One-sentence description : We introduce new visibility-based metrics to assess surface reconstruction from point clouds both in terms of completeness and accuracy.

Address for all authors: 73 Avenue de Paris, 94160 Saint-Mandé

Affiliation for all authors: LASTIG, Univ Gustave Eiffel, IGN-ENSG, F-94160 Saint-Mande, France

Evaluating surface mesh reconstruction using real data

Yanis Marchand (yanismarchan@gmail.com), Laurent Caraffa (laurent.caraffa@ign.fr), Raphael Sulzer (raphaelsulzer@gmx.de), Emmanuel Clédat (emmanuel.cledat@ensg.eu), Bruno Vallet (bruno.vallet@ign.fr)

Abstract

Surface reconstruction has been studied thoroughly but very little work has been done to address its evaluation. In this paper, we propose new visibility-based metrics to assess the completeness and the accuracy of 3-dimensional meshes based on a point cloud of higher accuracy than the one the reconstruction has been computed from. We use the position from which each high-quality point has been acquired to compute the corresponding ray of free space. Based on the intersections between each ray and the reconstructed surface, our metrics allow to evaluate both the global coherency of the reconstruction and the accuracy at close range. We validate this evaluation protocol by surveying several open-source algorithms as well as a piece of licensed software on three datasets. The results confirm the relevance of assessing local and global accuracy separately since algorithms sometimes fail at guaranteeing both simultaneously. In addition, algorithms making use of sensor positions perform better than the ones only relying on points and normals, indicating a potentially significant added value of this piece of information. Our implementation is available at GitHub/SurfaceReconEval.

Keywords: Computer Graphics, Surface Reconstruction, Evaluation, Mesh, Point Cloud

1. Introduction

Surface reconstruction is the task of producing a continuous digital representation of a real surface of which discrete information has been acquired. This information may come straight from point clouds produced by a laser scanner. This includes Time-of-Flight [1] and Structured-light [2] devices as well as terrestrial and airborne LiDAR [3] that allow scanning large environments. Point clouds can also be produced from images using Multi-view stereo [4] or Structure from motion [5].

This task has been extensively studied and a tremendous amount of approaches have been proposed. In Section 2, we provide a state of the art of these methods. However, very few papers address the evaluation of such a task. In real-case scenarios when the goal is to produce a digital model of a real object or scene, there is no ground truth other than the real surface itself. It is thus impossible to directly compute the "distance" or the "difference" between a digital

model and the ideal real surface. The only possible work-around is using synthetic data as in [6] where a realistic surface is chosen to be the ground truth and is then virtually scanned in a way that simulates the defects of a real acquisition, and the surface reconstruction algorithms to be evaluated are run on this virtual scan. This makes it more straightforward to compute metrics that assess the difference between the ground truth model and the reconstructed ones. Another possibility for working with data from real scenes is to sample points from a reconstructed mesh, but this introduces a large bias as methods producing the same features will be unfairly favoured. Our work tackles the real-world case where we do not have access to such a synthetic ground truth. We call "**real data**" the data acquired in the physical world with real sensors. This includes LiDAR scans, images, RGB-D images, etc. As is usually done to address this issue, we assess the reconstruction of real scenes from real data only based on other real data of significantly higher quality. Even though this idea is quite typical, the main contribution of our paper lies in the way that we assess the difference between the reconstructed surface and the high-quality real data as inconsistencies, inspired by recent work on change detection [7].

The fundamental interest of this work is to propose metrics to assess the quality of reconstructions from low-quality real data based on high-quality data. Although it is possible to assess the quality of models visually, this raises several issues. First, it is a subjective comparison, one might be tempted to favor their own or preferred method over others. Secondly, everyone has a different perception of visual quality and we might not agree even without conflict of interest. Thirdly, while it might sound reasonable to visually evaluate a few different models of a relatively small scene, it is very unlikely that one would be able to carry out a large-scale evaluation involving dozens of models representing large areas. Consequently, we believe it is essential to find relevant metrics to assess surface reconstruction and, in our opinion, current metrics are not entirely satisfying. As pointed out by [8], there is a lack of ground truth and of benchmarks in the field of urban reconstruction. Our paper aims to tackle this problem. This endeavor is hard because of several aspects.

Limits in the quality of the ground truth: the specificity of working with real data is the presence of *noise* in what we consider as the ground truth. In addition, real data is always sparse and incomplete, which means that we do not know the state of space (*occupied* by the object or *empty*) everywhere. This raises the question of how to assess pieces of reconstructed

surface in *unseen, unobserved* regions.

Our contributions are twofold:

1. We propose a setting where the high-quality data used to compute metrics is significantly better than the low-quality data on which surface reconstruction is performed in three separate ways:
    - Coverage: we use multiple data sources acquired from multiple points of view to ensure that the high-quality data has a significantly better coverage of the surface to reconstruct than the low-quality data.
    - Density: we ensure that the density of the points in the high-quality data is significantly better than that of the low-quality data.
    - Noise: we ensure that the noise level is lower in the high-quality data than in the low-quality data.
2. We propose metrics that penalize inconsistencies between the surface to be evaluated and the high-quality data: a piece of surface reconstructed within a volume unseen by the high-quality data will simply not be evaluated as we have no information on it. This does not mean that we do not evaluate the hole-filling capacity of the evaluated methods. As the high-quality data has more coverage, we evaluate hole filling exactly where we have the data to do so.

Assumptions and priors: algorithms make different assumptions about the type of shape that needs to be reconstructed and this leads to very different properties. Consequently, depending on the metrics' definitions, these assumptions can dramatically influence the assessment and sometimes in an unjustified manner. An example of such a situation is shown in Figure 1 where the left model would be attributed a bad mark because of the red piece of surface even though the rest of the model is correct. Does this red piece of surface need to be taken into account when assessing the model? We tackle this issue in two complementary ways: first we use a softer definition of watertightness adapted to the reconstruction of open scenes (see below), and second we define metrics that only assess hole filling where relevant *high-quality* data is available.

A surface is *watertight* if it has no border. In the case of a triangle mesh, this means each edge needs to have exactly two incident faces. We will call this property *hard watertightness*. When

trying to reconstruct an open scene, for example, it is often more realistic to authorize the reconstructed surface to intersect the boundary of the domain (a bounding box, for instance) as illustrated in Figure 1. We thus define *soft watertightness* as the property that a mesh has when it has no border except on the boundary of the domain. Only triangle edges lying on the domain boundary can have only one incident face. In practice, we only ask for the evaluated methods to be softly watertight, which means that they do not need to randomly fill the very large hole at the boundary of the domain when reconstructing open scenes. In fact, most reconstruction methods have this ability: Poisson reconstruction [9] using the Neumann boundary condition (opposite to Dirichlet) allows the reconstructed surface to be open at the domain boundary. For Delaunay-based methods, we can simply add eight points to the input point cloud to correspond to the corners of a bounding box, and remove all triangles belonging to this bounding box at surface extraction time (or in post-processing).

Our proposed metrics are based on the visibility information contained in the high-quality data. We assess the reconstructed surface only where the real one has been observed. For that purpose, we make an extensive use of *sensor positions* (positions from which points have been acquired). This information is easy to access and it provides us with a full ray along which we know that space is free, instead of just a single position where we know that the real surface lies. We used these newly-defined metrics to assess several open-source surface reconstruction algorithms and a licensed one on different types of scenes. Even though our protocol is intended for real data, one of the datasets that we used is synthetic. This allows to test the algorithms on the same kind of scene they have been trained or tuned with. Having better control on the data also prevents any experimental-related failures of the algorithms. In Section 2, we review surface reconstruction and present the algorithms that are evaluated in the present paper. In Section 3, we define our metrics and discuss their nature. In Section 4, we present the three datasets on which we tested our evaluation protocol and present the experimental setup we used to generate the high-quality data. Results are detailed and analyzed in Section 5, before we present the conclusions of our work in Section 6.

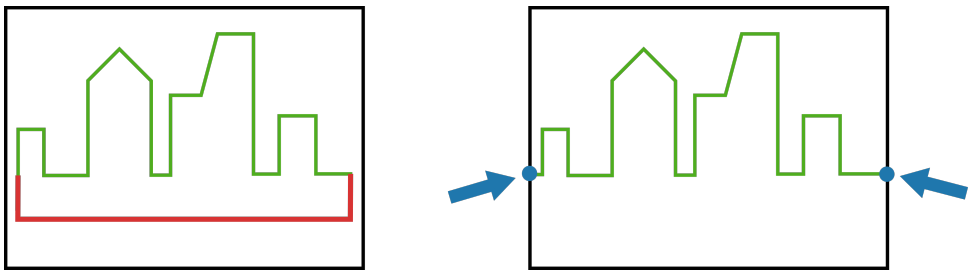


Figure 1: 2D Comparison between *hard* watertightness (left) and *soft* watertightness (right). The soft-watertight surface has a border (materialized by the blue dots) but only on the boundary of the domain. When trying to model urban environments, the red piece of surface from the hard-watertight surface does not have any significant meaning. This illustrates why soft-watertightness is better suited to open scenes.

