, visibility-aware, surface reconstruction method for large-scale, defect-laden point clouds. Our approach can cope with the scale and variety of point cloud defects encountered in real-life Multi-View Stereo (MVS) acquisitions. Our method relies on a 3D Delaunay tetrahedralisation whose cells are classified as inside or outside the surface by a graph neural network and an energy model solvable with a graph cut. We name this approach Delaunay-Graph Neural Network (DGNN). Our model, making use of both local geometric attributes and line-of-sight visibility information, is able to learn a visibility model from a small amount of synthetic training data and generalizes to real-life acquisitions.

4.1 Introduction

One of the most successful approaches for surface reconstruction from large point clouds in the wild is to (i) tessellate the convex hull of the point cloud using a 3DT, (ii) label the resulting cells as *inside* or *outside*, and (iii) extract the surface as the interface between cells with different labels [59, 71, 114]. This guarantees to produce non-self-intersecting and watertight surfaces, a useful requirement for downstream engineering applications. A remaining problem is that the methods typically rely on an energy formulation with handcrafted unary and binary potentials. Tuning the balance between data fidelity and regularity in these methods tends to be difficult due to the high variability in nature and amplitude of the defects of real-life point clouds.

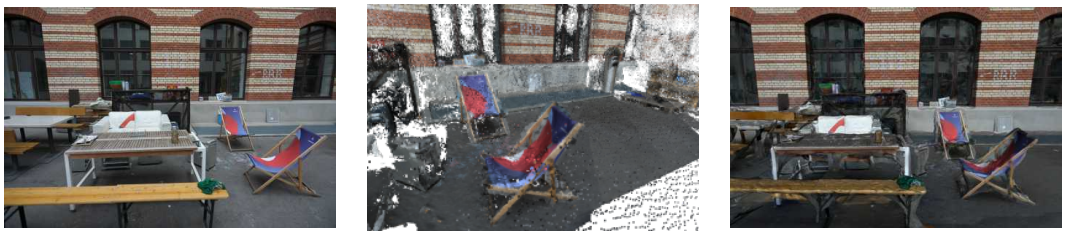
In the following chapter, we present a novel method for reconstructing watertight surfaces from large point clouds based on a 3DT whose cells are associated with a graph-adjacency structure, local geometric attributes, and visibility information derived from camera positions (see Figure 4.2). We then train a GNN to associate each cell with a probability of being inside or outside the reconstructed surface. In order to obtain a spatially regular cell labelling, these probabilities are incorporated into a global energy model that can be solved with a graph cut. This scheme directly predicts a spatially regular labeling, which leads to a smoother surface. Furthermore, graph-cut solving algorithms can easily scale to large point clouds, as opposed to other learning-based surface reconstruction methods, which tend to be limited to objects, or operate with sliding windows, as remarked by [92].

To the best of our knowledge, our method is the first deep learning-based mesh reconstruction algorithm able to take visibility information into account. This property is valuable, especially in areas lacking sufficiently dense input points. It is also the first deep learning surface reconstruction method using a memory-efficient GNN implementation built on a 3DT. We argue that combining the scalability of traditional computational geometry algorithms with the adaptability of modern deep learning approaches paves the way to learning-based large-scale 3D information processing. We validate our approach by showing that, even when trained on a small synthetic dataset, our method is able to generalize to large-scale, real-life, and complex 3D scenes and reach state-of-the-art performance on an open-access MVS dataset [99] (see Figure 4.1).

4.2 Related work

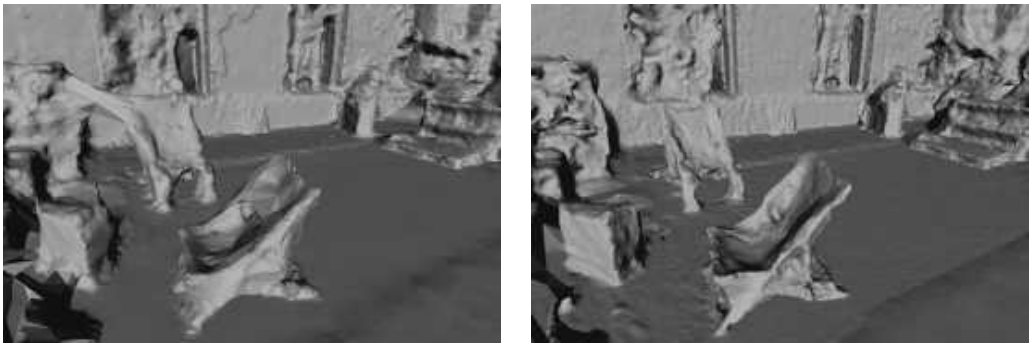
4.2.1 Graph Cut-Based Surface Reconstruction

Surface reconstruction based on inside-outside labelling and volumetric segmentation is traditionally formulated as a graph-cut optimization problem [25, 26, 59, 60, 114, 122]. The 3D space is discretized into the cells of a 3DT of captured points [71] or the cells of an arrangement of detected planes [33]. A graph $(\mathcal{T}, \mathcal{E})$ is formed for which the vertices



(a) Image from *courtyard* **(b)** Input MVS point cloud **(c)** Our reconstruction

(d) Screened Poisson [65] **(e)** Vu *et al.* [114]



(f) Jancosek *et al.* [60] **(g)** Ours

Figure 4.1: Scene-level reconstruction on ETH3D: Reconstruction of the *courtyard* scene of the ETH3D benchmark [99]. Top: a set of images, among which (a), is transformed into a dense MVS point cloud pictured in (b), from which our method reconstructs a mesh, displayed in (c) after texturation [115]. Bottom: untextured mesh reconstructions obtained by SPSR in (d), the algorithms of Vu *et al.* [114] in (e) and Jancosek *et al.* [60] in (f), and our proposed reconstruction in (g). Our method provides at the same time a higher accuracy (e.g., wall pattern in the background, that is reconstructed more truthfully) and a higher completeness (e.g., the back rest of the front chair).

are the cells $\mathcal{T}$ of the complex, and the edges in $\mathcal{E}$ connect cells with a common facet. Each cell $t \in \mathcal{T}$ is to be assigned with an occupancy label l_t in $\{0, 1\}$, where 0 means outside and 1 means inside. For this, each cell t is attributed a *unary potential* U_t expressing a likelihood of being inside or outside the scanned object. Additionally, each facet interfacing two adjacent cells s and t is attributed a *binary potential* $B_{s,t}$, which takes low values when the facet is likely to be part of a regular reconstructed surface and higher values otherwise. The label assignment of cells is performed by minimizing an energy in the following form:

$$E(l) = \sum_{t \in \mathcal{T}} U_t(l_t) + \lambda \sum_{(s,t) \in \mathcal{E}} B_{s,t}(l_s, l_t) \ , \tag{4.1}$$

where $\lambda \geq 0$ is the regularization strength. This energy E is globally minimized by computing a minimum cut in an appropriate flow graph or using a linear programming approach [20].

The unary potentials commonly depend on visibility criteria, such as: (i) cells with sensors are always outside, (ii) cells traversed by lines of sight (virtual lines between a sensor and an observed point) are likely outside, or (iii) cells *behind* a point are likely inside. These visibility models are not robust to the acquisition noise and outliers of real-life point clouds, so the unary potentials can be adjusted to the local point density [59, 60, 114, 122], or by using other modalities [25].

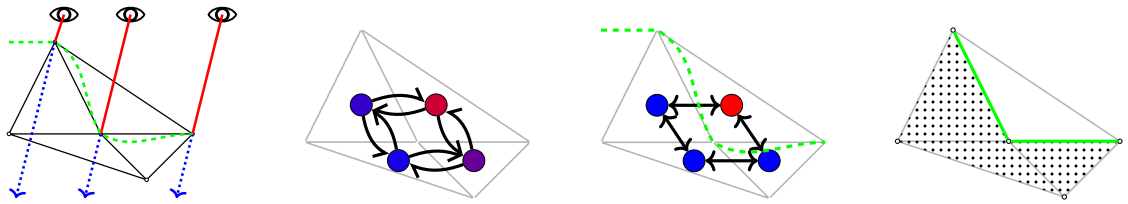
Binary potentials are used to force neighbouring cells crossed by the same line of sight to have the same labeling. Additionally, they can incorporate low-area [25, 26] or other shape-based priors [59, 71, 114]. Instead of hand-tuning the visibility model, we propose to learn it by training a neural network to produce unary potentials from local visibility and local geometric information.

4.2.2 Deep Learning-Based Surface Reconstruction

Recently, deep learning-based models have been proposed for reconstructing surfaces from point clouds or other modalities, operating on a discrete mesh or with continuous functions. We briefly discuss these approaches below, and refer the reader to the survey in Section 2.4 for a more recent and complete discussion of related works.

Surface-based approaches rely on transforming a discretized 2D surface, such as 2D patches or spheres [49, 74, 102, 119], meshes [42, 46, 52], charts [116], or learned primitives [43], in order to best fit an input point cloud. While such methods can lead to impressive visual results, they either cannot guarantee that the output mesh is watertight and intersection-free, or are limited to simple topologies and low resolution. Additionally, they are typically memory intensive, which prevents them from scaling to large scenes.

Volume-based approaches learn a continuous mapping from the input space $\mathbb{R}^3$ either to $\mathbb{R}$, defining the signed distance to the surface [4, 5, 31, 48, 89], or directly to an



(a) Delaunay triangulation and ray casting.

(b) Local and contextual learning with gnn.

(c) Global optimization with graph cuts.

(d) Surface reconstruction.

Figure 4.2: DGNN pipeline: (a) The input point cloud is triangulated, and visibility information is derived from lines of sight and from camera positions. (b) A graph neural network uses this local and contextual visibility information to predict an occupancy score for each tetrahedron. (c) A global energy derived from the network's output finds a minimal cut in an adapted flow graph. (d) The reconstructed surface is defined as the interface between cells with different (inside and outside) labels.

occupancy value $\{0, 1\}$ [79, 82, 92]. The network training can be either unsupervised [75], aided by geometric regularization [48], or supervised by ground-truth surface information [79, 82, 92]. Some continuous methods [79, 89, 92] predict the occupancy or signed distance conditionally to a latent shape representation, and thus learn a dataset-specific shape distribution. This can lead to difficulty in generalizing to shapes from unseen classes. Even though volume-based approaches define a surface in continuous space with implicit functions, they often rely on a discretization of 3D space to learn these functions [79, 82, 92]. Recent works propose to scale these methods to larger scenes using an octree structure [82] or a sliding window strategy [92].

While our method also relies on a discretization of space, our 3DT is directly computed from the input point cloud and is thus adaptive to the local resolution. Our method guarantees to produce watertight surfaces, can operate at large scale, and generalizes to unseen shapes and scenes.

4.3 Method

We explain here how to construct a 3DT augmented with expressive but lightweight visibility features that are leveraged by a memory-efficient GNN and used in a global energy formulation to extract the target surface.

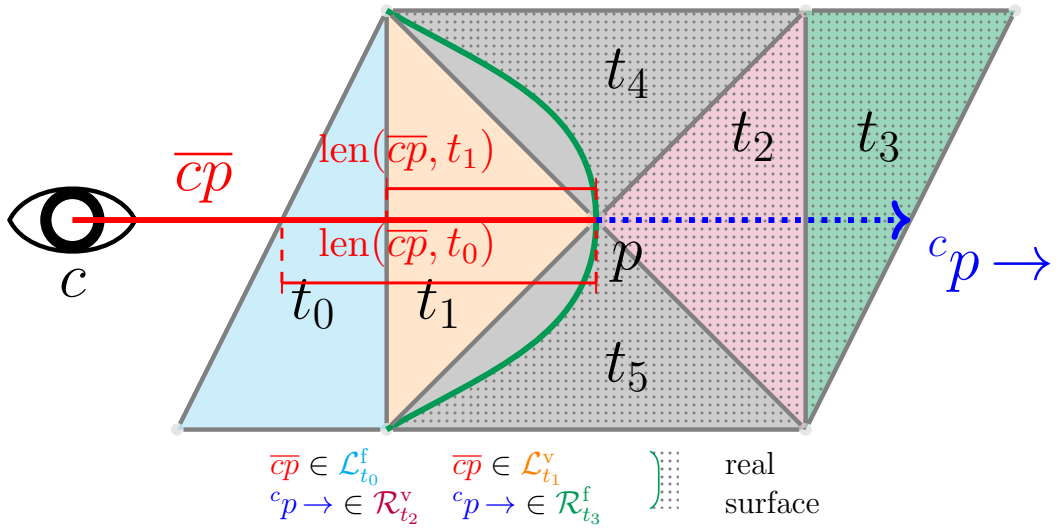


Figure 4.3: DGNN visibility features: A line-of-sight $\overline{cp}$ between a camera c and a visible 3D point p also defines a ray ${}^{c}p\rightarrow$. The line-of-sight $\overline{cp}$ traverses the two outside tetrahedra t_0 and t_1, while the ray ${}^{c}p\rightarrow$ traverses the two inside tetrahedra t_2 and t_3. Neither $\overline{cp}$, nor ${}^{c}p\rightarrow$ traverse t_4 or t_5; they thus do not contribute to their visibility information.

4.3.1 Visibility-Augmented 3D Tetrahedralization

We consider $\mathcal{P} \in \mathbb{R}^{3\times P}$ a 3D point cloud defined by the absolute point positions in space, where P is the number of points p in the cloud. Furthermore, we consider $\mathcal{C} \in \mathbb{R}^{3\times C}$ the absolute positions of a set of sensors used to capture these points, where C is the number of sensors c. We first construct a 3DT tessellating the convex hull of $\mathcal{P}$ into a finite set of tetrahedra $\mathcal{T}$. Each tetrahedron t is characterized by its four vertices $\mathcal{V}_t \in \mathbb{R}^{3\times 4}$ and four facets $\mathcal{F}_t \in \mathbb{N}^{3\times 4}$. At the boundary of the convex hull, each facet is incident to an infinite cell whose fourth vertex is at infinity. This ensures that each facet of the 3DT is incident to exactly two tetrahedra.

Let $\mathcal{L} \subset \mathcal{C} \times \mathcal{P}$ be the *lines-of-sight* from cameras c of $\mathcal{C}$ to points p of $\mathcal{P}$ seen from c. A *line-of-sight* $\overline{cp} \in \mathcal{L}$ is an oriented segment from c to p. These definitions are illustrated in Figure 4.3. In the case of MVS point clouds, a single point can be seen from multiple cameras. Similarly, we call ${}^{c}p\rightarrow \in \mathcal{R} \subset \mathcal{C} \times \mathcal{P}$ the *ray* extending line-of-sight $\overline{cp}$ from the seen point p to infinity. To simplify the computation of visibility information, we truncate the ray traversal after the second tetrahedron. For instance, in Figure 4.3, ${}^{c}p\rightarrow$ does not go beyond t_3.

4.3.2 Feature Extraction

The *occupancy*, or insideness, of a tetrahedron *w.r.t.* the target surface can be inferred by combining geometrical and visibility information. Indeed, a tetrahedron t traversed by a line-of-sight $\overline{cp}$ is *see-through*, and most likely lies outside the surface. Conversely, if a tetrahedron is traversed by a ray ${}^{c}p\rightarrow$ and no line-of-sight, it may lie inside the surface, especially if close to p.

However, visibility-based information is not sufficient to retrieve a perfect labelling of tetrahedra. First, there is no connection between the discretization of the space by the 3DT and the distribution of lines-of-sight. There may be a significant number of tetrahedra not traversed by any line-of-sight nor any ray, depending on the geometry of the acquisition. Second, noise and outliers — stemming from MVS for example — can result in inaccurate and unreliable visibility information. Thus, we propose to use a GNN to propagate and smooth visibility-based information, as well as other contextual information, to all tetrahedra in the 3DT of an object or a scene.

Tetrahedron features. While one could argue for directly learning features from tetrahedra and camera positions in an end-to-end fashion, this resulted in our experiments in a significant computational overhead and a very difficult geometric task for a neural network to learn. Instead, we propose to derive computationally light, yet expressive, handcrafted features encoding the local geometry and visibility information of tetrahedra.

For a tetrahedron $t \in \mathcal{T}$, we denote by $\mathcal{L}_t^{\mathrm{v}}$ the set of lines-of-sight that traverse t and that end at one of its vertices $v \in \mathcal{V}_t$, and by $\mathcal{L}_t^{\mathrm{f}}$ the set of lines-of-sight that intersect t through its facets and do not end at one of its vertices:

$$\mathcal{L}_t^{\mathrm{v}} = \{(c,p) \in \mathcal{L} \mid (\overline{cp}) \cap t \neq \varnothing,\ p \in \mathcal{V}_t\} \tag{4.2}$$

$$\mathcal{L}_t^{\mathrm{f}} = \{(c,p) \in \mathcal{L} \mid (\overline{cp}) \cap t \neq \varnothing,\ p \notin \mathcal{V}_t\}. \tag{4.3}$$

Likewise, we denote $\mathcal{R}_t^{\mathrm{v}}$ and $\mathcal{R}_t^{\mathrm{f}}$ the equivalent sets for rays in $\mathcal{R}$.

These sets are informative for determining the occupancy of a tetrahedron. Indeed, a tetrahedron t for which $\mathcal{R}_t^{\mathrm{v}}$ is nonempty indicates that it is directly *behind* an element of the surface, hinting at a higher probability of insideness. A nonempty $\mathcal{R}_t^{\mathrm{f}}$ indicates that t was hidden by a surface, hinting at a possible insideness. Indeed, since the *hit* occurred *before* t, this carries less confidence as it could be due to an occlusion or a thin structure.

Conversely, a tetrahedron t with nonempty $\mathcal{L}_t^{\mathrm{f}}$ indicates that it is traversed by a line-of-sight, indicating a high probability of outsideness. A nonempty $\mathcal{L}_t^{\mathrm{v}}$ also indicates that t is traversed by a line-of-sight, but since the *hit* is on one of the corners of the tetrahedron, this prediction is likely to be affected by acquisition noise, and hence has a lower confidence.

To characterize the influence of lines-of-sights and rays with respect to a given tetrahedron t, we define two measures: $\text{count}(t)$ and $\text{dist}(t)$. $\text{count}(t) \in \mathbb{N}^4$ corresponds to the number of each type of lines or rays intersecting with t:

$$\text{count}(t) = \left[|\mathcal{L}_t^{\text{v}}|, |\mathcal{L}_t^{\text{f}}|, |\mathcal{R}_t^{\text{v}}|, |\mathcal{R}_t^{\text{f}}| \right] \ . \tag{4.4}$$

Then, to measure the *proximity* between t and the impact point p of a traversing line-of-sight $\overline{cp}$, we define $\text{len}(\overline{cp}, t)$ as the distance between p and the exit point of $\overline{cp}$ in t seen as from p. As represented in Figure 4.3, this corresponds to the length of the longest segment between p and the portion of $\overline{cp}$ intersecting t:

$$\text{len}(\overline{cp}, t) = \max_{y \in (\overline{cp}) \cap t} \|p - y\| \ . \tag{4.5}$$

When $(\overline{cp}) \cap t$ is empty, $\text{len}(\overline{cp}, t)$ is set to zero. We define $\text{len}({}^c p \rightarrow, t)$ in the same manner for rays. Finally, $\text{dist}(t)$ characterizes the proximity of tetrahedron t with the observed points p of its intersecting lines-of-sight and rays:

$$\begin{aligned} \text{dist}(t) = \Bigg[& \min_{\overline{cp} \in \mathcal{L}_t^{\text{v}}} \text{len}(\overline{cp}, t), \min_{\overline{cp} \in \mathcal{L}_t^{\text{f}}} \text{len}(\overline{cp}, t), \\ & \min_{{}^c p \rightarrow \in \mathcal{R}_t^{\text{v}}} \text{len}({}^c p \rightarrow, t), \min_{{}^c p \rightarrow \in \mathcal{R}_t^{\text{v}}} \text{len}({}^c p \rightarrow, t) \Bigg] \ . \end{aligned} \tag{4.6}$$

We complement the 8 visibility features defined by $\text{count}(t)$ and $\text{dist}(t)$ with 4 morphological features: the volume of t, the length of its shortest and longest edges, and the radius of its circumsphere. This leads to a set of 12 handcrafted features f_t for each tetrahedron $t \in \mathcal{T}$, that we normalize (zero mean and unit standard deviation) independently.

It is important to note that none of the aforementioned features can be computed in a meaningful way for infinite cells of the 3DT. We simply set all feature values to zero, which can be interpreted as a padding strategy.

4.3.3 Contextual Learning

We learn contextual information with a GNN using the propagation scheme GraphSAGE of Hamilton *et al.* [50] with a depth of K (see Figure 4.4). This scheme can be performed independently for each tetrahedron, allowing us to perform inference on large graphs with limited memory requirements.

We denote by $G = (\mathcal{T}, \mathcal{E})$ the undirected graph whose edges $\mathcal{E} \subset \mathcal{T}^2$ link cells that are adjacent, i.e., share a facet. We consider one tetrahedron t in $\mathcal{T}$, and compute $\text{hop}(t, K)$ its K-hop neighborhood in G, i.e., the set of nodes s of $\mathcal{T}$ which can be linked to t using

at most K edges.We leverage the local context of a tetrahedron t with a message-passing scheme over its local neighborhood in G. We first initialize the features of all nodes s in the subgraph $\text{hop}(t, K)$ with the handcrafted features defined in Section 4.3.2: $x_s^0 = f_s$. We then apply the following update rule in two nested loops over $k = 0, \ldots, K-1$ and for all $s \in \text{hop}(t, K-1)$:

$$x_s^{k+1} = \sigma\left(\text{norm}\left(W^{(k)}\left[x_s^k \,\middle\|\, \underset{u \in \mathcal{N}(s)}{\text{mean}}\left(x_u^k\right)\right]\right)\right) \ , \tag{4.7}$$

with $\mathcal{N}(s)$ the one-hop neighborhood of node s, σ an activation layer, norm a normalization layer, and $[\cdot \,||\, \cdot]$ the concatenation operator. $\{W^{(k)}\}_{k=0}^{K-1}$ is a set of K learned matrices, each operating only at the k-th iteration. After K iterations, a multilayer perceptron (MLP) maps the embedding x_t^K to a vector of dimension 2 representing the inside/outside scores for tetrahedron t:

$$(i_t, o_t) = \text{MLP}(x_t^K) \ . \tag{4.8}$$

The main advantage of this simple scheme is that it can be performed *node-wise* from the K-hop neighborhood of each node and run the update scheme locally. Memory requirements only depend on K, i.e., subgraph extraction, and not on the size of the full graph G. This allows us to scale inference to large graphs. Likewise, training can be done by sampling subgraphs of depth at least K, and does not require to load large graphs in memory.

4.3.4 Loss Function

For defect-laden point clouds affected by noise, the ground-truth surface is generally not exactly aligned with the faces of the 3DT created from the input points. Consequently, tetrahedra intersecting the true surface can be only partially inside or outside, and cannot be attributed a *pure* 0 or 1 occupancy label. Instead, we define the ground-truth insideness/outsideness $i^{\text{g}} \in [0,1]^{|\mathcal{T}|}$ as the proportion of each cell's volume lying inside of a ground-truth closed object; i^{g} can take any value between 0 and 1.

We convert the tetrahedra's predicted inside/outside scores to an occupancy using the sotfmax fonction:

$$\hat{\imath}_t = \frac{\exp(i_t)}{\exp(i_t) + \exp(o_t)} \tag{4.9}$$

We define the fidelity loss for each t as the Kullback-Leibler divergence of the true occupancy i_t^{g} and the softmaxed predicted occupancy $\hat{\imath}_t$:

$$\text{KL}_t\left(\hat{\imath}_t\right) = i_t^{\text{g}} \log\left(\hat{\imath}_t\right) + \left(1 - i_t^{\text{g}}\right) \log\left(1 - \hat{\imath}_t\right) + q \ , \tag{4.10}$$

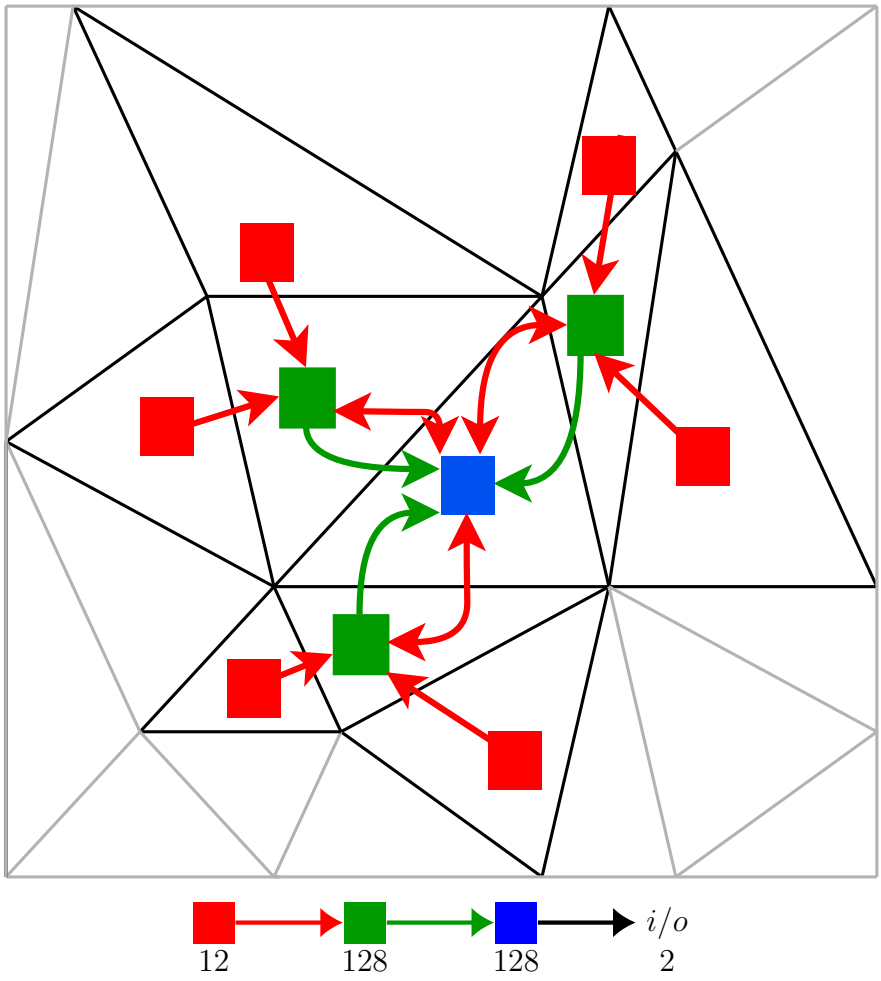


Figure 4.4: Graph neural network scheme: Illustration of our GNN update for tetrahedron ■. Information is pooled at different hops k (here $K = 2$) into an increasingly rich descriptor. A linear layer based on the last cell embedding assigns an inside/outside score to the central tetrahedron. Note that each prediction can be performed independently for each tetrahedron by considering only the K-hop subgraph.

with q a quantity that does not depend on $\hat{\imath}_t$, and can thus be ignored while training the network. We define the total loss as the average of all tetrahedra's fidelity weighted by their volume V_t:

$$L\left(\hat{\imath}\right) = \frac{1}{\sum_{t\in\mathcal{T}} V_t} \sum_{t\in\mathcal{T}} V_t \,\mathrm{KL}_t\left(\hat{\imath}_t\right) \ . \tag{4.11}$$

4.3.5 Global Formulation

Defining the target surface directly from the inside/outside scores predicted by the network can result in a jagged surface due to non-consistent labelling of neighboring tetrahedra. To achieve a smooth label assignment, even in areas with heavy noise, we use the inside/outside scores i_t, o_t to define the unary potentials in the formulation of Equation 4.1:

$$U(l_t) = i_t\,[l_t = 0] + o_t\,[l_t = 1] \ , \tag{4.12}$$

with $[x = y]$ the Iverson bracket, equal to 1 if $x = y$ and 0 otherwise. We also add a constant factor α_{vis} to o_t for tetrahedra containing a camera, indicating that they must lie outside the surface.

We define the binary potentials introduced in Equation 4.1 with a surface quality term that allows us to reconstruct a smooth surface and to efficiently remove isolated or non-manifold components in the final surface mesh. We use the same surface quality term

$$B_{s,t}(i_s, i_t) = \mathbb{1}(i_s \neq i_t)\,\beta_{s,t} \tag{4.13}$$

as Labatut *et al.* [71] for a facet interfacing the tetrahedra s and t. Considering the intersection of the circumspheres of s and t with the facet, with angles ϕ and ψ, then $\beta_{s,t}$ is defined as:

$$\beta_{s,t} = 1 - \min\{cos(\phi), cos(\psi)\} \ . \tag{4.14}$$

The energy $E(l)$ in Equation 4.1 with unary and binary potentials as defined above can be minimized efficiently by constructing a flow graph and using a min-cut solver [22].

4.3.6 Surface Extraction and Cleaning

We can define the target surface by considering the labeling of $\mathcal{T}$ obtained by minimizing $E(l)$. The reconstructed surface is composed of all triangles whose adjacent tetrahedra have different labels. Triangles are oriented towards the outside tetrahedra. For open

scene reconstruction, we optionally apply a standard mesh cleaning procedure, implemented in OpenMVS [30], by removing spurious and spike faces (whose edges are too long). This is especially useful for outdoor scenes containing areas with very little input data, such as far-away background or sky, and for which outliers can result in isolated components with low-quality surfaces. In our experiments, all competing methods, at least visually, benefit from this classic postprocessing for open scene reconstruction.