2. Related Work

2.1. Surface Reconstruction

Here we review some existing methods to reconstruct a triangle mesh from point cloud and classify them by the paradigm they use. See [10] and [11] for an even deeper analysis of them (even though the most recent methods do not appear). Methods evaluated in Section 5 are typesetted in bold.

2.1.1. Indicator function

Often used to achieve watertight reconstructions, this class of algorithms proceed by computing a space segmentation. The object itself is defined as the region of space where the labeling equals a certain value. The surface is then computed by finding the changes in the segmentation. Popular approaches in this field are [12], **Poisson** reconstruction [13], and the differentiable Poisson solver that has been introduced in [14].

Recently, lots of *learning-based methods* have been proposed. While they often outperform non-learning-based ones on simple geometries, especially closed objects, they have not been proved as able to deal with the complexity of large and open scenes. *IM-NET* [15] is a learning framework which predicts whether any point (x, y, z) is inside or outside the given shape needing to be reconstructed. *Occupancy Networks* [16] presents a similar way of computing the so-called *occupancy function* of the 3D object. *Convolutional Occupancy Networks* [17] introduced a learning-based framework to compute implicit surfaces. Recently, [18] proposed a general learning framework dubbed *AtlasNet* to take as input a 3D point cloud or an RGB image. It proceeds by concatenating this data with a sampling of a *patch*, namely the unit square, before passing it to multilayer perceptrons (MLPs) with rectified linear unit (ReLU) nonlinearities, producing as output a point cloud of arbitrary resolution. A mesh can be generated either by

transferring the connections between vertices of a mesh defined on the patch to their 3D image points or by using Poisson Surface Reconstruction [13] on a sufficiently dense point cloud.

2.1.2. Volumetric Segmentation

This is a sub-discipline of indicator functions as it consists in giving information about whether a region of space is filled by the object or is empty. The data structure can be:

- *the Delaunay Triangulation* of input samples as in [19], [20], [21] and [22].
- *voxels*: [23] labels them as *free space*, *occupied* or *unknown*. To achieve this, point locations combined with sensor positions allow computing the ray corresponding to a beam of free space. An interesting feature is that undesirable moving objects such as humans can be erased in the final surface thanks to scans of the same area from different sensor positions.

Robust and Efficient Surface Reconstruction (RESR) [19] label as *inside* or *outside* each tetrahedron of the Delaunay triangulation of the point samples. The triangles separating an *empty*-labelled tetrahedron from an *occupied*-labelled one are extracted thanks to a graph-cut optimisation of an energy function (equation 1) defined thanks to the lines of sight (emanating from the vertex and pointing at the laser scanner) and the shape of the triangles.

$$l_T = \underset{l_T \in L^T}{\arg\min} \Big(E_{data}(l_T) \; + \; \lambda \; E_{prior}(l_T) \Big) \tag{1}$$

Delaunay-Graph Neural Networks (DGNN) [24] also use this paradigm (equation 1) but they estimate the occupancy of the tetrahedra thanks to a graph neural network.

2.1.3. Signed-distance function

Another way of generating a watertight surface is to compute the signed-distance function f to the surface and to extract its zero-level set. This is the approach chosen in Multi-level Partition of Unity (MPU) [25] and **Smooth Signed Distance (SSD)** [26].

Recently, *DeepSDF* [27] has shown how to learn the surface distance field.

2.1.4. Primitive-based

In this field, *PolyFit* [28] uses RANSAC [29] to detect planar segments and refine them. The surface is extracted by combining the optimisation of an objective function which favours data fitting, point coverage and model complexity and the enforcement of watertightness and manifoldness.

Point Set Structuring (PSS) [20] relies on the Delaunay triangulation of input points and the labelling of its tetrahedrons as *empty* or *occupied*, but their specificity resides in the extraction of primitives as a pre-processing step, a resampling of the resulting structures and the combination of points from planar regions and unstructured ones in the reconstruction step.

2.1.5. MLS-based

Moving least squares (MLS) was first introduced by Lancaster in [30], based on the work conducted by, amongst others, Shepard in [31]. Since then, a tremendous amount of extensions have been added as pointed out by a survey conducted in 2008 in [32]. For instance, [33], [34] and [35] significantly contributed to the advances in MLS-based algorithms. As explained in [32], MLS-based algorithms can be roughly classified into two main categories:

Implicit MLS surfaces require the computation of a level set function of which the zero isosurface can be extracted.

Projection MLS surfaces consist of first computing a projection operator that maps any point of the space to a point on the surface. The surface is then made of the set of stationary points.

2.2. Evaluation of Surface Reconstruction

In order to assess the quality of a reconstruction, there is need for a ground truth, an input point cloud and a means of calculating the difference between a given output surface and the so-called ground truth. Let us present the various possibilities that have so far been considered for these three aspects.

2.2.1. Ground Truth

Ground truth could potentially take any surface form, i.e. implicit field, triangle mesh, volumetric segmentation, point set, deformed model, skeleton curve, primitives. However, only two have so far been considered: **triangle mesh** [36], [12], [6] and **implicit field** [37].

2.2.2. Input Point Cloud

Producing point samples from a surface can be carried out in several ways:

- **Real scanning**: Based on a physical object (or scene), laser-based scanning generates a point cloud directly. Such technologies include Time-of-Flight [1] and Structured-light [2]

devices. In addition, terrestrial or airborne LiDAR [3] offer the possibility to deal with large areas.

- **Image-based**: Multi-view stereo [4] and Structure from motion [5] allow creating a 3D model from images, which can be the starting point for surface reconstruction.

- **Model sampling**: Based on a continuous digital input model, synthetic sampling has the advantage of making it possible to fully control the data. In particular, one can generate more realistic data by adding noise, outliers, misalignment and occlusions, and by setting the density. In this field, several procedures have been considered: **random** or **uniform sampling** [12], [38], [39], **synthetic raytracing** [40], [37], [6] or **z-buffering** [36]. Of particular interest is the recent work presented in [41], [6] and [42]. *Helios++* [41] is an open-source tool for the simulation of airborne, UAV-based, and terrestrial static and mobile laser scanning implemented in C++ . [6] provided an airborne LiDAR simulator and [42] developed *LiDARsim*: a virtual terrestrial LiDAR platform generating realistic point clouds based on a high-quality mesh, free of moving objects.

2.2.3. Comparison

With regards to comparing an output reconstruction, three main possibilities have been explored:

- **Visually**: Most of the time, surface reconstruction aims at producing a digital representation as visually similar as possible to a real object. Hence, Poisson [13], MPU [25] and SSD [26] have simply compared models on a visual basis. This obviously raises the issues of being sensitive to the observer's perception, conflict of interest and the lack of quantitative information.

- **Point-to-mesh distance computation**: When the only ground truth that is available comes in the form of a point cloud, it is relatively straightforward to compute the distance from each of those points to the reconstructed model. [9] have evaluated their method by randomly partitioning their point cloud into two equal-sized subsets: points serving as input for the reconstruction algorithms and validation points from which distances to the output meshes are computed.

- **Mesh-to-mesh distance computation**: This method comes with the advantage of providing a quantitative quality assessment which is independent of any human bias. [36], [9]

use the *Metro* tool [43] which works as follows: given two meshes (a sampled one $\mathcal{M}_s$ and a target one $\mathcal{M}_t$), *Metro* samples $\mathcal{M}_s$ and measures the shortest distance from each sample to $\mathcal{M}_t$. *Metro* then computes the mean distance, the max and the Root Mean Square (RMS) over all samples. [6] have used such a distance on the reconstructed and the ground truth meshes. Their specificity was to filter out triangles further than the input point cloud used for reconstruction. They thus produce a curve of distances for the different threshold values of this filtering procedure.

- **Mesh-to-implicit distance computation**: [37] chose to use an implicit field that we will call Ω as ground truth, and consequently they adapted the *Metro* methodology in order to compute the distance from a nearly uniform sampling of Ω to the evaluated mesh and vice-versa. The evaluation process answers the question: how well does the reconstructed mesh fit to the implicit surface computed by the MPU [25] algorithm? To address this issue, several measures are proposed by the benchmark [37]: Hausdorff distance (equation 4), mean distance (equation 5), max (equation 6) and mean angle deviation (equation 7). The former two allow us to know how *close* the two surfaces are to each other while the latter two give an insight into how similar the *local orientation* is.