4.4 Experiments

In this section, we present the results of two sets of numerical experiments to show the performance of our reconstruction method for both objects and large-scale scenes. In both settings, our method is only trained on a small synthetic dataset, and yet outperforms state-of-the-art learning and non learning-based methods, highlighting its high generalization capability. A third experiment, where we trained DGNN on large object shape databases was presented in Section 2.6.

4.4.1 Evaluation Setting

Training Set. We train our network on a small subset of 10 shapes for each of the 13 classes of the ShapeNet subset from [38]. We found this small number to be sufficient for our network to learn diverse local shape configurations. We produce watertight meshes of these models using the method of Huang *et al.* [55]. We then synthetically scan the models with different degrees of outliers and noise and build corresponding 3DTs.

To obtain the ground-truth occupancy, we sample 100 points in each tetrahedron and determine the percentage of these sampled points lying inside their corresponding ground-truth models. In total, we train our network on around 10M tetrahedra.

Hyper-Parameters. We train our model by extracting batches of 128 subgraphs of depth $K = 4$ centered around random tetrahedra of our training set. We parameterize our model with $K = 4$ linear layers of width 64, 128, 256 and 256 for each hop respectively. After each linear layer, we apply batch normalization [58] and ReLU non-linearities. The final cell embeddings are mapped to inside/outside scores using an MLP $256 \to 64 \to 2$. We train the network with the Adam optimizer [67] with an initial learning rate of 10^{-4} which we decrease by a factor of 10 every 10 epochs. For the graph-cut optimization, we set the camera bias term α_{vis} to 100 and the regularization strength to $\lambda = 1$.

We use the same hyper-parameters for all dataset variants, in particular for all settings of the scanning procedure of Berger et al.'s benchmark [8]. While we could choose parameters that better fit specific noise and outlier levels, we argue that a single real-life

scene can present multiple defect configurations simultaneously. Consequently, reconstruction algorithms should be versatile enough to handle different noise and outlier ratios with a single parameterization.

Competing Methods. We compare our model with other mesh reconstruction methods that have available (or re-implementable) code, and the ability to scale to large scenes with several million points:

- **ConvONet** [92] is a deep model, like ours, but that does not take visibility into account. We use the model (with multi-plane decoder) pretrained on the entirety of ShapeNet for the object-level reconstruction. Among all the available pretrained models, this one gave the best performance. We use the volume decoder model pretrained on the synthetic indoor scene dataset [92] for scene-level reconstruction, where we set the voxel size to 4 cm.
- IGR [48] is a deep model which we optimize for $30,000$ iterations on each object using the official implementation.
- SPSR [65] is a classic non-learning-based method which approximates the surface as a level-set of an implicit function estimated from normal and point information. We chose an octree of depth 10 and Dirichlet boundary condition for object-level reconstruction and 15 and Neumann boundary condition for scene-level reconstruction. We also experimented with post-processing the Screened Poisson reconstruction with the included surface trimming tool, but could not find consistent trimming parameters that improve the mean values of Poisson presented in Table 4.2 and Table 4.3.
- **Labatut *et al.*** [71] is a graph-cut-based method for range scans that makes use of visibility information. We use our own implementation of the algorithm and use the parametrization suggested by the authors ($\alpha_{vis} = 32$ and $\lambda = 5$) and with σ set according to an estimation of the scan noise.
- **Vu *et al.*** [114] is an extension of Labatut *et al.* [71] to MVS data. We use its OpenMVS [30] implementation with *min-point-distance = 0.0*, *free-space-support = 0* and default parameters for all other settings.
- **Jancosek *et al.*** [60] also exploits visibility in a graph-cut formulation, with special attention to weakly-supported surfaces. We use the OpenMVS [30] implementation with *min-point-distance = 0.0*, *free-space-support = 1* and default parameters for all other settings.

For scene reconstruction, we compare all methods without mesh cleaning and with default clean (Section 4.3.6) options in OpenMVS.

Evaluation Metrics. We use the evaluation metrics presented in Section 2.5.4 for object-level reconstruction.

For scene-level reconstruction, we measure intrinsic mesh properties, such as the number of components and the surface area. We also extrinsicly evaluate the mesh reconstruction methods at a given precision τ. To this end, we uniformly sample random points from the reconstructed meshes. We then evaluate (i) accuracy, (ii) completeness, and (iii) F1-score of the reconstructions. Accuracy is defined as the fraction of sampled points on the reconstructed mesh within distance τ to any point in the ground truth. Completeness is defined as the fraction of ground-truth points for which there exists a point sampled on the reconstructed mesh within a distance τ. The harmonic mean of accuracy (precision) $P(\tau)$ and completeness (recall) $R(\tau)$ is the F1-Score $F(\tau)$ defined as:

$$F(\tau) = \frac{2P(\tau)R(\tau)}{P(\tau) + R(\tau)}. \tag{4.15}$$

We use the ETH3D Evaluation Program [99] to compute these values from the ground-truth LiDAR scans and samplings of the meshed surfaces. In the original benchmark, the authors evaluate MVS reconstructions with threshold τ as low as 1 cm. Generating such mesh samplings implies sampling over 300 million points for some scenes. To accelerate this procedure, we only sample 900 points per m^2 on the reconstructed meshes. This allows us to compute accuracy and completeness with a threshold of 5 cm and up. The protocol accounts for incomplete ground truth by segmenting the evaluation space into *occupied*, *free* and *unobserved* regions and sampled (reconstruction) points in the unobserved space are later ignored.

4.4.2 Design Choices and Ablation Study

In this section, we evaluate the effect of several of our design choices on the performance of our algorithm.

Direct Prediction. We assessed the impact of the graph cut step by evaluating the quality of the surface obtained using only the unary terms: tetrahedrons with an insideness over 0.5 are predicted as inside, and the others outside. This leads to very fragmented reconstructed surfaces (over 10 times more components), especially in the background of the scenes. Given that our objective is to produce compact watertight surfaces, we chose to use a regularization, here with a global energy minimization.

Learning Binary Weights. We designed an GNN able to predict binary weights in the energy model along the unaries. However, this lead to more fragmented surfaces and overall lower performance. The difficulties of learning the potentials of an energy

Table 4.1: Ablation Study. Mean IoU of different design choices of our method. We train all models on a subset consisting of 1100 shapes of ModelNet10 and test on ModelNet10 and ShapeNet.

Model	**ModelNet10**	**ShapeNet**
Vanilla model	86.1	84.1
Feature ablation		
No geometric	-1.4	-1.3
No visibility	-4.5	-6.4
No vertex	-0.2	-0.2
No facet	-0.3	-0.4
Model ablation		
No edge convolution	-0.3	-0.5
No volume weighting	+0.4	0.0
No graph-cut	+0.1	-0.3
Receptive field		
3-hop	-0.3	-0.3
4-hop	0.0	0.0
5-hop	+0.1	+0.1

model with a neural network are expected, as neural networks operate locally and in continuous space, while graph cuts operate globally and in discrete space. In fact, we can interpret our GNN prediction as the marginal posterior inside/outside probability of each tetrahedron, while the graph cut provides an inside/outside labeling of maximum posterior likelihood (MAP) in a fitting Potts model [23]. These two tasks being conceptually different, we were not able to successfully learn our surface reconstruction in an end-to-end fashion and leave this endeavor for future work.

Using Facet Features With Edge Conditioned Convolutions. We tried replacing our GNN scheme with the Dynamic Edge Conditioned Convolution of Simonovsky *et al.* [103] for its ability to leverage facet features derived from both ray and tetrahedrons. This resulted in a marginal increase in performance (under 1% decrease of the Chamfer distance) at the cost of an increase in computational and memory requirements. For the sake of simplicity and with scalability in mind we keep the simple GraphSAGE scheme.

Relevance of Geometric Features. We tried training a model using only visibility features and no tetrahedron-level geometric features. In doing this ablation, we lose between 10% of F1-Score on ETH3D. This demonstrates that visibility informa-

tion should be combined with geometric information, which is not typically done in traditional approaches.

4.4.3 Object-Level Reconstruction

Experimental Setting. To evaluate the adaptability of our method to a wide range of acquisition settings, we use the surface reconstruction benchmark of Berger *et al.* [8]. It includes five different shapes with challenging characteristics such as a non-trivial topology or details of various feature sizes. The provided benchmark software allows to model a variety of range scanning settings to produce shape acquisitions with different defect configurations. We apply to each shape different settings such as varying resolution, noise level, and outlier ratio, meant to reproduce the variety of defects encountered in real-life scans. See Section 2.5 for details about the scanning procedure.

Results. The results are presented in Table 4.2 and illustrated Figures 4.5 to 4.9. We observe that the competing learning-based methods have a hard time with this dataset. ConvONet [92] does not generalize well from the simple models of ShapeNet to the more challenging objects evaluated here. As for IGR [48], it works well in the absence of noise and outliers but produces heavy artifacts on defect-laden point clouds. In contrast, our method is able to generalize to the new unseen shapes and significantly outperforms ConvONet and IGR. Our method also outperforms the state-of-the-art and highly specialized algorithm of Labatut *et al.* [71], showing that the graph neural network is able to learn a powerful visibility model with a higher accuracy than methods based only on handcrafted features.

4.4.4 Large-Scale Scene Reconstruction

Experimental Setting. To evaluate the ability of our method to scale to entire scenes, we experiment with the high-resolution MVS benchmark ETH3D [99]. This benchmark is originally designed to evaluate MVS algorithms (point cloud reconstruction) under challenging real-life conditions. Ground-truth point clouds and camera poses are openly available for a training set including 7 indoor and 6 outdoor scenes. The ground truth consists of LiDAR scans post-processed to only contain reliable points.

While we cannot train our network on this dataset due to the lack of closed surfaces in the ground truth, we can evaluate the quality of the output of our algorithm after sampling points on the reconstructed surface. To this end, we generate dense point clouds from downsampled images (3100×2050 pixels) of the 13 training scenes using a patch-based MVS algorithm [6] implemented in OpenMVS [30]. The point clouds and associated camera poses are used as inputs for all mesh reconstruction methods evaluated in Section 4.4.1. Additionally, as input for the Screened Poisson and IGR

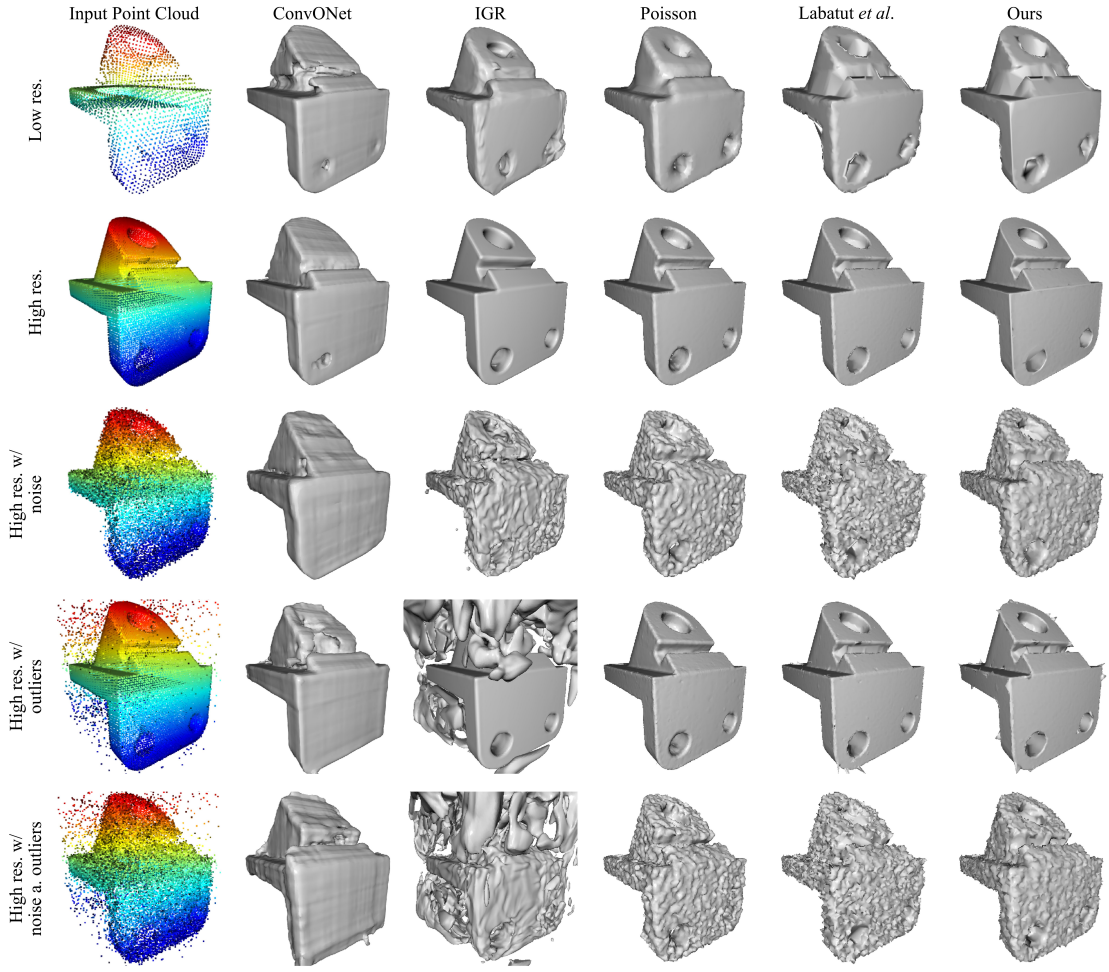


Figure 4.5: Qualitative results on Berger et al.'s anchor: We show the input point clouds in column 1. ConvONet [92] (column 2) does not generalize well to the unseen new shape. IGR [48] (column 3) works well at high resolution but fails in the other cases. The Screened Poisson [65] algorithm (column 4) does not reconstruct the sharp features well, but is robust against outliers, even close to the surface. The reconstructions of Labatut *et al.* [71] (column 5) and ours (column 6) are visually similar for the easier high resolution case. Our method performs slightly better on the low resolution, and noise cases.