  In order to compute these, defining point correspondences between the two surfaces are needed. Let us denote by M the implicit surface and by $\overline{M}$ the output triangle mesh. As defined in [44], the mapping $\Phi : M \longrightarrow \overline{M}$ attributes to one point $p \in M$ the intersection of the normal line through p and the mesh $\overline{M}$. The inverse mapping $\Phi^{-1} : \overline{M} \longrightarrow M$ attributes to $\Phi(p)$ its closest neighbor on the implicit surface M. This definition, associated with a sampling P_M of M produces a set of nearest neighbor correspondences that we call C_M (equation 2).

$$C_M = \{(x, p) | p \in P_M,\ x = \Phi(p)\} \tag{2}$$

By defining the corresponding operator $\Psi : \overline{M} \longrightarrow M$ and a sampling $P_{\overline{M}}$ of the reconstructed mesh $\overline{M}$, we get $C_{\overline{M}}$ (equation 3).

$$C_{\overline{M}} = \{(p, x) | x \in P_{\overline{M}},\ p = \Psi(x)\} \tag{3}$$

Denoting $|S| = |C_M| + |C_{\overline{M}}|$ and with $\gamma(p, x)$ the angle between the normals $N_M(p)$ and

$N_{\overline{M}}(x)$, error measures are the following:

$$H(M, \overline{M}) = \max \left\{ \max_{(x,p) \in C_M} |x - p|, \max_{(p,x) \in C_{\overline{M}}} |p - x| \right\} \tag{4}$$

$$\mu(M, \overline{M}) = \frac{1}{|S|} \left(\sum_{(x,p) \in C_M} |x - p| + \sum_{(p,x) \in C_{\overline{M}}} |p - x| \right) \tag{5}$$

$$H_N(M, \overline{M}) = \max \left\{ \max_{(x,p) \in C_M} \gamma(p, x), \max_{(p,x) \in C_{\overline{M}}} \gamma(p, x) \right\} \tag{6}$$

$$\mu(M, \overline{M}) = \frac{1}{|S|} \left(\sum_{(x,p) \in C_M} \gamma(p, x) + \sum_{(p,x) \in C_{\overline{M}}} \gamma(p, x) \right) \tag{7}$$

3. Evaluation protocol

3.1. Intuition

Let us give some examples of situations where current metrics are not adequate to evaluate surface reconstruction, and what we suggest would be an improvement. This is going to help understand our metrics' definitions in Section 3.3.

First, as presented in Section 2.2, comparing a reconstructed mesh with a ground truth point cloud can be done by computing the distances from those points to the mesh model. While this seems like a good starting point to assess how well holes have been filled around those points, it is inadequate to evaluate the overall accuracy of the surface. Figure 2 shows an example of such a situation where a surface would be evaluated as almost perfect even though large portions are clearly incompatible with the ground truth if we take into account the positions from which points have been acquired.

Secondly, it is possible to measure accuracy solely with ground truth points by sampling the reconstructed surface and measuring the distance from these samples to the ground truth points. Nevertheless, large pieces of the reconstructed surface might be judged as being of poor quality (if lying far from the nearest ground truth point) despite being correct, just because of a low ground truth density. We want to assess both accuracy and completeness only in regions

where ground truth information is available, and this is possible using sensor positions as shown in Figure 2.

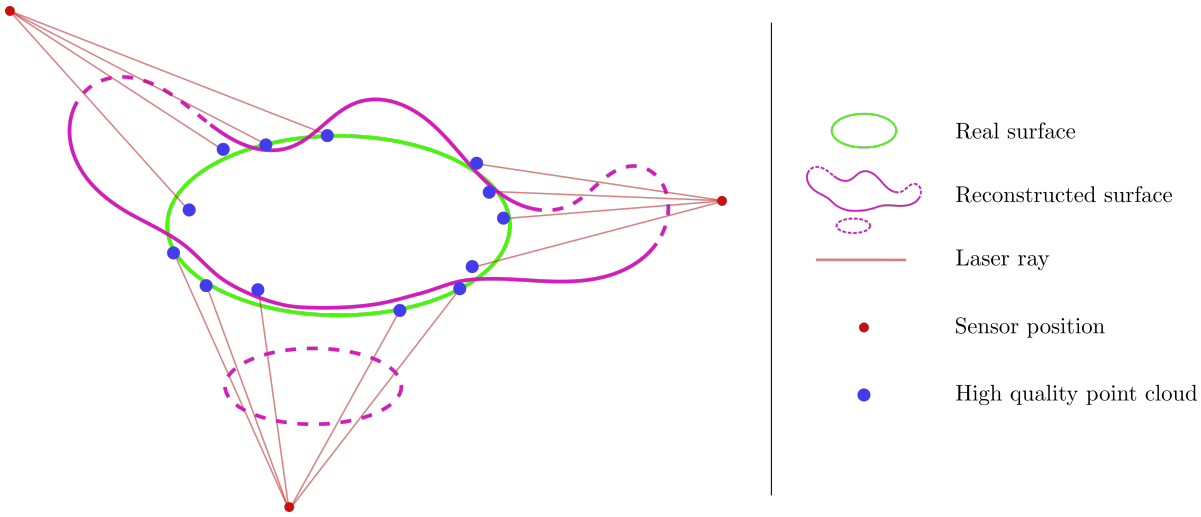


Figure 2: The importance of sensor positions: the dashed part of the reconstructed surface can be identified as wrong by making use of sensor positions when the high-quality point cloud does not provide enough information.

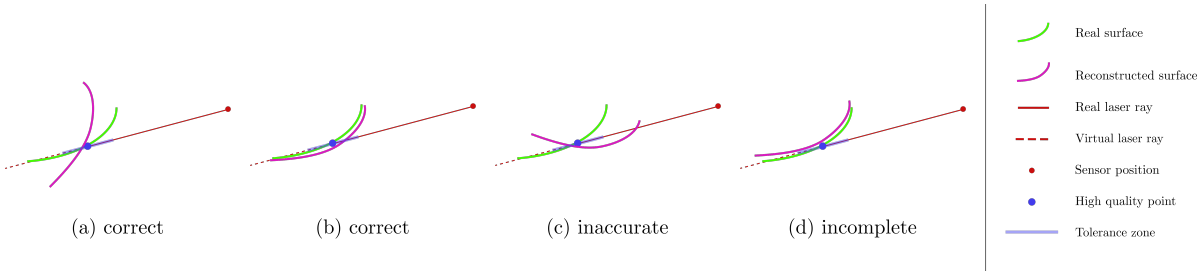


Figure 3: These four cases would be evaluated in the same way by a basic point-to-mesh distance. However, they are very different in terms of what a human being would be able to see from the sensor position.

Thirdly, as surface reconstruction has often been evaluated visually, we wanted to find metrics that would imitate this human intuition-based assessment. We believe that visibility-based metrics are more appropriate for assessing how the reconstructed surface matches the real one everywhere where we can *see* and compare them. Figure 3 shows four situations for which the piece of reconstructed surface would be marked similarly by a point-to-mesh distance. The distance from the high-quality point to the nearest piece of reconstructed surface is indeed the same in all four situations. However, we are certain that they should correspond to three completely different outcomes and we want our metrics to be able to differentiate between them. In particular:

- in (a), the reconstructed surface lies slightly *behind* the high-quality point. We consider it as correct at the threshold defined by our tolerance zone.

- in (b), the reconstructed surface lies slightly *in front of* the high-quality point. As in (a), we consider it as correct even though we can measure a slight error.

- in (c), whilst the piece of real surface corresponding the high-quality point has been recovered just like in (a), the reconstructed surface hides the nearest intersection by crossing the laser ray, resulting in an obvious inaccuracy. What one would see by looking in the direction of the laser ray is not the right piece of reconstructed surface but rather something else far in front of it.

- situation (d) might look similar to (a) but there is actually no intersection between the reconstructed surface and the laser ray. Consequently, what one would see by looking in the direction of the laser ray is not the right piece of reconstructed surface but something else in the background. We thus consider that the surface has not been recovered at all here.

ETH3D benchmark [45] proposed metrics to evaluate how well a two- or multi-view stereo point cloud matches a LiDAR-based one, using sensor positions. While surface reconstruction is a different task to MVS (in particular in terms of the expected output properties), [45] still evaluates the matching between two 3-dimensional structures and we were inspired by their use of sensor positions. However, we chose different paradigms that are more adapted to surfaces, so our evaluation protocol is considerably different.

The *ETH3D* benchmark [45] defines *completeness* as the proportion of ground truth points for which the distance to its closest reconstructed point is below a given threshold. We could keep this definition and find the point from the reconstructed surface that minimises the distance to each point of the high-quality point cloud. Nonetheless, given that we know in what direction this point is supposed to be encountered thanks to the sensor position, we find it more relevant to compute the distance between the high-quality point and the nearest intersection between the corresponding laser ray and the mesh model. In other words, while the most natural adaptation of this point-to-point distance would be a point-to-mesh distance, we believe that for each ray that hit the the real surface there should be a piece of reconstructed surface close by and *along the ray*.

However, we also leverage the information given by each laser ray: the space between each

sensor position and its associated high-quality point should be *empty*. We soften this property by defining a tolerance zone: a piece of reconstructed surface will be considered as correct if its distance along the ray from the high-quality point is smaller than a given threshold d_{max}. Every piece of surface further than d_{max} and situated *in front of* the high-quality point will affect the accuracy of the model.

Contrary to *ETH3D* benchmark [45], we do not model the shape of a laser beam as a truncated cone. The first reason for this is that we do not *need* to, since the model we are trying to evaluate (a surface) is continuous instead of discrete (a point cloud). Hence, we are not at risk of missing any part of it. In addition, it makes it simpler to get a point as the intersection between a ray and a piece of surface. This way, every couple (ray - piece of surface) gives the same amount of information.
[45] uses voxels to prevent a "cheating" strategy from achieving both high accuracy and completeness despite raising other issues. For example, regions of low ground truth density contribute as much as high density ones while encapsulating less information. A cheating strategy for surface reconstruction would be to add several parallel layers of surface in regions of high confidence. We do not need to discretise space as in [45] since for each ray we propose to keep only the closest intersection as a potential correct one and penalise all the ones situated in front of it. Note that even layers situated behind the closest intersection might be obstacles to rays pointing at another object in the background.