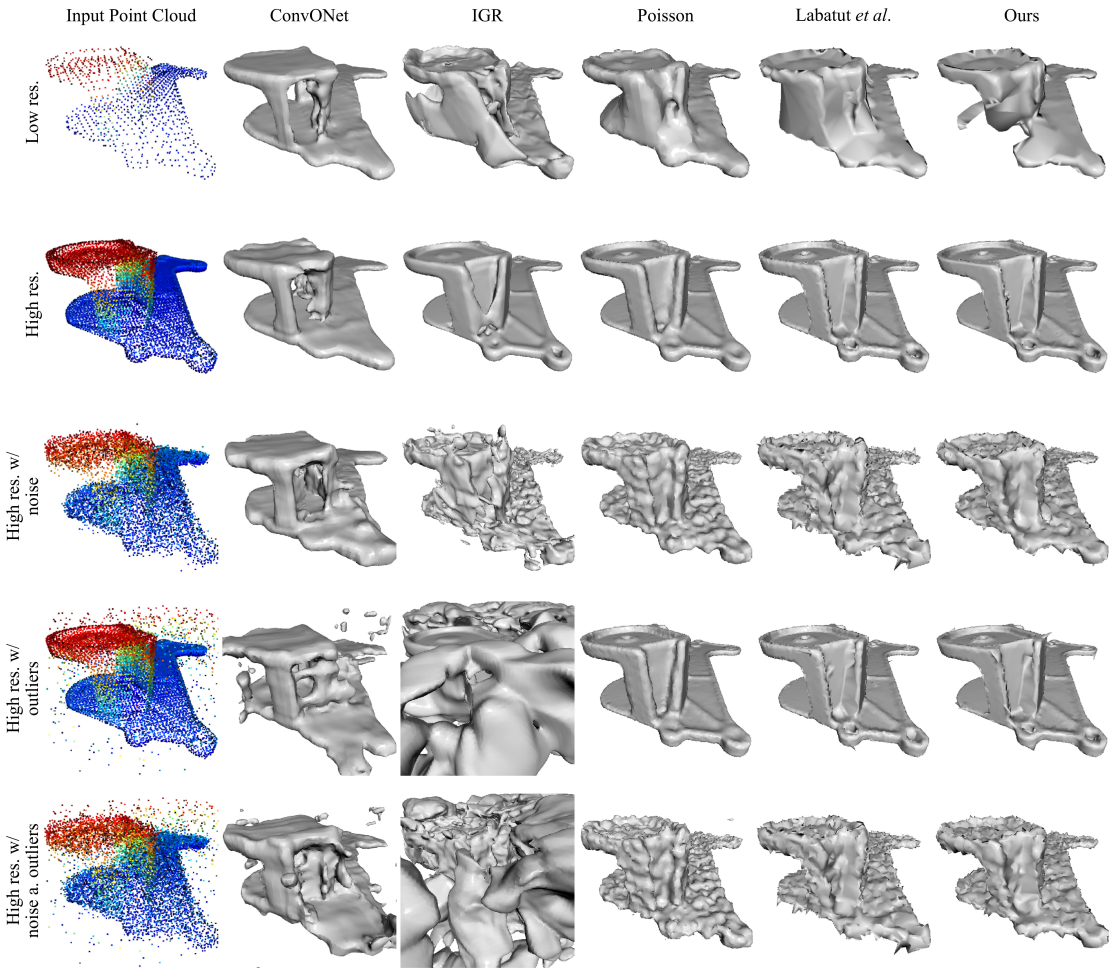


Figure 4.6: Qualitative results on Berger et al.'s daratech: We show the input point clouds in column 1. ConvONet [92] (column 2) does not generalize well to the unseen new shape. As with other shapes, IGR [48] (column 3) works well at high resolution but generates artefacts or fails in other settings. The Screened Poisson [65] algorithm (column 4) does not reconstruct the sharp features well, but is robust against outliers, even close to the surface. In the low resolution setting, our algorithm is incomplete where Labatut creates unwanted surface parts.

Table 4.2: Numerical results for Berger et al. benchmark: Object-level reconstruction with various point cloud settings: low resolution (LR), high resolution (HR), high resolution with added noise (HRN), high resolution with added outliers (HRO), high resolution with noise and outliers (HRNO). We measure the symmetric Chamfer distance to the ground truth (per-point average for objects of size 75 as done in Berger et al.'s benchmark [8]), volumetric IoU (%), number of components (ground-truth meshes all have only one component) and number of non-manifold edges (none in the ground truth). All metrics are averaged over the 5 shapes of the benchmark dataset. We compare IGR [48] "optimized" for $30,000$ iterations on each of the 5 variants (LR, HR, HRN, HRO, HRNO) of the 5 shapes, screened Poisson reconstruction [65] with an octree of depth 10, Labatut *et al.* [71] with $\alpha_{vis} = 32$, $\lambda = 5$ and σ set according to an estimation of the scan noise, and ConvONet [92] and our method trained on the ShapeNet subset from [38].

Method	**Chamfer distance** (per-point ave. %) [↓]						**Volumetric IoU** (%) [↑]					
	LR	HR	HRN	HRO	HRNO	Mean	LR	HR	HRN	HRO	HRNO	Mean
ConvONet [92]	1.90	1.80	2.31	2.91	3.73	2.53	67.8	71.3	62.9	61.4	57.3	64.1
IGR [48]	1.03	0.44	0.80	11.87	11.50	5.13	80.4	93.0	84.2	27.5	27.8	62.6
Poisson [65]	1.09	0.48	0.80	0.46	0.86	0.74	79.1	91.9	84.3	91.9	83.3	86.1
Labatut et al. [71]	0.89	0.42	0.89	0.46	0.95	0.72	81.9	94.5	80.9	94.3	80.6	86.4
Ours	**0.88**	**0.41**	**0.77**	**0.41**	**0.78**	**0.65**	**82.0**	**95.6**	**84.7**	**95.3**	**84.7**	**88.5**

Method	**Number of components** [↓]						**Number of non-manifold edges** [↓]					
	LR	HR	HRN	HRO	HRNO	Mean	LR	HR	HRN	HRO	HRNO	Mean
ConvONet [92]	3.2	2.0	6.0	12.8	14.0	7.6	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
IGR [48]	2.2	2.2	43.0	43.0	101.2	38.3	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
Poisson [65]	1.4	1.2	5.2	3.0	28.0	7.8	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
Labatut et al. [71]	**1.0**	**1.0**	2.6	1.2	4.0	2.0	0.8	0.6	18.4	0.4	14.4	6.9
Ours	1.2	**1.0**	**1.0**	**1.0**	**1.2**	**1.1**	**0.0**	2.0	0.2	1.2	**0.0**	0.7

Table 4.3: Numerical results for ETH3D reconstructions: We report the following extrinsic and intrinsic metrics on the ETH3D dataset: F1-Score at 5 cm (F1), number of connected components (CC), and surface area of the mesh in square meters $\times 10^{-2}$ (Area). Numbers in parentheses are from the meshes before the cleaning step. Note that the F1-Score is calculated from the mean values of accuracy and completeness over all scenes, in contrast to Table 4.4 where it is the mean F1-Score over all scenes.

Method	F1	CC	Area
Poisson [65]	66.8 (67.2)	83 (23131)	82 (116)
Vu *et al.* [114]	70.6 (70.8)	17 (560)	17 (125)
Jan. *et al.* [60]	70.0 (67.7)	14 (667)	14 (78)
Ours	**73.1 (72.1)**	23 (253)	11 (24)

algorithm, we estimate surface normals using Jets [29] and consistently orient them towards the sensor.

We also assess the scalability and generalization capability of ConvONet on real world outdoor scenes. We use the volume decoder model pretrained on the synthetic indoor scene dataset [92] operating on a sliding window. To avoid prohibitively expensive computations caused by far away outliers, we manually crop most of the scenes to limit the bounding volume. It is important to note that our method requires no such preprocessing. However, even in this prepared setting, the resulting surfaces were of significantly lower quality than all other methods. This can be explained by the fact that, even if the ConvONet model was trained on collections of ShapeNet models, the distribution of objects in this training set is very different from the real-life scenes of ETH3D. Our model being purely local, does not suffer from this lack of generalizability (see Figure 4.10 for a qualitative comparision of ConvONet and our method). A time and memory comparison between ConvONet, Vu *et al.* and our method is given in Table 4.5. As for IGR, the size of its network prevents us from reconstructing ETH3D scenes. Consequently, in the following, we exclude ConvONet and IGR from evaluations on ETH3D.

Results. In Figure 4.11, we present the accuracy-completeness curve for $\tau = 5,\ 10,\ 20$ and 50 cm, illustrating the varying trade-offs between completeness and accuracy for the different methods. For instance, at $\tau = 50$ cm, all methods have an F1-score of around 95%. In comparison, our method provides a lower completeness but a higher accuracy; nevertheless, it results in a better overall F-score. The higher accuracy provided by our method is illustrated in Figure 3.1 and Figure 4.12, where fine details that are hard to reconstruct are better preserved. In Table 4.3, we report intrinsic mesh quality measures at $\tau = 5$ cm for different methods. We improve the F-score by 2.5 points, while producing a surface up to 35% more compact.

In Table 4.4, we show the F1-Score at $\tau = 5$ cm of all 13 scenes of the ETH3D dataset

Table 4.4: Numerical results for ETH3D reconstruction per scene: Per scene and mean F1-Score of all scenes of the train dataset of ETH3D [99] for uncleaned and cleaned mesh reconstructions at distance $\tau = 5\,\text{cm}$. The best (highest) values per scene are in bold. We perform better than all competing methods on 8 scenes out of 13. On average, our method performs between 2 and 5% better than the competing methods, and improve the F1-score for 8 out of 13 scenes. The mesh cleaning only improves the F1-score of the reconstruction of Jancosek *et al.* [60].

	F1-score - uncleaned mesh				F1-score - cleaned mesh			
scene	Poisson	Vu et al.	Jan. et al.	Ours	Poisson	Vu et al.	Jan. et al.	Ours
kicker	0.75	**0.79**	0.75	0.76	0.75	**0.81**	0.78	0.78
pipes	0.77	**0.79**	0.77	0.76	0.77	**0.78**	0.77	0.75
delivery_area	0.69	0.70	0.66	**0.71**	0.69	0.70	0.68	**0.71**
meadow	0.45	0.52	0.51	**0.58**	0.40	0.50	0.50	**0.60**
office	0.60	**0.65**	0.59	0.59	0.60	**0.64**	0.62	0.58
playground	0.61	**0.70**	0.63	**0.70**	0.60	0.69	0.66	**0.73**
terrains	0.73	**0.78**	0.76	0.75	0.74	**0.78**	0.77	0.76
terrace	0.79	0.76	0.74	**0.83**	0.79	0.79	0.78	**0.85**
relief	0.72	0.67	0.64	**0.80**	0.73	0.69	0.67	**0.80**
relief_2	0.70	0.68	0.67	**0.79**	0.71	0.70	0.70	**0.78**
electro	0.65	0.64	0.60	**0.68**	0.65	0.65	0.64	**0.69**
courtyard	0.76	0.75	0.72	**0.77**	0.75	0.75	0.74	**0.77**
facade	0.50	0.52	0.50	**0.53**	0.51	**0.55**	0.54	0.50
mean	0.67	0.69	0.66	**0.71**	0.67	0.69	0.68	**0.71**

for both uncleaned and cleaned mesh reconstructions. Our method produces the best reconstruction scores for 9 out of 13 scenes. Mesh cleaning did not significantly alter the scores as it resulted in less complete but more accurate reconstructions.

It is important to note that the ETH3D stereo benchmark is typically used to evaluate the quality of point clouds produced by dense MVS methods. In contrast, the approaches we evaluate concern the reconstruction of compact and watertight meshes. It is a much harder task. Watertightness in particular, requires special attention regarding holes and close parallel surfaces, while algorithms producing point clouds may ignore such considerations. Consequently, the comparison of the methods we evaluate with other entries of the ETH3D benchmark is not valid.

We would like to stress that, while our model is learning-based, its training set [32] is very different from the one used to evaluate the performance [99]: we train our method on few artificial, simple and closed objects, while we evaluate on complex real-life scenes. Furthermore, our network does not optimize towards the main evaluation metrics. Instead, we optimize towards a high volumetric IoU of outside and inside cells. This implies that our model, while being simple, can learn a relevant visibility model that is able to generalize to data of unseen nature.

4.4.5 Speed and memory.

In Table 4.5, we report the speed and GPU memory requirements of the different competing methods for reconstructing the *meadow* scene from ETH3D. Our approach compares favorably to ConvONet for all space-time trade-offs, on top of improved reconstruction metrics. Note that this is mainly due to the adaptability of our method to the highly varying point cloud density of the scene. Our method only has to process a small number of large tetrahedron in empty space, while the sliding window approach of ConvONet still has to perform time-consuming decoding in such regions (cf. Table 2.8). While the added GNN inference step of our method results in a slower overall prediction compared to Vu *et al.* [114], we argue that the added accuracy justifies the extra processing time. Our method can process the entirety of the ETH benchmark in under 25 minutes.

Besides, thanks to our efficient modified GraphSAGE scheme, the unary potentials can be computed purely locally; global prediction agreement is achieved by the graph cut. We can control precisely the memory usage by choosing the number of tetrahedra to process at a time, each one using around 10 MB of memory. This memory usage can be further improved with a memory-sharing scheme between nodes, allowing us to process up to $400,000$ tetrahedra simultaneously with 8 GB of VRAM, which is the same amount of memory necessary for ConvONet to process a single sliding window.

Table 4.5: Runtimes and memory footprint: We report, for the reconstruction of the *meadow* scene (ETH3D), the computation time for point/tetrahedron features (Feat), tetrahedralization (3DT), network inference (Inference), graph cut (GC), and marching cubes (MC). Batch size is given in number of subgraphs / sliding windows. Our model alone fills 470 MB of VRAM, while ConvONet fills 540 MB.

	Batch size	Feat.	3DT	Inference		GC / MC	Total
Vu et al. [114]	-	13 s	4 s	-	-	14 s	**31 s**
Ours	400k	14 s	4 s	24 s	7.9 GB	16 s	58 s
Ours	1	14 s	4 s	75 s	**0.5 GB**	16 s	109 s
ConvONet [92]	1	5 s	-	145 s	7.9 GB	14 s	164 s

4.5 Limitations and perspectives

While neural implicit fields can also be used in conjunction with image encoders, allowing for a direct end-to-end image-to-surface reconstruction, our approach only works with point clouds as input.

Besides, as common in Delaunay-based methods, our reconstructed surface is bound to go through the triangles of the Delaunay tetrahedralization. This can limit precision when the acquisition is noisy, and prevent us from reconstructing details below the sampling resolution. Future work could address this issue by incorporating mesh refinement strategies such as vertex displacement into the learning architecture.

As for learning-based methods in general, our approach requires the training and test datasets to have comparable distributions. However, since the inference is purely local, we do not need both datasets to contain similar objects. Yet the characteristics of the acquisition must be similar in terms of accuracy and density.