If the nearest intersection is found *behind* the high-quality point at a greater distance than d_{max}, or if no intersection is found at all, then we consider that this piece of surface has not been recovered. This therefore affects the completeness of the model.

3.2. Definitions and notations

In this paper we want to evaluate the quality of surface reconstructions from low-quality data $\mathcal{P}_{LQ}$, with only having access to high-quality data $\mathcal{P}_{HQ}$ of the same scene and without having access to the perfect ground truth surface that the algorithms are supposed to produce.

- $\mathcal{P}_{LQ}$: the *low-quality* point cloud that will be fed to the evaluated surface reconstruction methods to produce the output surface meshes to be evaluated.

- $\mathcal{P}_{HQ}$: the *high-quality* (ground-truth) point cloud with better coverage, higher density and less noise than $\mathcal{P}_{LQ}$ and for which we know the sensor positions, defining one ray per point.
- $\mathcal{M}_E$: the reconstructed mesh to evaluate, produced by an algorithm from $\mathcal{P}_{LQ}$.
- d_{max}: the maximum distance at which we evaluate the reconstruction. It is a parameter which influences the different metrics as we use it to separate noise (distance $< d_{max}$) from outliers (distance $> d_{max}$).

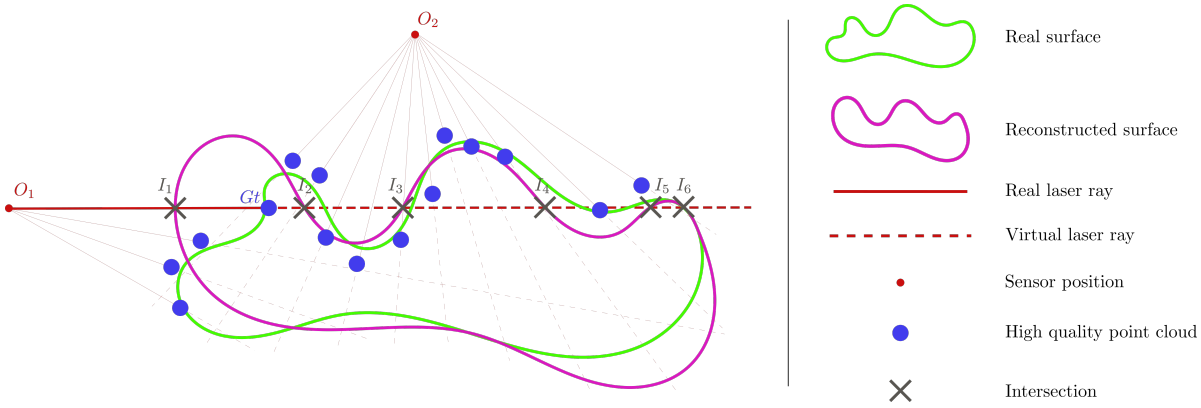


Figure 4: Toy example to visualize the definitions of the metrics. The real surface has been scanned from two positions: O_1 and O_2. A given real laser ray (the thick one) was cast from O_1 and it hit the real surface at Gt. Note that the position of the intersection might be noisy, hence the shift between the real surface and the high-quality point cloud. We compute all intersections between the associated virtual ray (i.e. the extension of the real laser ray and the reconstructed surface. In this case, it results in six intersections $I_{1,\dots,6}$. The closest intersection to Gt happens to be I_2 so the "**ray distance**" metric for that particular ray is the distance (Gt, I_2). We found one intersection I_1 *on the way to* the closest intersection I_2 which is counted as a **FP**. Secondly, if $(Gt, I_2) < d_{max}$, Gt is to be counted as a **TP**, otherwise it will not be taken into account in the evaluation since it is situated *after* Gt. Note that this piece of surface might still be evaluated thanks to another ray (as one emanating from O_2, for example).

3.3. Metrics definitions

Similar to [6], we would like to assess both the precision of each part of the reconstructed surface (each part of the surface should lie near some part of the real surface) and the completeness of the model (there should be as few missing parts of the real surface as possible). As we do not have access to a digital model of the real surface, we cannot compute the actual *precision* and *recall* as in [6]. Our knowledge of the ground truth is limited to the high-quality data $\mathcal{P}_{HQ}$. However, we also know where the surface is **not** supposed to lie since we know the sensor position from which every point of $\mathcal{P}_{HQ}$ has been acquired, thus defining a ray of free space. Therefore, we define a *precision* metric that penalises inconsistencies between the

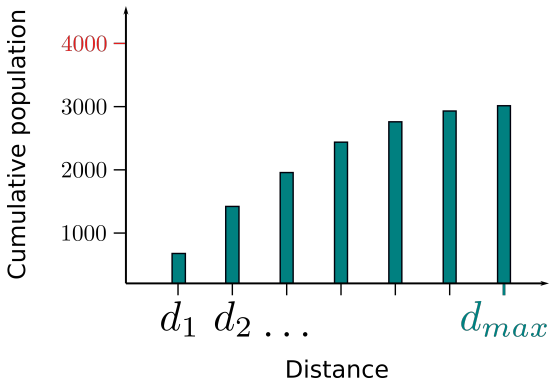
reconstructed surface and the visibility information contained in $\mathcal{P}_{HQ}$. We propose an equivalent of the *recall* metric: for each $[OP]$ ray from $\mathcal{P}_{HQ}$, we compute the distance from the P to the closest intersection between the $[OP)$ half line and the reconstructed surface. Here is the formalisation of these metrics in more detail:

- "Ray distance": for each point/ray $(p, r) \in \mathcal{P}_{HQ}$, we compute the distance from p to the closest intersection (which we will denote as c) between r and $\mathcal{M}_E$ (among all potential intersections, we choose the one with the smallest distance with respect to p). If this distance is $< d_{max}$ then the piece of reconstructed surface is considered as being *correct* and we add this distance to an array of distances.

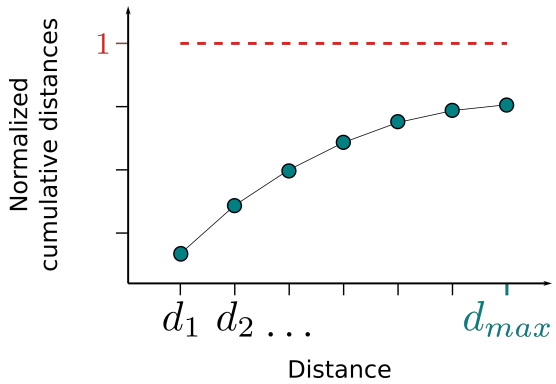
- "Precision": for each point/ray $(p, r) \in \mathcal{P}_{HQ}$, if the "ray distance" (between p and c) is $< d_{max}$ then we count c as a *True Positive (**TP**)* (since there is a piece of surface and it is correct). Otherwise, if the intersected point c lies at a distance greater than d_{max} and it is *before* the corresponding high-quality point p, we consider it as being false and we count it as a *False Positive (**FP**)* (since there is a piece of surface and it is false). We consider that we cannot say anything about the closest point c if it lies at a greater distance than d_{max} and it is situated *after* the corresponding high-quality point p (neither can we for all intersected points lying after it), so we just ignore them. All intersected points lying *before* this closest intersection c are also counted as ***FP*** as they are inconsistent with the corresponding ray of free space (p, r). This is enough to define the precision ratio (equation 8).

- "Recall": it is defined as the ratio between the number of ***TP*** and the number of cast rays (equation 9. Every high-quality point $p \in \mathcal{P}_{HQ}$ is either mapped to its corresponding ***TP*** (in which case the piece of real surface has actually been recovered by the algorithm) or not (meaning a lack of exhaustiveness of the reconstruction and corresponding to a *False Negative* ***FN***). The number of rays thus equals the sum of the ***TP*** and the ***FN***.

- "Cumulative distances": the cumulative histogram of the ray distances where the x-axis corresponds to the distance and on the y-axis we plot the number of points for which the ray distance is below the x-distance, divided by the total number of rays cast. It contains the information of both recall (the right-most value) and mean ray distance. Figure 5b shows an example of this histogram.



(a) Cumulative histogram: each bar indicates the total number of points for which the ray distance is smaller than the corresponding distance (d_1, d_2, ... d_{max}).



(b) Normalized cumulative histogram: the value of each bar from graph (a) is divided by the total number of rays. The value corresponding to d_{max} is the recall.

Figure 5: Construction of the "cumulative distances" metric. In this example, we assume that 4000 rays have been cast. 3000 intersections correspond to True Positives (their ray distance is smaller than d_{max}). The distribution of these 3000 ray distances is shown in graph (a). Cumulative population values cannot be compared between different datasets, so we normalize them by dividing them by the total number of cast rays from the corresponding dataset. This leads to the normalized "cumulative distances" shown in graph (b).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{8}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{Number of rays}} \tag{9}$$

$$\text{F-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{10}$$

As we define precision and recall as ratios, the harmonic mean (F-score, see equation 10) allows ranking the methods by taking into account both metrics.

3.4. Tuning / Training

The algorithms we evaluate in this paper are either *tunable* for the most part or they *learn* parameters in order to reconstruct surfaces. For example, **DGNN** [24] needs a training dataset in order to learn the parameters of its model and **Poisson** [13] can be run at different resolutions by changing the *depth* of the octree that is used. In order to be as fair as possible, we used the same dataset to *tune* or *train* the algorithms. We therefore ran the non-learning-based methods with different values for their parameters and carried out an evaluation. A first interesting result is that for some methods, it can be hard to obtain a good performance regarding the *precision* and the *recall* metrics at the same time. For **PSS**, the higher the value of the *trade-off* parameter λ (therefore the more importance given to the prior term, see equation 1), the higher the precision (up to a certain value) but the lower the recall. Maximizing the score of one metric (by varying λ) results in minimizing the score of the other one (we explain why in Section 5). Thus, we selected the two λ values that maximize each of these metrics individually.

The *tuning/training* dataset that we decided to use is composed of three scenes from *STRAS*. The reason behind this choice is the availability of ground-truth meshes which are absolutely necessary for the training phase of **DGNN**. We then also used these three scenes to find the best parameters (in terms of *ray distance*, *precision* and *recall*) for the non-learning-based algorithms.

4. Input data

We aim to evaluate the algorithms in very different scenarios as the methods' priors might influence the quality of the reconstruction depending on the type of scene or the type of data involved. We therefore compute the metrics introduced above on three significantly varying datasets.

4.1. STRAS: Strasbourg dataset and LiDAR simulator

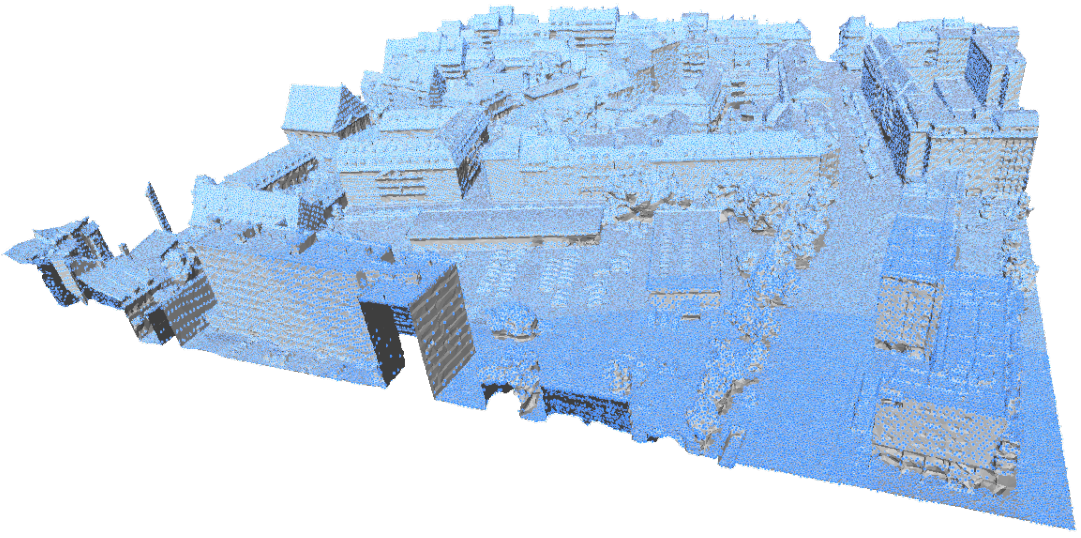


Figure 6: Strasbourg scene (mesh in grey, point cloud in blue)

In order to control the data itself, we started by using a synthetic dataset [1] taking the form of a high-quality mesh representing a large area covering the "Métropole de Strasbourg". Figure 6 shows a 250 meter by 250 meter tile of this mesh. In order to generate the ground-truth and the input point clouds $\mathcal{P}_{HQ}$ and $\mathcal{P}_{LQ}$, we used the **aerial LiDAR simulator** from [6]. This is because such a large urban environment is typically scanned using an airborne LiDAR system. Their code offers the possibility to simulate a plane flying above the scene with the laser ray

[1] The dataset is available at: 3d.strasbourg.eu. It was produced by "Ville et Eurométropole de Strasbourg" with financial support from the European Union as part of a "Fonds Européen de Développement Régional".

of a LiDAR system following a parallel line pattern. Realistic noise can be added as a post-processing step to imitate typical devices from the market. The problem with the parallel line pattern is that facades that are perpendicular to the direction of the plane are not reachable by the laser ray and thus are absent from the resulting point cloud. In order to overcome this issue, we implemented the **elliptical scanning pattern** . It is indeed better suited to urban environments, for the laser ray will be able to point at far more facades than with the parallel line pattern, resulting in a better coverage.

4.1.1. Elliptical aerial LiDAR simulator

We use $(O, \vec{e_x}, \vec{e_y}, \vec{e_z})$ as the global coordinate frame, in which mesh vertices coordinates are expressed as shown in Figure 7. We model the acquisition by a linear trajectory of the LiDAR optical centre M moving straight from $A\,(x_A, y_A, z_A)$ to $B(x_B, y_B, z_B)$ at constant speed v_0. We define a local coordinate frame $\left(M, \vec{i}, \vec{j}, \vec{k}\right)$ associated to M defined as:

$$\vec{k} = \frac{\overrightarrow{AB}}{\left\|\overrightarrow{AB}\right\|} \;; \quad \vec{j} = \frac{\vec{e_z} \wedge \vec{k}}{\left\|\vec{e_z} \wedge \vec{k}\right\|} \;; \quad \vec{i} = \vec{j} \wedge \vec{k} \tag{11}$$

Denoting $\vec{r}$ as the direction of the laser ray, and using $(M, \vec{u}, \vec{v}, \vec{w})$ as the canonical spherical coordinate frame, $\vec{r}$ is rotating around $\vec{w}$ at constant angular speed $\omega = \dot{\phi}$ with ϕ as the azimuthal angle of $\vec{r}$. The polar angle θ is constant. In accordance with [6], the noise that is added to the point positions follows a normal distribution that we can split between a planimetric $\Delta x, \Delta y$ and altimetric Δz component:

$$\Delta x, \Delta y \sim \mathcal{N}\left(\mu_{xy}, \sigma_{xy}^2\right) \;; \quad \Delta z \sim \mathcal{N}\left(\mu_z, \sigma_z^2\right) \tag{12}$$

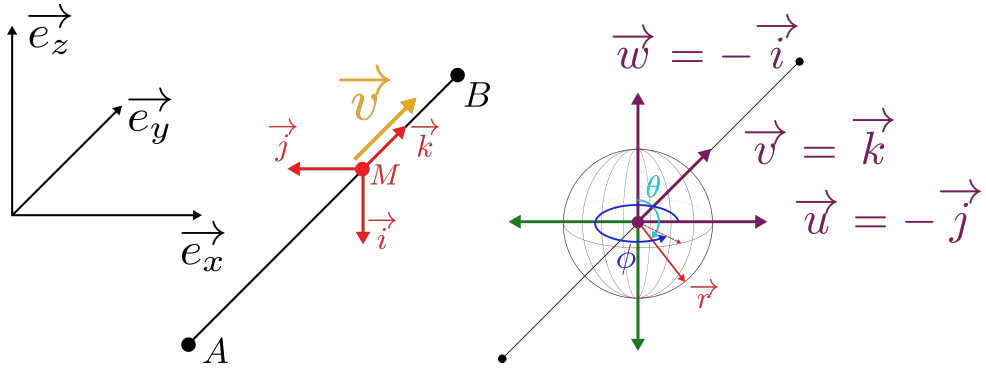


Figure 7: Elliptical scanning pattern.

The values of all the parameters we used can be found in Table 1. We chose these values based on the default values of a real aerial LiDAR system (*Leica TerrainMapper*). In the future, we intend on fine-tuning this simulator to maximize point coverage.

Table 1: Values of experimental parameters used.

Symbol	Value	Unit	Description
h	1 000	m	Flying altitude
v_0	60	$m.s^{-1}$	Flying speed
ω	150	Hz	Angular speed
θ_0	160	$(^\circ)$	Polar angle
f_p	400 000	Hz	Pulse frequency
σ_{xy}	0.13	m	Planimetric error
σ_z	0.05	m	Altimetric error

4.1.2. Experimental setup

We aim to produce two point clouds $\mathcal{P}_{HQ}$ and $\mathcal{P}_{LQ}$ in such a way that $\mathcal{P}_{HQ}$ should have less occlusions and be denser than $\mathcal{P}_{LQ}$. The scenes are all 250 meter by 250 meter tiles of an urban environment for which positions are expressed in a global coordinate frame $(O, \overrightarrow{e_x}, \overrightarrow{e_y}, \overrightarrow{e_z})$ such that $\overrightarrow{e_z}$ represents the ascending vertical direction. In our setup, a trajectory of the LiDAR system for each scene is a single straight pass of the plane along axis $\overrightarrow{e_y}$ for $x = \alpha_x \left(x_{max} - x_{min}\right)$, $\alpha_x \in [0, 1]$. We generate $\mathcal{P}_{LQ}$ thanks to one pass of the plane with $\alpha_x = 0.5$ and $\mathcal{P}_{HQ}$ thanks to three passes with $\alpha_x \in \{0.25, 0.5, 0.75\}$. We denote P_{α_x} as the point cloud resulting from the flight $x = \alpha_x \left(x_{max} - x_{min}\right)$. We then have: $\mathcal{P}_{LQ} = P_{0.5}$ and $\mathcal{P}_{HQ} = P_{0.25} \cup P_{0.5} \cup P_{0.75}$. This way, $\mathcal{P}_{LQ}$ forms part of $\mathcal{P}_{HQ}$ and contains a lot more occlusions in particular on facades parallel to the direction of the plane. Figure 8 gives an example of such a situation.