We also do not fully exploit the potential of deep learning by providing the network with handcrafted features in the first GNN layer. One way to replace the handcrafted visibility features would be to directly introduce lines-of-sight as nodes in the graph, and connect these nodes to the cells which are traversed by the lines-of-sight. The handcrafted geometric features could be replaced by PointNet features, similar to the approach of ConvONet [92].

Initial experiments also show that DGNN can be used without subsequent graph-cut optimization by using a regularization loss at subgraph level. Depending on the acquisition defects, we achieve results comparable to the ones with subsequent graph-cut.

Our current architecture relies on local subgraphs with a small receptive field. While this enables fast runtimes for training and inference, it limits the feature size our learning algorithm can capture. Possible research directions involve using bigger receptive fields and incorporating pooling and up-sampling operations into the graph neural network. This could help the learning algorithm to better process information at different scales

and make the triangulation more adaptive to different features sizes.

4.6 Conclusion

We propose a scalable surface reconstruction algorithm based on graph neural networks and graph-cut optimization. Our method, trained from a small artificial dataset, is able to rival with state-of-the-art methods for large-scale reconstruction on real-life scans. Thanks to the locality of the prediction of the unary potentials associated with tetrahedra, our method can perform inference on large clouds with millions of tetrahedra. Our approach demonstrates that it is possible to integrate deep-learning with computational geometry techniques to successfully tackle the hard problem of watertight surface reconstruction at large scale.

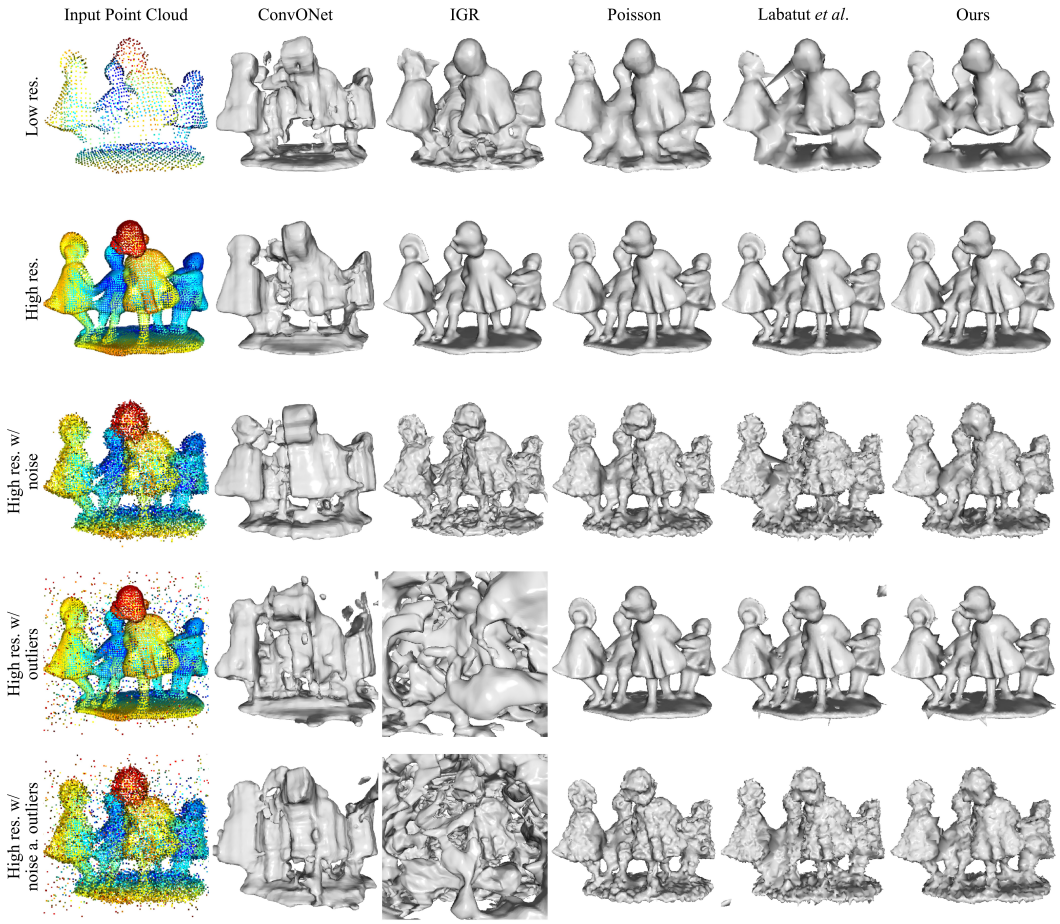


Figure 4.7: Qualitative results on Berger et al.'s dancing children: We show the input point clouds with different levels of noise and outliers to emulate challenging MVS settings in column 1. Note that in contrast to ConvONet [92], our method generalizes much better to unseen objects, is highly resilient to outliers, and does not produce the floating artifacts of the IGR [48] and Labatut [71] algorithms. The Screened Poisson reconstruction [65] is visually similar to ours, but occasionally produces unwanted surface parts.

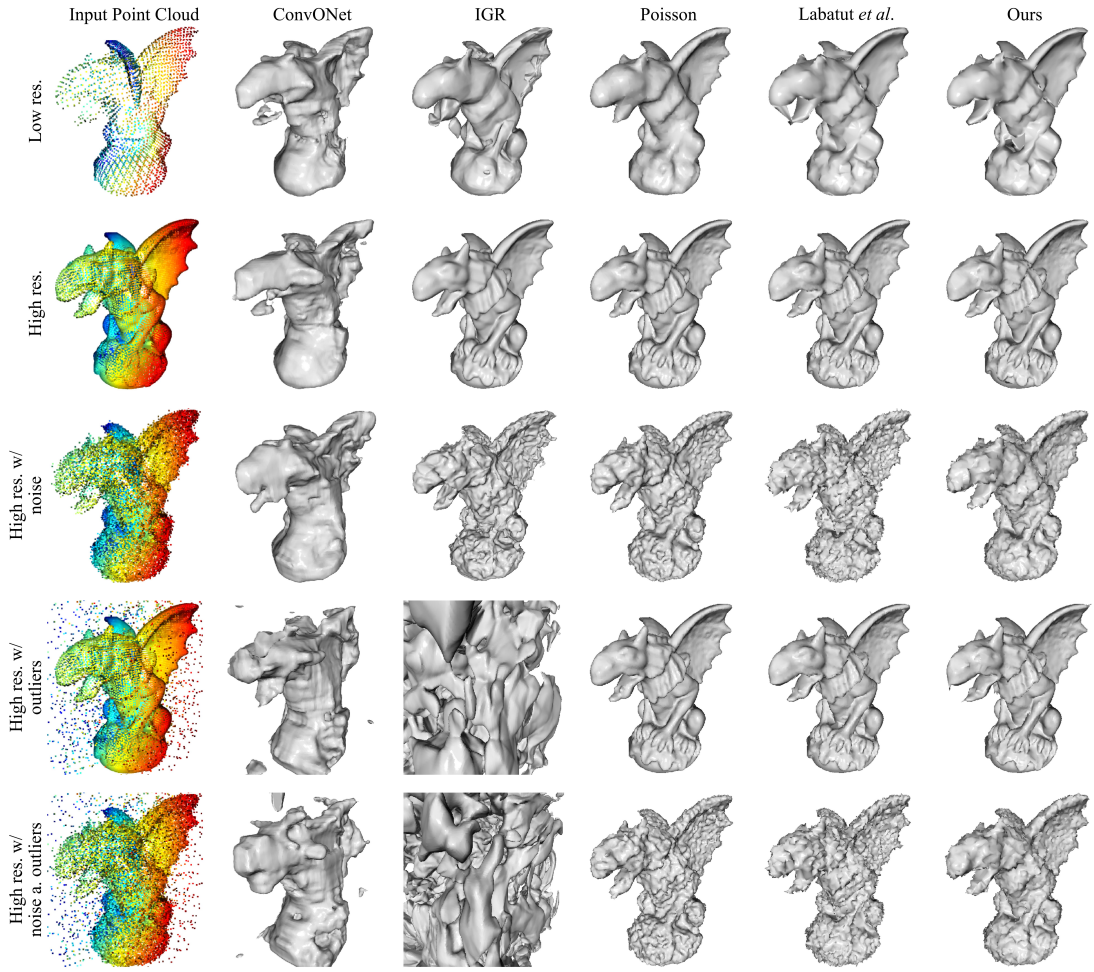


Figure 4.8: Qualitative results on Berger et al.'s gargoyle: We show the input point clouds in column 1. ConvONet [92] (column 2) does not generalize well to the unseen new shape. IGR [48] (column 3) generates many surface components from outliers. The Screened Poisson [65] algorithm (column 4) does not reconstruct the sharp features well, but is robust against outliers, even close to the surface. The reconstructions of Labatut *et al.* [71] (column 5) and ours (column 6) are visually similar for the easier high resolution case. While both methods are very robust against outliers, our method performs slightly better on the low resolution, outlier and noise cases.

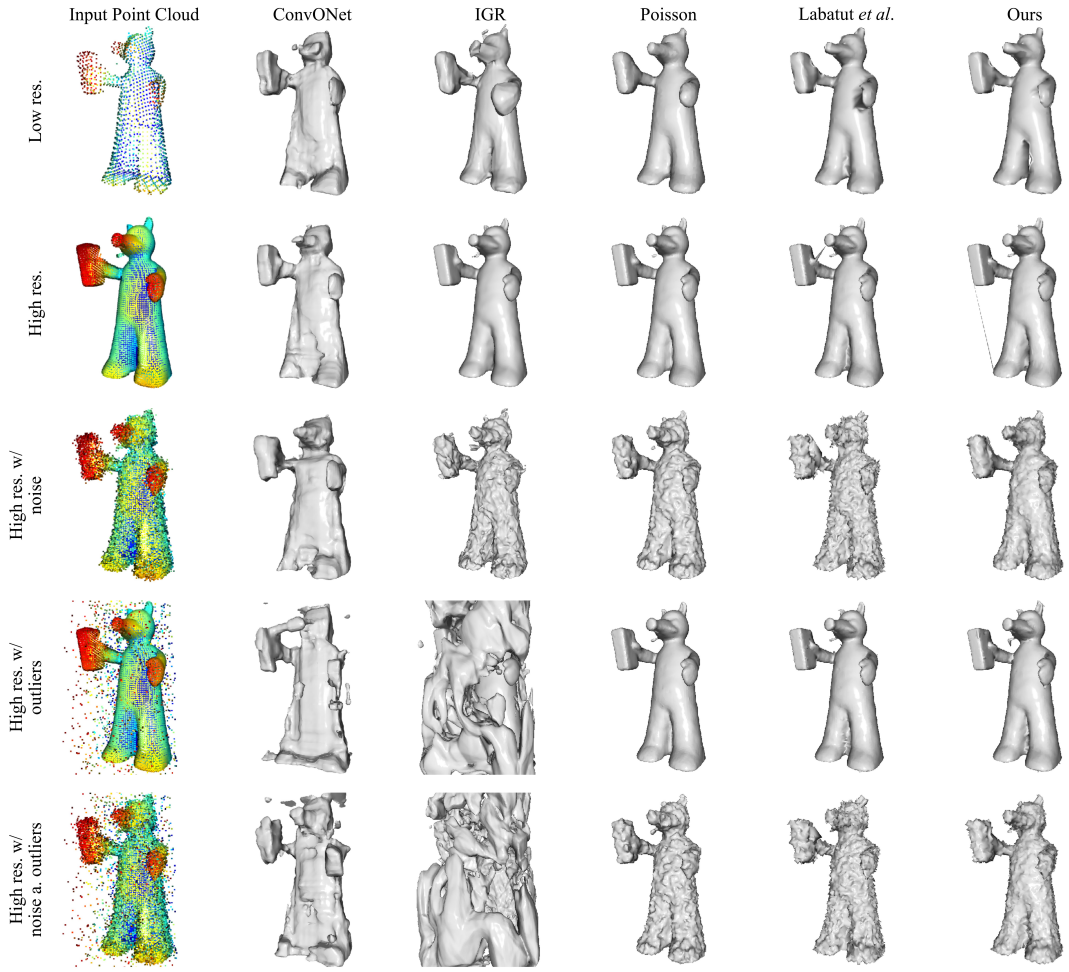
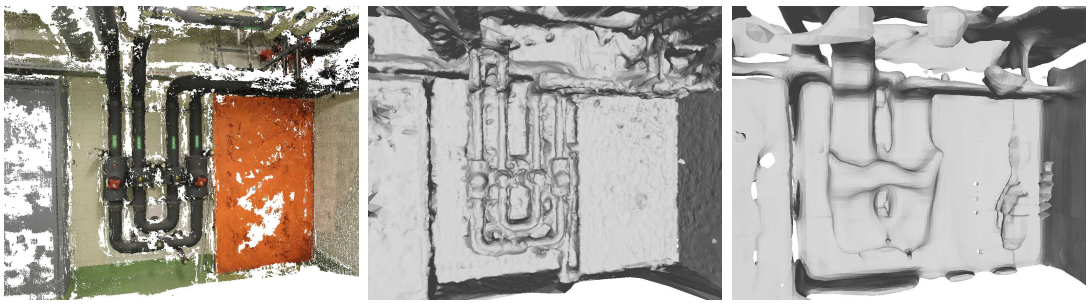


Figure 4.9: Qualitative results on Berger et al.'s lord quasimoto: We show the input point clouds in column 1. ConvONet [92] (column 2) does not generalize well to the unseen new shape. IGR [48] (column 3) is not able to filter outliers in the scan. The Screened Poisson [65] algorithm (column 4) does not reconstruct the sharp features well. The reconstructions of Labatut *et al.* [71] (column 5) and ours (column 6) are visually similar for the defect-free cases. Both methods produce small artifacts in the high resolution case: between the book and nose for Labatut *et al.* [71] and between the book and left foot for ours. Both methods are very robust against outliers.



(a) Dense MVS input. **(b)** Ours. **(c)** ConvONet [92].

Figure 4.10: Comparison of DGNN and ConvONet on ETH3D reconstruction: Reconstruction of the *pipes* scene of the ETH3D benchmark [99]. We show the dense MVS point cloud in (a), the mesh reconstructions obtained by ConvONet [92] in (c) and our proposed reconstruction in (b). Similar to object-level reconstruction, ConvONet does not generalize well to the unseen new shapes in this scene. Our learning algorithm, operating purely locally, is able to reconstruct the pipes and fill all holes in the point cloud acquistion.