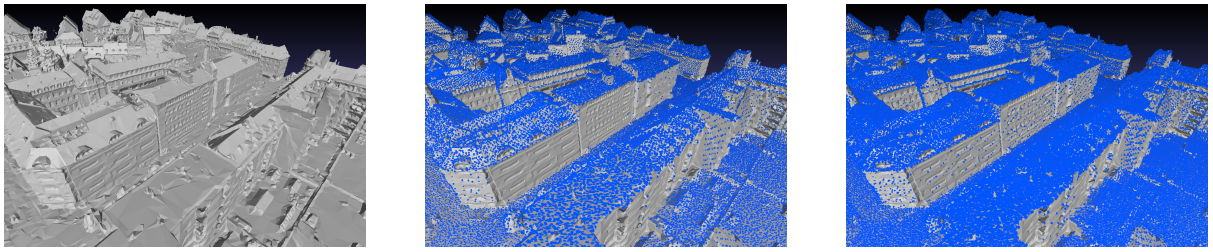


Figure 8: Left: Mesh, Center: $\mathcal{P}_{LQ}$, Right: $\mathcal{P}_{HQ}$. The facades parallel to the direction of the plane are a lot more occluded in $\mathcal{P}_{LQ}$.

4.2. ENSG dataset: indoor and outdoor terrestrial LiDAR scan

As the original goal of our study was to propose an evaluation protocol suited for real data, this dataset is only based on real data that we acquired ourselves. The resulting point clouds intensity channel can be visualized on Figure 9.

4.2.1. Stationary LiDAR station

We used the stationary LiDAR station "*Leica ScanStation P40*" for which we will give a brief introduction. Once the station is settled, rays are cast 360° horizontally around its origin and 290° vertically (the ground area immediately underneath the station remains unobserved during the acquisition). It can acquire up to $1,000,000$ points per second from 0.4 meter to 270 meters with a 3D position accuracy of 3 millimeters at 50 meters. In order to satisfy the condition of $\mathcal{P}_{HQ}$ being of higher quality than $\mathcal{P}_{LQ}$, we decided to acquire points from more viewpoints to generate $\mathcal{P}_{HQ}$, using the same LiDAR station. We thus scanned each environment from five positions: the four vertices of an approximately regular 1.5 meters side length tetrahedron (O_1, O_2, O_2, O_4) and its center of mass O_5 as shown in Figure 10. In order to evaluate surface reconstruction algorithms in several different scenarios, we repeated this procedure for three scenes:

- "*Building*": an outdoor scene made of a building, a sloped road, trees and an opening to another scene called "*Parking Lot*".
- "*Parking Lot*": an indoor scene with pipes, partially occluded cars and open doors, including one communicating with the outdoor scene "*Building*".
- "*Clutter*": a closed indoor scene with a lot of occlusions due to a high density of objects.

4.2.2. Matrix format and sub-sampling

Following the definition of our protocol (see Section 3), we need to generate poorer-quality point clouds to run reconstructions. Our LiDAR system has a spherical geometry . The horizontal resolution and vertical resolution of the scanner define a fixed number w of values for ϕ and a fixed number h of values for θ respectively. Laser rays are thus cast in the directions given by every pair of angles (ϕ, θ). We can thus represent the points as a matrix of *height* h and *width* w and then index all points by their $(i, j) \in [|1, h|] \times [|1, w|]$ coordinates. Figure 9 shows the intensity of the returns in this matrix-like format. Each raw acquired point cloud did not

fit into the memory of our machine so we down-sampled them by keeping odd-indexed points . Starting from the raw point clouds with origins centered on O_1, O_2, O_3 O_4, and O_5 respectively, we down-sampled them all (roughly four times) following this matrix-based scheme and $\mathcal{P}_{HQ}$ is the union of these five four-time down-sampled point clouds. We repeated this matrix-based down-sampling scheme on the point cloud centered on O_5 to generate $\mathcal{P}_{LQ}$.

4.3. ETH3D dataset

[45] presents a two- and multi-view stereo benchmark. Their dataset contains several scenes with:

- input images at 24 Megapixel resolution on several scenes
- ground truth 3D laser scan point clouds

4.3.1. Generating the point clouds: Multi-View Stereo and real LiDAR

In order to generate the low-quality data, we used the OpenMVS [46] library to generate dense point clouds from images using the provided camera poses of three scenes (*Terrace*, *Courtyard*, *Pipes*) of the ETH3D train dataset. We used the DensifyPointCloud tool of OpenMVS with the standard settings, except for the following parameters: *number-views-fuse = 2*, *optimize = 0* and *resolution-level = 4*. We used the provided LiDAR point clouds as our high-quality data. A typical MVS pipeline generates a much sparser and more noisy point cloud than what a laser scan provides. It also contains more outliers. For all these reasons, we consider it relevant to carry out an evaluation on a set of MVS-based point clouds. We thus used three scenes from the *ETH3D* dataset [45], which can be seen in Figure 12.

5. Results

In this survey, we assessed:

- **RESR** [19]
- **SSD** [26] with two combinations of the octree *depth* and the B-spline *degree* parameters: $(depth = 8, degree = 2)$ and $(depth = 12, degree = 3)$.

- **Poisson** [9] with two values for the octree *depth* parameter: 8 and 11. The B-spline *degree* will always be 2.
- **DGNN** [24]
- **PSS** [20] with two values for the *trade-off* parameter: 0.1 and 0.6
- **Agisoft Metashape** 1.6.4 (the user manual can be found here) with *Extrapolated mode* and *ultra high* resolution. Only the **meshing tool** has been used on the point cloud data.

For the *ETH3D* dataset only, we assess two other surface reconstruction algorithms as they are part of the OpenMVS [46] pipeline. Mesh reconstruction is initiated using **Exploiting Visibility Information in Surface Reconstruction to Preserve Weakly Supported Surfaces** (**WSS**) [47]. A refinement step is then carried out using **High Accuracy and Visibility-Consistent Dense Multiview Stereo** (**DMS**) [48]. We thus computed the metrics on the resulting meshes from both these methods. According to the definitions provided in Section 3.3, Tables 2, 3, 4 and 5 give the mean ray distance (for those smaller than d_{max}), the precision and recall ratios as well as the F-score.

Table 2: Raw numerical results for $d_{max} = 50\,cm$ (MD stands for mean ray distance, P for precision, R for recall and F1 for F-score)

STRAS PC3E44_3 ($d_{max} = 50\,cm$)							
Method	TP	FP	TP + FN	MD (cm)	P (%)	R (%)	F1(%)
RESR	1278734	57089	1365120	6.58	95.73	93.67	94.69
DGNN	1246228	72899	1365120	6.79	94.47	91.29	92.85
Poisson 11/2	1267301	128888	1365120	8.91	90.77	92.83	91.79
PSS 0.6	1217138	81549	1365120	7.39	93.72	89.16	91.38
SSD 12/3	1271969	172285	1365120	9.56	88.07	93.18	90.55
SSD 8/2	1203150	167204	1365120	13.04	87.80	88.14	87.97
Poisson 8/2	1174069	151134	1365120	12.77	88.60	86.00	87.28
A. Metashape	1033179	195725	1365120	19.41	84.07	75.68	79.66
PSS 0.1	1280230	686825	1365120	7.47	65.08	93.78	76.84

5.1. STRAS dataset

Table 2 shows the results for one single mesh from the *STRAS* dataset and Table 3 shows the average of those results for the three meshes of *STRAS* dataset. In accordance with the survey conducted and published in [6] on the same dataset but with different assumptions and metrics, Table 3 shows that **RESR** achieves the best performance again on the urban environment from *STRAS* regarding both precision and recall. As evaluating surface reconstruction from real

Table 3: Average numerical results on the three scenes from *STRAS* dataset sorted by decreasing F-score for $d_{max} = 50\,cm$

STRAS ($d_{max} = 50\,cm$)				
Method	mean distance (cm)	precision (%)	recall(%)	F-score(%)
RESR	**5.98**	**96.68**	94.88	**95.77**
DGNN	6.13	96.08	92.56	94.29
Poisson 11/2	7.99	93.19	94.31	93.75
PSS 0.6	6.67	95.08	91.35	93.17
SSD 12/3	8.53	90.72	94.63	92.63
SSD 8/2	11.57	90.11	90.48	90.30
Poisson 8/2	11.49	91.11	88.39	89.73
A. Metashape	16.21	87.92	80.63	84.11
PSS 0.1	6.76	72.72	**95.00**	82.28
mean methods	9.04	90.40	91.36	90.67

data only is harder than using synthetic data (we do not have an exhaustive ground truth), it is a very sound validation that the metrics that we defined without access to the ground truth surface show similar tendencies to the metrics that are based on ground truth surfaces.