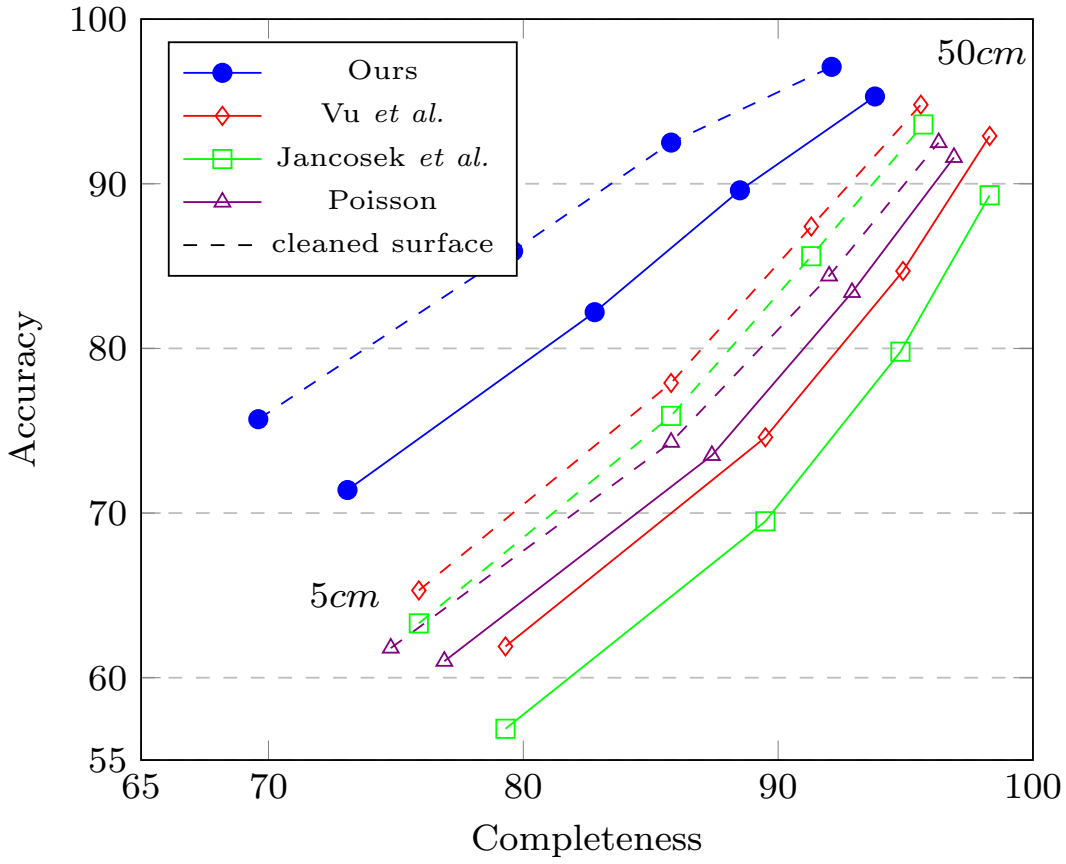
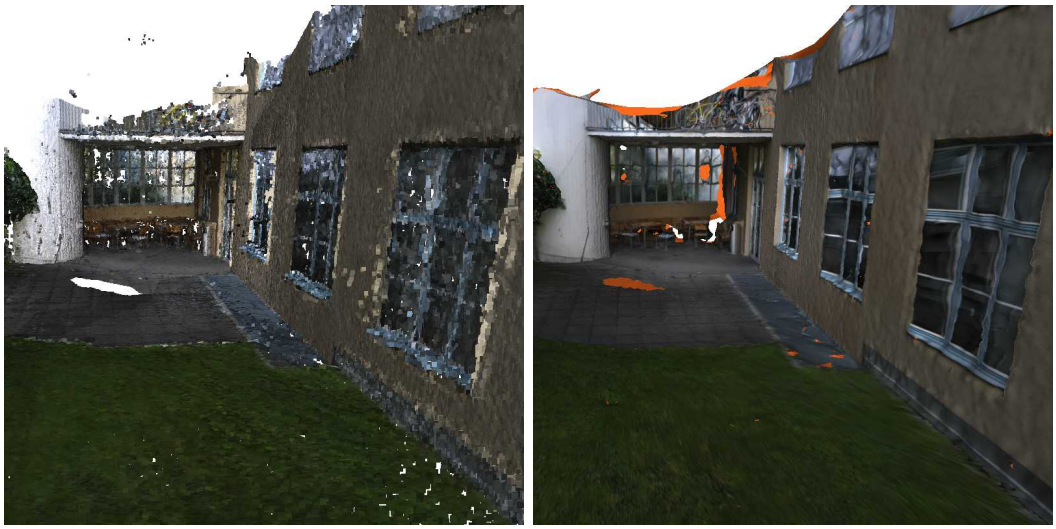
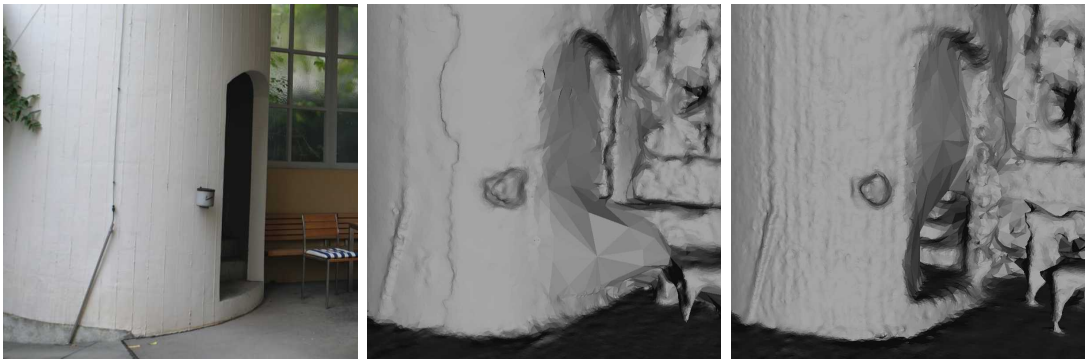


Figure 4.11: Numerical results on ETH3D: Each point corresponds to the accuracy and completeness at a given error threshold, respectively at 5, 10, 20 and 50 cm. Dashed lines represent the performance of meshes cleaned by post-processing. Our method produces meshes with a higher accuracy but a lower completeness.

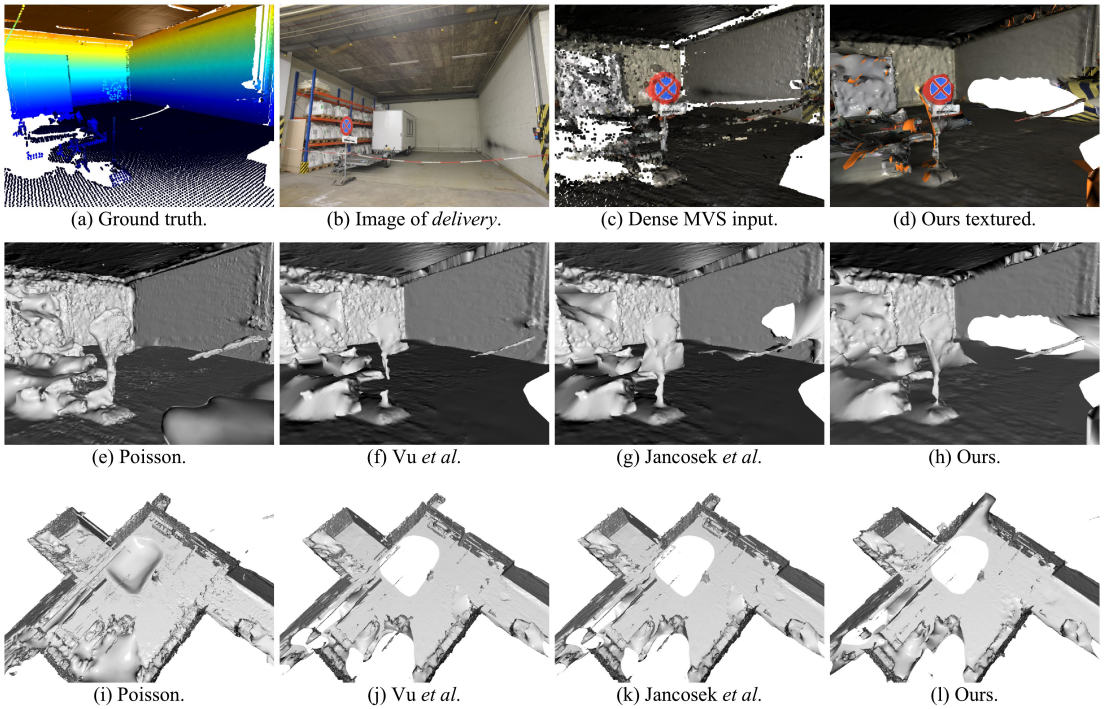


(a) Dense MVS input. **(b)** Our textured mesh.



(c) Details Image. **(d)** Jancosek et al. [60]. **(e)** Ours.

Figure 4.12: Comparison of DGNN and Jancosek et al. on ETH3D reconstruction: Our mesh reconstruction method takes as input a dense MVS point cloud (a) and produces a mesh (b), simultaneously preserving fine details and completing missing parts (here textured with [115]). We represent: in (c), a cropped image of a detail from the *terrace* scene of the ETH3D benchmark [99]; in (d), the reconstruction by Jancosek and *et al.* [60]; and in (e), our reconstruction. Notice the missing staircase and spurious vertical pattern on the concrete wall in (d). In contrast, our method (e) reconstructs part of the staircase as well as the fine-grained wall textures.



(a) Ground truth. (b) Image of *delivery*. (c) Dense MVS input. (d) Ours textured.

(e) Poisson. (f) Vu *et al.* (g) Jancosek *et al.* (h) Ours.

(i) Poisson. (j) Vu *et al.* (k) Jancosek *et al.* (l) Ours.

Figure 4.13: Failure case on ETH3D: Reconstruction of the *delivery area* scene of the ETH3D benchmark [99]. We show the ground truth that is used for evaluation in (a). A set of images, such as the one represented in (b), is transformed into a dense MVS point cloud (c), from which a mesh can be reconstructed and textured [115], as shown in (d) with our proposed mesh reconstruction. We show the untextured mesh reconstructions obtained by the screened Poisson algorithm in (e,i), the algorithms of Vu *et al.* [114] in (f,j) and of Jancosek *et al.* [60] in (g,k), and finally our proposed reconstruction in (h,l). Our method does not close the wall on the right, but performs slightly better on reconstructing the no-parking sign. Yet, considering the whole scene, the holes we create do not cover a larger area than other methods.

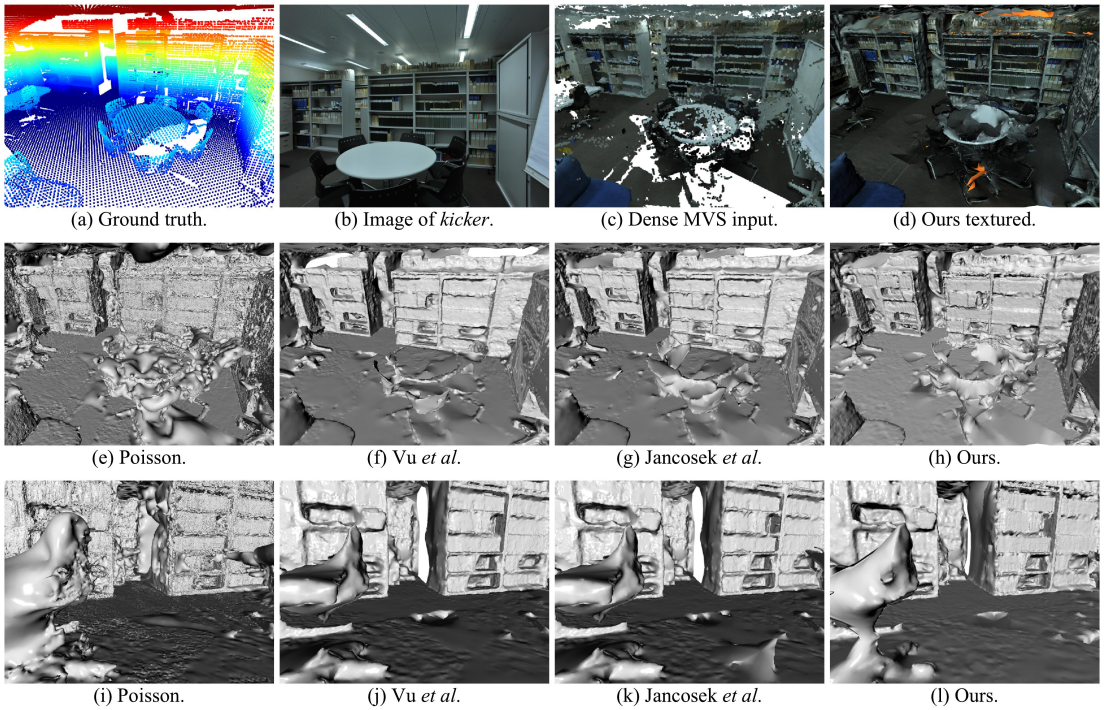


Figure 4.14: Indoor ETH3D reconstruction: Reconstruction of the *kicker* scene of the ETH3D benchmark [99]. We show the ground truth that is used for evaluation in (a). A set of images, such as the one represented in (b), is transformed into a dense MVS point cloud (c), from which a mesh can be reconstructed and textured [115], as shown in (d) with our proposed mesh reconstruction. We show the untextured mesh reconstructions obtained by the screened Poisson algorithm in (e,i), the algorithms of Vu *et al.* [114] in (f,j) and of Jancosek *et al.* [60] in (g,k), and finally our proposed reconstruction in (h,l). All methods struggle to reconstruct the table and the chairs, that have little data support.