We also carried out a more detailed evaluation by testing several values for the main parameters of selected methods. We found that the *trade-off* of **PSS** [20] has a big effect on the metrics. More precisely, the lower we set it (the more confidence we give to the data), the lower the precision but the higher the recall (and vice versa). A high confidence in the data results in a lot more interfaces between occupied tetrahedra and empty ones. Conversely, a higher *trade-off* λ gives more power to the regularization term, resulting in fewer couples of adjacent tetrahedra being labeled differently, and so fewer triangles in the output mesh. When more confidence is given to the data, there are a lot more undesired triangles "floating" in regions of free space, which dramatically affects the precision. However, small structures might be erased from the mesh if less confidence is given to the data term, resulting in a poorer recall.

Poisson [13] and **SSD** [26] are both influenced positively by an increase in the octree depth. This was expected since more points are used to reconstruct the mesh, which results in an increase in the computation time and memory footprint.

DGNN performs a lot better on the *STRAS* dataset than on the two others, which highlights a problem in its capacity to generalize to scenes that differ from the ones in the training set. However, its poorer F-score performance on *ETH3D* and *ENSG* is mostly due to the

precision metric. **DGNN** often succeeds in recovering the scene features but adds too many undesired triangles in the scene.

While precision, recall and ray distance provide complete information on the quality of the reconstruction, one might find it more intuitive to start by having a look at the cumulative distances shown in Figure 13. The precision at small range can be estimated as the area under the curve. The closer the curve is to the top left-hand corner, the better it is since this means that all the *True Positives* are actually very close to it. Besides, the highest value of each curve is the recall of the corresponding method so the gap between the cumulative population in the last category and the line $y = 1$ should be as small as possible.

5.2. ENSG dataset

Table 4: Average numerical results on the three scenes from *ENSG* dataset sorted by decreasing F-score for $d_{max} = 20\,cm$

ENSG ($d_{max} = 20\,cm$)				
Method	mean distance (cm)	precision (%)	recall(%)	F-score(%)
RESR	**0.45**	**93.10**	95.99	**94.51**
Poisson 11/2	0.90	78.38	96.96	86.22
Poisson 8/2	2.72	78.66	88.16	83.05
SSD 12/3	1.27	72.41	96.04	82.23
A. Metashape	1.63	79.00	83.95	81.37
SSD 8/2	3.04	71.95	87.77	78.96
DGNN	0.49	52.99	96.22	68.28
PSS 0.6	0.54	28.44	97.92	43.94
PSS 0.1	0.54	22.30	**98.19**	36.18
mean methods	1.29	64.14	93.47	72.75

The *ENSG* dataset, having been generated using a stationary LiDAR system, is the one containing the least amount of noise hence the overall good performance of all of the methods. In particular, we can see that the mean distance is generally a lot smaller than with other datasets even though the scenes themselves have much more complicated geometries and more occlusions.

Figure 14 shows the meshes reconstructed by every assessed algorithm on the **Parking Lot** scene (part of the *ENSG* dataset). One can fairly easily interpret the performance achieved by these methods by analysing the type of mistake they made on the corresponding scene. **RESR** succeeds at reconstructing most of the visible parts, and very few undesired triangles lie in free space (most of them are connecting the pipes to the wall and the ceiling).

At first glance, **Poisson** 11/2 reconstruction seems to be a lot more accurate than **Poisson** 8/2 so it might not be obvious why they have the same precision. This situation actually shows the interest of the mean distance metric. While the two reconstructed models are structurally the same, the difference between them is visible at close range: under the threshold d_{max}. Consequently, **Poisson** 8/2 has a much higher mean distance than **Poisson** 11/2 but achieves a similar precision. The same kind of argument holds for explaining the relatively poor performance of **SSD** 12/3 and **A. Metashape**: whilst being *locally* more accurate than **Poisson** 8/2, the meshes are *structurally* not in accordance with the visibility information provided by the high-quality point cloud. The precision metric is dramatically affected by large portions of surface lying in free space. **DGNN** and **PSS** have the same problem: whilst having a high recall, denoting their capacity to recover most of the existing pieces of surface (and very accurately, given the very low mean distance metric), they connect too many regions of space with triangles lying in empty space, thus affecting their precision.

The accordance between all of these visual observations and the corresponding quantitative results given by our metrics prove their relevance.

5.3. ETH3D dataset

Table 5: Average numerical results on the three scenes from *ETH3D* dataset sorted by decreasing F-score for $d_{max} = 20\,cm$

ETH3D ($d_{max} = 20\,cm$)				
Method	mean distance (cm)	precision (%)	recall(%)	F-score(%)
DMS	**2.04**	**95.33**	93.72	**94.47**
Poisson 11/2	2.56	93.91	94.04	93.96
RESR	2.62	94.12	92.65	93.29
A. Metashape	2.57	95.09	91.18	93.00
WSS	2.66	91.31	93.79	92.51
SSD 12/3	2.66	90.65	94.44	92.48
SSD 8/2	4.21	93.38	89.03	91.13
Poisson 8/2	3.99	94.66	86.94	90.59
DGNN	2.61	79.47	93.63	85.95
PSS 0.6	2.52	67.41	94.63	78.52
PSS 0.1	2.56	53.79	**95.16**	66.99
mean methods	2.82	86.28	92.66	88.44

MVS-based point clouds are known to be noisy and contain quite a lot of outliers. This seems to have an effect on the performances of the different methods. The ones performing best on *ENSG* and *STRAS* seem to struggle more, and surprisingly, **Poisson** 8 [13] achieves a fairly high precision on this dataset. We believe that this is because it is more capable of filtering out

the noise with an 8-depth than with an 11-depth octree. That would explain why **SSD** 8/2 also has a better precision than **SSD** 12/3. However, their poorer recall indicates that more pieces of real surface have not been recovered.

The method that performs best, however, is **DMS**, which is not very surprising considering the fact that it is the best version of a real MVS pipeline, fed with images and not with an image-derived point cloud.

The relatively good performance of **A. Metashape** on *ETH3D* compared to the other datasets might suggest it copes pretty well with outliers. More generally, considering it is a licensed solution, we might have expected a better overall performance on at least one of our datasets.

5.4. General remarks

Overall, **RESR** is the method that performs best almost everywhere. **DGNN** shows that learning how to reconstruct large, complex and open scenes is indeed possible, but it faced generalisation problems since the metrics on *ENSG* and *ETH3D* datasets are significantly lower than those on *STRAS* (from which its training set was extracted). However, with **RESR** and **DGNN** being the only methods from this survey making use of sensor positions, we believe that this is an important reason behind their good results. Sensor positions give an important piece of information that neither the points themselves nor the associated normals provide.

Poisson generally performs structurally better than **SSD**. Small-scale differences are noticeable when changing the octree depth used by both these algorithms.

PSS often reconstructs meshes very close to the real surface but also connects pieces of surface in regions of space that should remain empty. We can assume that we failed to find the right parameter settings because it was definitely the hardest algorithm to tune, but this is the best performance we managed to get.

6. Conclusion

Surface reconstruction is hard to evaluate since it is impossible to directly compute the difference between the real surface and a reconstructed one. It has often been assessed visually because it seems fairly intuitive to know whether a piece of surface has been accurately recovered. However, human perception can be unfair and a purely visual evaluation lacks quantitative information. In this paper, we proposed new metrics to assess surface reconstruction. We have leveraged the visual information obtained by combining the acquired points and the associated sensor positions in order to define what we believe are more relevant metrics than the ones currently used. They imitate the process of a human being looking at and comparing the two surfaces (the real one and the reconstructed one). This goal has been achieved since our survey validates behaviours that a human can interpret by just looking at the meshes. In Section 5 we drew parallels between the specific visual observations and the quantitative evidence provided by our metrics that confirms them.

Our metrics enable the assessment of the completeness of the reconstructions as well as their precision both *locally* and *globally*. One can thus analyse the results from different points of view. As a relevant outcome, our survey also confirms that sensor positions are very relevant when trying to separate occupied from empty space.

As well as all these advantages that make surface reconstruction evaluation more objective, the fact that we only use raw data acquired with basic sensors makes it easy to set up a new experiment. Having access to expensive data is not a requirement. We provide a tool to make surface reconstruction evaluation easier and wish to see it used widely.

Figure 9: Equirectangular projection of the LiDAR points. **Top**: Outdoor scene ("*Building*") from O_3 - **Middle**: indoor scene ("*Parking Lot*") from O_2 - **Bottom**: Indoor Scene ("*Clutter*") from O_1. "*Building*" and "*Parking Lot*" share some space thanks to what can be seen through the open door in the middle of both images.