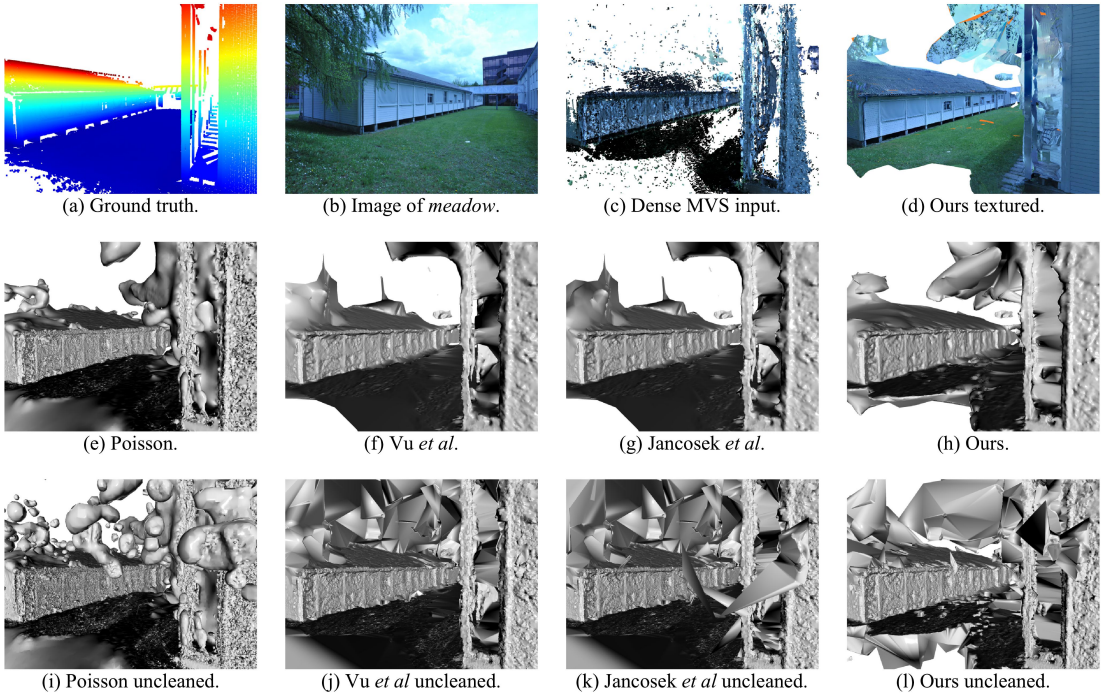


Figure 4.15: Outdoor ETH3D reconstruction: Reconstruction of the *meadow* scene of the ETH3D benchmark [99]. We show the ground truth that is used for evaluation in (a). A set of images, such as the one represented in (b), is transformed into a dense MVS point cloud (c), from which a mesh can be reconstructed and textured [115], as shown in (d) with our proposed mesh reconstruction. We show the untextured mesh reconstructions obtained by the screened Poisson algorithm in (e,i), the algorithms of Vu *et al.* [114] in (f,j) and of Jancosek *et al.* [60] in (g,k), and finally our proposed reconstruction in (h,l). Trees and outliers in the sky lead to a large number of isolated components in all mesh reconstructions. Most of these small components can be removed with the heurestic mesh cleaning step that we apply as post-processing.

"There is no real ending. It's just the place where you stop the story."

Frank Herbert

5 Conclusion

In this last chapter, we summarise the main findings of this thesis and give an outlook on future work.

5.1 Summary and conclusion

Learning-based surface reconstruction. In this thesis, we revisited the problem of surface reconstruction from point clouds with modern learning-based techniques. We focused on watertight surface reconstruction from defect-laden point clouds. We showed that learning-based methods are well suited for learning point cloud characteristics and defects from a training set of point clouds and corresponding true surfaces. This allows for reconstructing more accurate surfaces compared to traditional methods, provided that the input point clouds exhibit similar defects in train and test sets. However, we also showed that most learning based methods do not generalize to point clouds with unseen defect types. We thus advocate the use of learning-based surface reconstruction when the nature and characteristics of the defects are consistent between train and test set. Methods that mostly rely on local information do not need both datasets to contain similar objects, but the training dataset has to be sufficiently large and include diverse shapes. We further advice the use of DGNN, when the training set is small, and when short runtimes during training and inference are important. If no adequate training data is available, traditional methods are still a good choice, as they are robust to different densities and acquisition defects.

Visibility information. We introduced an easy to implement, yet powerful method to incorporate visibility information into deep surface reconstruction networks. This method consistently improves the accuracy of reconstructed surfaces and the capability of DSR networks to generalize to unseen domains. Visibility information helps to correctly orient the reconstructed surface and provides valuable information for space occupancy. Most sensors for point cloud acquisition can provide visibility information in the form of sensor poses. Sensor poses can be stored with little memory cost. We believe that the benefits of visibility information are worth this additional cost and visibility information should further be used for deep surface reconstruction.

Combining local learning with global optimization. We also showed that the combination of locally learned occupancy priors in combination with a global optimization can outperform traditional methods for the task of large-scale surface reconstruction from point clouds in the wild. The locality of our learning algorithm has several benefits. First, we require only a small amount of training data, which is a useful property for methods that require expensive 3D supervision. This is due to the fact that local shape patterns can be shared across different shape classes. Second, local learning allows to better control the memory requirements of the learning algorithm, as it removes the need to embed a global representation of the surface. Global optimization, implemented in our DGNN architecture with a graph-cut, helps to filter artifacts resulting from heavy noise and outliers.

3D Delaunay tetrahedralisation. We also showed that a 3DT can be used in combination with graph convolutions. The 3DT provides a data structure which is easy to construct and well suited for discretising the domain of non-uniform point clouds. The 3DT makes DGNN adaptive to the density of the input point clouds. This allows for training on point clouds with different densities, and for using all input points during inference. At the same time, the 3DT removes the need for expensive neighborhood searches by using the inherent graph structure to define local neighborhoods.

Point clouds in the wild. As our reconstruction of complex real-world scenes show, surfaces reconstructed from point clouds in the wild still exhibit a variety of defects. One key issue which we consider unresolved is the size of the receptive field of the learning algorithm. Point clouds in the wild include defects and shapes in a large variety of scales. This requires providing shape priors in a variety of scales, *e.g.* for closing holes in large regions of missing data, while reconstructing fine details in noisy regions at the same time. More complex approaches with graphs of several scales or graph convolution schemes with pooling and up-sampling operations may be an answer to this multi-scale issue.

5.2 Outlook and future work

Training on real data. Most surface reconstruction networks can be trained on real data with the intent to learn acquisition- or sensor-specific point cloud priors. However, the availability of real-world point clouds with corresponding true surfaces is sparse. We have experimented with training on real data and ground truth surfaces reconstructed from high quality acquisitions, but could so far not achieve good results. This area needs further investigation. Possible research directions involve learning without a ground truth surface, or simulating or measuring data for sufficiently large training sets for certain types of acquisitions and sensors.

Surface generalization. Surface reconstruction networks could also be trained with simplified ground truth surfaces, such as polygon meshes with large planar facets [7]. Most deep surface reconstruction networks, including DGNN, could be trained with such surfaces without any modification. Large datasets of point clouds and simplified models are for example available through city wide LiDAR or MVS aquisitions and hand-engineered 3D city models. This could allow for directly learning simplified models from point clouds end-to-end. A similar approach has already been implemented by using neural implicit fields to segment precomputed polyhedral volumes [34].

Visibility information. Given the high impact of visibility information in our experiments, we also suggest to explore the potential outside of surface reconstruction, for

other task involving point clouds such as semantic segmentation. Recent developments around NeRFs and differentiable volumetric rendering also heavily rely on visibility information. They use a similar strategy to ours for incorporating visibility information with sampled points along an extended ray [87]. However, their optimization process is currently slow due to dense evaluation of the neural implicit field along the ray [91]. Future work could involve incorporating 3D priors with such methods or combining them with LiDAR point clouds in a joint optimization.

Joint reconstruction and semantisation. Incorporating semantic information directly in the reconstruction process could be beneficial. A semantic aware reconstruction algorithm could (i) identify regions with known object categories, (ii) use strong shape priors in such regions, and (iii) rely on geometric priors in semantically unknown regions.

Surface requirement for real-world applications. Future investigations should also focus on assessing the impact of geometric and topological surface errors on certain types of analysis and applications. This has *e.g.* be done for 3D city models [12]. In this way, surface reconstruction algorithms could focus on the most important surface properties for a given task at hand, such as sharp feature recovery or outlier removal.

Furthermore, the quality of the triangulation itself may also be important. For example, applying finite element methods directly to a mesh output, requires a mesh with well-constructed triangles without small angles. While DGNN already provides a good triangulation by relying on a 3DT, one could directly optimize for a mesh output with such qualities.

Deep surface analysis. Recent research also suggests to use end-to-end learning for directly performing surface analysis, such as fluid dynamics from scanned point clouds. This approach removes the need to explicitly compute a continuous surface as an intermediate representation. While an interesting field of research, it can lead to more computation. Each new analysis requires a new optimization from a raw point cloud instead of a richer information surface. Additionally, such methods do not allow surface visualisation.

Delaunay triangulations and graph neural networks. We advice to do further research on the combination of graph neural networks and Delaunay triangulations. Graph neural networks are a powerful class of learning algorithms. Delaunay triangulations contain rich adjacency information while being fast to compute. They are adaptive to the data at hand and applicable to a variety of geometric problems. The combination of GNNs and Delaunay triangualtions could be useful for a variety of different appli-

cations in geometry processing such as mesh refinement or simplification, or even for tasks such as semantic segmentation.

Bibliography

[1] Alliez, P. (2017). *Surface Reconstruction*. Symposium on Geometry Processing 2017 Graduate School Lecture. http://school.geometryprocessing.org/summerschool-2017/slides/Alliez_SurfaceReconstruction_SGP.pdf. 1, 6

[2] Amenta, N., Bern, M., & Eppstein, D. (1998). The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical models and image processing*, 60(2), 125–135. 6, 25

[3] Attali, D., Boissonnat, J.-D., & Lieutier, A. (2003). Complexity of the delaunay triangulation of points on surfaces the smooth case. In *Proceedings of the nineteenth annual symposium on Computational Geometry* (pp. 201–210). 25

[4] Atzmon, M. & Lipman, Y. (2020). SAL: Sign agnostic learning of shapes from raw data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 56, 78

[5] Atzmon, M. & Lipman, Y. (2021). SALD: sign agnostic learning with derivatives. In *International Conference on Learning Representations (ICLR)*. 56, 78

[6] Barnes, C., Shechtman, E., Finkelstein, A., & Goldman, D. B. (2009). PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*. 90

[7] Bauchet, J.-P. & Lafarge, F. (2020). Kinetic shape reconstruction. *ACM Transactions on Graphics (TOG)*, 39(5), 1–14. 110

[8] Berger, M., Levine, J. A., Nonato, L. G., Taubin, G., & Silva, C. T. (2013). A benchmark for surface reconstruction. *ACM Transaction on Graphics*. 16, 30, 32, 86, 90, 93

[9] Berger, M., Tagliasacchi, A., Seversky, L., Alliez, P., Guennebaud, G., Levine, J., Sharf, A., & Silva, C. (2016). A survey of surface reconstruction from point clouds. *Computer Graphics Forum*. 5, 6, 16

[10] Bernardini, F. & Bajaj, C. L. (1997). Sampling and reconstructing manifolds using alpha-shapes. *Proc. 9th Canad. Conf. Comput. Geom.* 25

[11] Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., & Taubin, G. (1999). The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4), 349–359. 21, 22

[12] Biljecki, F., Heuvelink, G., Ledoux, H., & Stoter, J. (2018). The effect of acquisition error and level of detail on the accuracy of spatial analyses. *Cartography and Geographic Information Science*, 45(2), 156–176. 111

[13] Biljecki, F., Ledoux, H., Du, X., Stoter, J., Soon, K. H., & Khoo, V. (2016). The most common geometric and semantic errors in citygml datasets. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4. 5

[14] Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., & Çöltekin, A. (2015). Applications of 3d city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4), 2842–2889. 1

[15] Boissonnat, J.-D. (1984). Geometric structures for 3d shape representation. *ACM Transactions on Graphics*, 3(4). 6, 23, 25

[16] Boissonnat, J.-D. & Oudot, S. (2005). Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5), 405–451. 21

[17] Bolle, R. M. & Vemuri, B. C. (1991). On three-dimensional surface reconstruction methods. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(01), 1–13. 16

[18] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., & Lévy, B. (2010). *Polygon mesh processing*. CRC press. 11, 16, 17

[19] Boulch, A. (2020). Convpoint: Continuous convolutions for point cloud processing. *Computers & Graphics*. 39, 60

[20] Boulch, A., de La Gorce, M., & Marlet, R. (2014). Piecewise-planar 3D reconstruction with edge and corner regularization. *Computer Graphic Forum*. 78

[21] Boulch, A. & Marlet, R. (2022). Poco: Point convolution for surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 6302–6314). 16, 22, 39, 42, 43, 47, 50, 51, 56, 60, 62, 63, 64, 67, 68, 69, 70, 71, 72, 73

[22] Boykov, Y. & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*. 85

[23] Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *PAMI*. 89

[24] Breitenmoser, A. & Siegwart, R. (2012). Surface reconstruction and path planning for industrial inspection with a climbing robot. In *2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI)* (pp. 22–27).: IEEE. 1

[25] Bódis-Szomorú, A., Riemenschneider, H., & Van Gool, L. (2016). Efficient volumetric fusion of airborne and street-side data for urban reconstruction. In *International Conference on Pattern Recognition (ICPR)*. 7, 54, 76, 78

[26] Caraffa, L., Brédif, M., & Vallet, B. (2017). 3D watertight mesh generation with uncertainties from ubiquitous data. In *Asian Conference on Computer Vision (ACCV)*. 25, 54, 76, 78

[27] Caraffa, L., Marchand, Y., Brédif, M., & Vallet, B. (2021). Efficiently distributed watertight surface reconstruction. In *3DV*. 54

[28] Cazals, F. & Giesen, J. (2006). Delaunay triangulation based surface reconstruction. In *Effective computational geometry for curves and surfaces* (pp. 231–276). Springer. 6, 16, 20, 21, 23, 25

[29] Cazals, F. & Pouget, M. (2005). Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*. 37, 38, 60, 61, 62, 94

[30] Cernea, D. (2020). OpenMVS: Multi-view stereo reconstruction library. 59, 86, 87, 90

[31] Chabra, R., Lenssen, J. E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., & Newcombe, R. (2020). Deep local shapes: Learning local SDF priors for detailed 3D reconstruction. In *European Conference on Computer Vision (ECCV)*. 16, 27, 54, 78

[32] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., & Yu, F. (2015). *ShapeNet: An Information-Rich 3D Model Repository*. Technical report, Stanford University, Princeton University, Toyota Technological Institute at Chicago. 27, 59, 96

[33] Chauve, A.-L., Labatut, P., & Pons, J.-P. (2010). Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 76

[34] Chen, Z., Khademi, S., Ledoux, H., & Nan, L. (2021). Reconstructing compact building models from point clouds using deep implicit fields. *arXiv preprint arXiv:2112.13142*. 110

[35] Chen, Z. & Zhang, H. (2019). Learning implicit fields for generative shape modeling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 22, 26, 27, 54

[36] Chibane, J., Alldieck, T., & Pons-Moll, G. (2020a). Implicit functions in feature space for 3D shape reconstruction and completion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 56

[37] Chibane, J., Mir, A., & Pons-Moll, G. (2020b). Neural unsigned distance fields for implicit function learning. In *Conference on Neural Information Processing Systems (NeurIPS)*. 56

[38] Choy, C. B., Xu, D., Gwak, J., Chen, K., & Savarese, S. (2016). 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *European Conference on Computer Vision (ECCV)*. 34, 59, 86, 93

[39] Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016). 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention* (pp. 424–432).: Springer. 27

[40] Dai, A., Chang, A., Savva, M., Halber, M., Funkhouser, T., & Nießner, M. (2017). ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 59

[41] Dai, A., Diller, C., & Nießner, M. (2020). SG-NN: Sparse generative neural networks for self-supervised scene completion of RGB-D scans. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 54, 56

[42] Dai, A. & Nießner, M. (2019). Scan2Mesh: From unstructured range scans to 3D meshes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 78

[43] Deprelle, T., Groueix, T., Fisher, M., Kim, V., Russell, B., & Aubry, M. (2019). Learning elementary structures for 3D shape generation and matching. In *Conference on Neural Information Processing Systems (NeurIPS)*. 78

[44] Digne, J. (2014). An analysis and implementation of a parallel ball pivoting algorithm. *Image Processing On Line*, 4, 149–168. 21

[45] Erler, P., Ohrhallinger, S., Mitra, N., & Wimmer, M. (2020). Points2Surf: Learning implicit surfaces from point clouds. In *European Conference on Computer Vision (ECCV)*. 16, 22, 50, 56, 60, 62, 63, 64, 67, 68, 69, 70, 71, 72

[46] Gkioxari, G., Johnson, J., & Malik, J. (2019). Mesh R-CNN. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 78

[47] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org. 11

[48] Gropp, A., Yariv, L., Haim, N., Atzmon, M., & Lipman, Y. (2020). Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning (ICML)*. 22, 28, 29, 37, 45, 47, 51, 54, 78, 79, 87, 90, 91, 92, 93, 99, 100, 101

[49] Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., & Aubry, M. (2018). AtlasNet: A papier-mâché approach to learning 3D surface generation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 9, 22, 24, 78

[50] Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Conference on Neural Information Processing Systems (NeurIPS)*. 82

[51] Handa, A., Patraucean, V., Stent, S., & Cipolla, R. (2016). SceneNet: An annotated model generator for indoor scene understanding. In *International Conference on Robotics and Automation (ICRA)*. 59

[52] Hanocka, R., Metzer, G., Giryes, R., & Cohen-Or, D. (2020). Point2Mesh: A self-prior for deformable meshes. *ACM Transaction on Graphics*. 22, 24, 38, 45, 47, 51, 54, 56, 78

[53] Hiep, V. H., Keriven, R., Labatut, P., & Pons, J.-P. (2009). Towards high-resolution large-scale multi-view stereo. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1430–1437).: IEEE. 25

[54] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., & Stuetzle, W. (1992). Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on computer graphics and interactive techniques* (pp. 71–78). 17, 25

[55] Huang, J., Su, H., & Guibas, L. (2018). Robust watertight manifold surface generation method for ShapeNet models. *arXiv preprint arXiv:1802.01698*. 86

[56] Huang, J., Zhou, Y., & Guibas, L. (2020). Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups. *arXiv preprint arXiv:2005.11621*. 34, 59

[57] Huang, X., Mei, G., Zhang, J., & Abbas, R. (2021). A comprehensive survey on point cloud registration. *arXiv preprint arXiv:2103.02690*. 7

[58] Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*. 86

[59] Jancosek, M. & Pajdla, T. (2011). Multi-view reconstruction preserving weakly-supported surfaces. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 25, 54, 76, 78

[60] Jancosek, M. & Pajdla, T. (2014). Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces. *International Scholarly Research Notices*. 25, 54, 76, 77, 78, 87, 94, 95, 104, 105, 106, 107

[61] Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., & Aanæs, H. (2014). Large scale multi-view stereopsis evaluation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 17, 29, 34, 59, 71

[62] Jiang, C. M., Sud, A., Makadia, A., Huang, J., Nießner, M., & Funkhouser, T. (2020). Local implicit grid representations for 3D scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 16, 22, 37, 45, 47, 51, 54, 56, 60, 62

[63] Kazhdan, M., Bolitho, M., & Hoppe, H. (2006). Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (pp. 61–70).: Eurographics Association. 25

[64] Kazhdan, M., Chuang, M., Rusinkiewicz, S., & Hoppe, H. (2020). Poisson Surface Reconstruction with Envelope Constraints. *Computer Graphics Forum*, 39(5), 173–182. 26

[65] Kazhdan, M. & Hoppe, H. (2013). Screened Poisson surface reconstruction. *ACM Transaction on Graphics*. 4, 9, 22, 24, 26, 28, 38, 39, 42, 43, 45, 47, 49, 50, 51, 56, 60, 77, 87, 91, 92, 93, 94, 99, 100, 101

[66] Kettner, L. (1999). Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry*, 13(1), 65–90. 20

[67] Kingma, D. P. & Ba, J. (2015). ADAM: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*. 86

[68] Knapitsch, A., Park, J., Zhou, Q.-Y., & Koltun, V. (2017). Tanks and Temples. *ACM Transactions on Graphics (TOG)*. 17, 29, 34, 59, 71

[69] Kolluri, R., Shewchuk, J. R., & O'Brien, J. F. (2004). Spectral surface reconstruction from noisy point clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (pp. 11–21). 25

[70] König, S. & Gumhold, S. (2009). Consistent propagation of normal orientations in point clouds. In *International Workshop on Vision, Modeling, and Visualization (VMV)*. 56

[71] Labatut, P., Pons, J. P., & Keriven, R. (2009). Robust and efficient surface reconstruction from range data. *Computer Graphics Forum (CGF)*. 9, 12, 22, 25, 38, 39, 42, 43, 45, 47, 49, 50, 51, 54, 76, 78, 85, 87, 90, 91, 93, 99, 100, 101

[72] Lemmens, M. (2020). *An Introduction to Pointcloudmetry: Point Clouds from Laser Scanning and Photogrammetry*. Whittles. 11

[73] Li, Z., Zhu, C., & Gold, C. (2004). *Digital terrain modeling: principles and methodology*. CRC press. 1

[74] Liu, M., Zhang, X., & Su, H. (2020). Meshing point clouds with predicted intrinsic-extrinsic ratio guidance. In *European Conference on Computer Vision* (pp. 68–84).: Springer. 22, 23, 78

[75] Liu, S., Saito, S., Chen, W., & Li, H. (2019). Learning to infer implicit surfaces without 3D supervision. In *Conference on Neural Information Processing Systems (NeurIPS)*. 79

[76] Lorensen, W. E. & Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4), 163–169. 21, 37, 61

[77] Luo, Y., Mi, Z., & Tao, W. (2021). Deepdt: Learning geometry from delaunay triangulation for surface reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35 (pp. 2277–2285). 25

[78] Mäntylä, M. (1987). *An introduction to solid modeling*. Computer Science Press, Inc. 20

[79] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., & Geiger, A. (2019). Occupancy networks: Learning 3D reconstruction in function space. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 9, 22, 26, 54, 59, 79

[80] Metzer, G., Hanocka, R., Giryes, R., & Cohen-Or, D. (2021a). Self-sampling for neural point cloud consolidation. *ACM Transactions on Graphics (TOG)*, 40(5), 1–14. 11

[81] Metzer, G., Hanocka, R., Zorin, D., Giryes, R., Panozzo, D., & Cohen-Or, D. (2021b). Orienting point clouds with dipole propagation. *ACM Transactions on Graphics (TOG)*. 56

[82] Mi, Z., Luo, Y., & Tao, W. (2020). SSRNet: Scalable 3D surface reconstruction network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 9, 79

[83] Michalkiewicz, M., Pontes, J., Jack, D., Baktashmotlagh, M., & Eriksson, A. (2019). Implicit surface representations as layers in neural networks. In *International Conference on Computer Vision (ICCV)*. 54

[84] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*. 10, 56

[85] Mostegel, C., Prettenthaler, R., Fraundorfer, F., & Bischof, H. (2017). Scalable surface reconstruction from point clouds with extreme scale and density diversity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 904–913). 25

[86] Niemeyer, M., Mescheder, L., Oechsle, M., & Geiger, A. (2020). Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 3504–3515). 10, 56

[87] Oechsle, M., Peng, S., & Geiger, A. (2021). Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *International Conference on Computer Vision (ICCV)*. 10, 56, 111

[88] O'neill, B. (2006). *Elementary differential geometry*. Elsevier. 17

[89] Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2019). DeepSDF: Learning continuous signed distance functions for shape representation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 16, 22, 26, 27, 54, 78, 79

[90] Paschalidou, D., Ulusoy, O., Schmitt, C., Van Gool, L., & Geiger, A. (2018). Raynet: Learning volumetric 3d reconstruction with ray potentials. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 56

[91] Peng, S., Jiang, C. M., Liao, Y., Niemeyer, M., Pollefeys, M., & Geiger, A. (2021). Shape as points: A differentiable poisson solver. In *Conference on Neural Information Processing Systems (NeurIPS)*. 9, 22, 27, 28, 38, 42, 43, 45, 47, 50, 51, 60, 62, 63, 64, 67, 68, 69, 70, 71, 72, 111

[92] Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., & Geiger, A. (2020). Convolutional occupancy networks. In *European Conference on Computer Vision*

(ECCV). 16, 22, 27, 32, 38, 42, 43, 47, 50, 51, 56, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 76, 79, 87, 90, 91, 92, 93, 94, 97, 99, 100, 101, 102

[93] Petitjean, S. & Boyer, E. (2001). Regular and non-regular point sets: Properties and reconstruction. *Computational Geometry*, 19(2-3), 101–126. 21

[94] Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 73

[95] Rakotosaona, M.-J., Guerrero, P., Aigerman, N., Mitra, N. J., & Ovsjanikov, M. (2021). Learning delaunay surface elements for mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 22–31). 16, 22, 23

[96] Rakotosaona, M.-J., La Barbera, V., Guerrero, P., Mitra, N. J., & Ovsjanikov, M. (2020). Pointcleannet: Learning to denoise and remove outliers from dense point clouds. In *Computer Graphics Forum*, volume 39 (pp. 185–203).: Wiley Online Library. 6, 11

[97] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234–241).: Springer. 27

[98] Schertler, N., Savchynskyy, B., & Gumhold, S. (2017). Towards globally optimal normal orientations for large point clouds. *Computer Graphics Forum (CFG)*. 54, 56, 60, 61, 62

[99] Schöps, T., Schönberger, J. L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., & Geiger, A. (2017). A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 4, 7, 8, 17, 29, 76, 77, 88, 90, 95, 96, 102, 104, 105, 106, 107

[100] Seitz, S., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 17, 29, 34, 59, 71

[101] Sharf, A., Lewiner, T., Shamir, A., Kobbelt, L., & Cohen-Or, D. (2006). Competing fronts for coarse–to–fine surface reconstruction. In *Computer Graphics Forum*, volume 25 (pp. 389–398).: Wiley Online Library. 22, 24

[102] Sharp, N. & Ovsjanikov, M. (2020). "pointtrinet: Learned triangulation of 3d point sets". In *European Conference on Computer Vision (ECCV)*. 22, 23, 56, 78

[103] Simonovsky, M. & Komodakis, N. (2017). Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 89

[104] Sinha, S. N., Mordohai, P., & Pollefeys, M. (2007). Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *2007 IEEE 11th international conference on computer vision* (pp. 1–8).: IEEE. 25

[105] Solem, J. E. (2012). *Programming Computer Vision with Python: Tools and algorithms for analyzing images.* " O'Reilly Media, Inc.". 11

[106] Song, S., Cui, Z., & Qin, R. (2021). Vis2mesh: Efficient mesh reconstruction from unstructured point clouds of large scenes with learned virtual view visibility. In *International Conference on Computer Vision (ICCV)*. 56

[107] Strecha, C., von Hansen, W., Gool, L. V., Fua, P., & Thoennessen, U. (2008). On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Conference on Computer Vision and Pattern Recognition (CVPR)*: IEEE Computer Society. 17, 29

[108] Sulzer, R., Landrieu, L., Boulch, A., Marlet, R., & Vallet, B. (2022). Deep surface reconstruction from point clouds with visibility information. In *International Conference on Pattern Recognition (ICPR)*. 12

[109] Sulzer, R., Landrieu, L., Marlet, R., & Vallet, B. (2021). Scalable surface reconstruction with delaunay-graph neural networks. *Computer Graphics Forum*, 40(5), 157–167. 9, 12, 16, 22, 25, 38, 42, 43, 47, 50, 51, 54, 56, 60, 62, 67

[110] Tang, J., Lei, J., Xu, D., Ma, F., Jia, K., & Zhang, L. (2021). SA-ConvONet: Sign-agnostic optimization of convolutional occupancy networks. In *International Conference on Computer Vision (ICCV)*. 56

[111] Tognola, G., Parazzini, M., Ravazzani, P., Svelto, C., & Grandori, F. (2001). 3d reconstruction of anatomical surfaces from unorganized range data. In *2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 3 (pp. 2534–2536).: IEEE. 1

[112] Ummenhofer, B. & Koltun, V. (2021). Adaptive surface reconstruction with multiscale convolutional kernels. In *International Conference on Computer Vision (ICCV)*. 56

[113] Vetsch, M., Lombardi, S., Pollefeys, M., & Oswald, M. R. (2022). Neuralmeshing: Differentiable meshing of implicit neural representations. In B. Andres, F.

Bernard, D. Cremers, S. Frintrop, B. Goldlücke, & I. Ihrke (Eds.), *Pattern Recognition* (pp. 317–333). Cham: Springer International Publishing. 21

[114] Vu, H. H., Labatut, P., Pons, J. P., & Keriven, R. (2012). High accuracy and visibility-consistent dense multiview stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 4, 6, 25, 54, 76, 77, 78, 87, 94, 96, 97, 105, 106, 107

[115] Waechter, M., Moehrle, N., & Goesele, M. (2014). Let there be color! large-scale texturing of 3D reconstructions. In *ECCV*. 77, 104, 105, 106, 107

[116] Williams, F., Schneider, T., Silva, C., Zorin, D., Bruna, J., & Panozzo, D. (2018). Deep geometric prior for surface reconstruction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 78

[117] Williams, F., Trager, M., Bruna, J., & Zorin, D. (2021). Neural splines: Fitting 3d surfaces with infinitely-wide neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 56

[118] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 59

[119] Yang, Y., Feng, C., Shen, Y., & Tian, D. (2018). FoldingNet: Point cloud auto-encoder via deep grid deformation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 78

[120] You, C. C., Lim, S. P., Lim, S. C., San Tan, J., Lee, C. K., & Khaw, Y. M. J. (2020). A survey on surface reconstruction techniques for structured and unstructured data. In *2020 IEEE Conference on Open Systems (ICOS)* (pp. 37–42).: IEEE. 16

[121] Zhao, W., Lei, J., Wen, Y., Zhang, J., & Jia, K. (2021). Sign-agnostic implicit learning of surface self-similarities for shape modeling and reconstruction from raw point clouds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 56, 60

[122] Zhou, Y., Shen, S., & Hu, Z. (2019). Detail preserved surface reconstruction from point cloud. *Sensors*. 25, 54, 76, 78