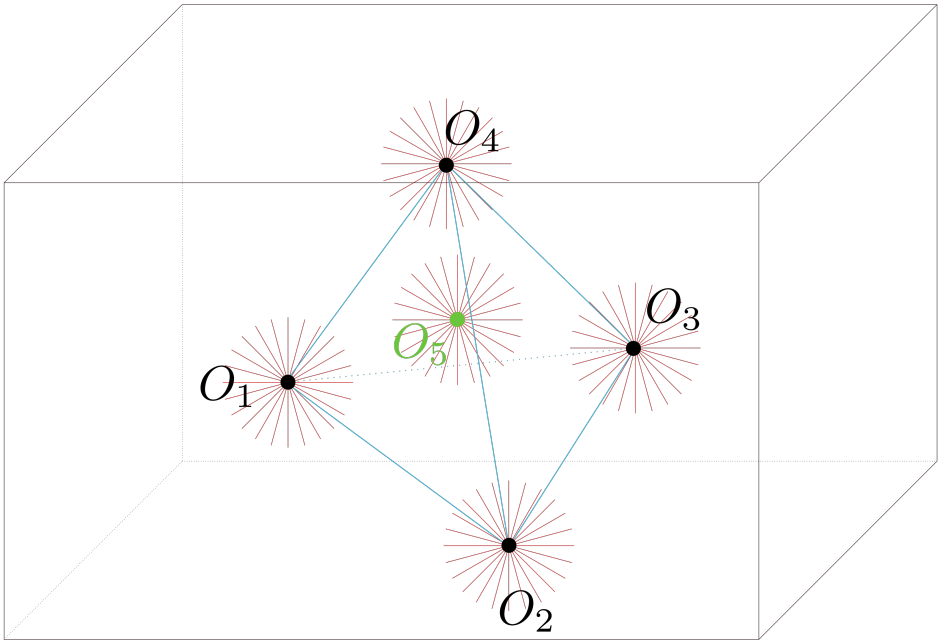
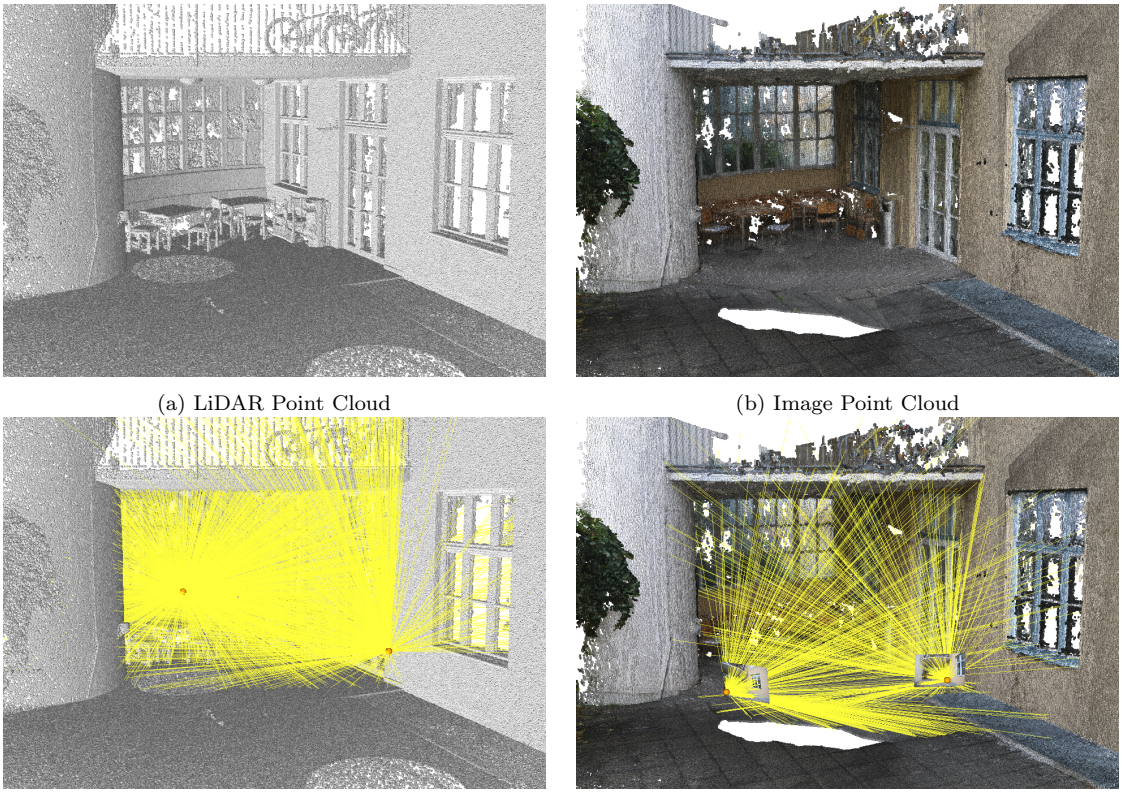


Figure 10: Tetrahedron-like viewpoints. (O_1, O_2, O_2, O_4) form an approximately regular 1.5 metres side length tetrahedron and O_5 is positioned at its centre of mass. The black rectangle represents a room in which we installed our stationary LiDAR.



(a) LiDAR Point Cloud

(b) Image Point Cloud

(c) LiDAR Point Cloud With Visibility

(d) Image Point Cloud With Visibility

Figure 11: **Point Clouds with Visibility Information:** Terrestrial point clouds (a, b) from the *Terrace* scene of ETH3D [45]. We visualise some of the sensor positions ● and lines-of-sight ——— (c, d).

Figure 12: Images of the three scenes from the *ETH3D* dataset we used. **Top**: Outdoor scene ("*Courtyard*") - **Middle**: indoor scene ("*Pipes*") - **Bottom**: Outdoor Scene ("*Terrace*").

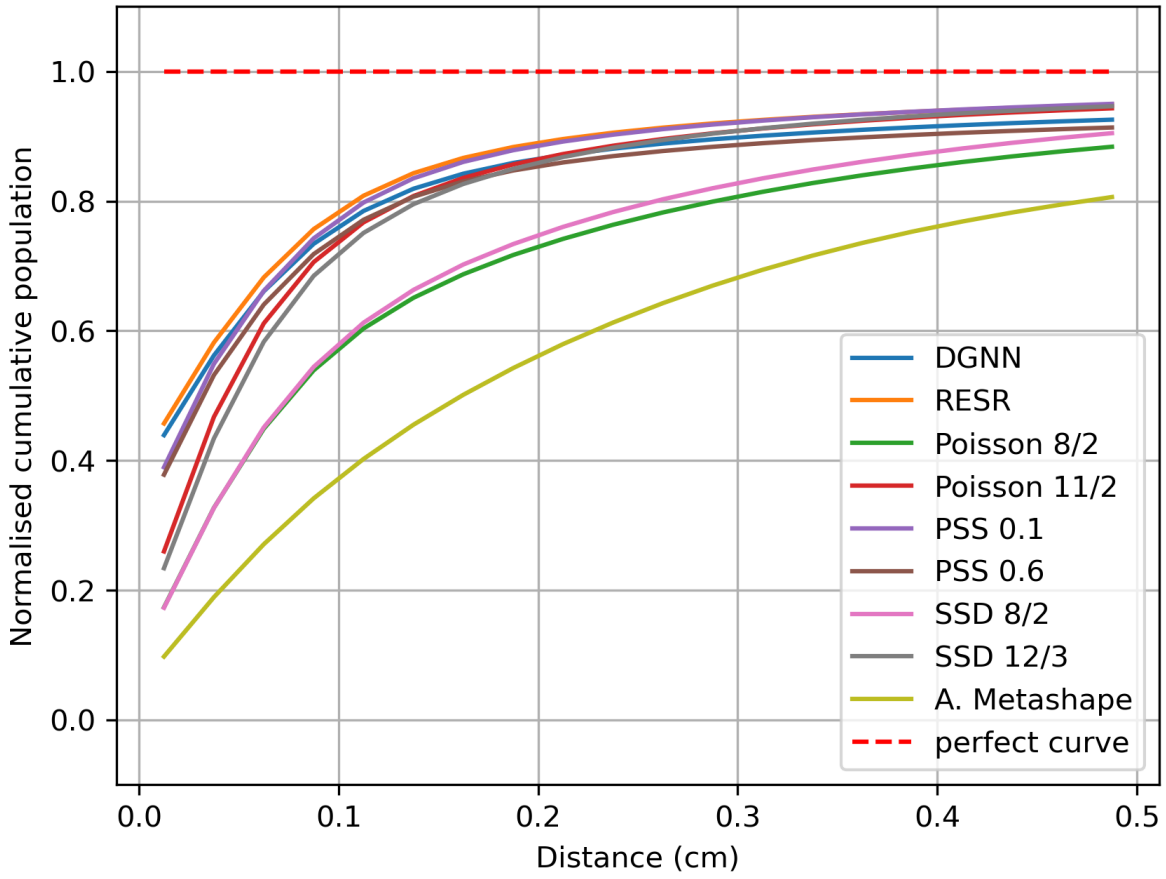
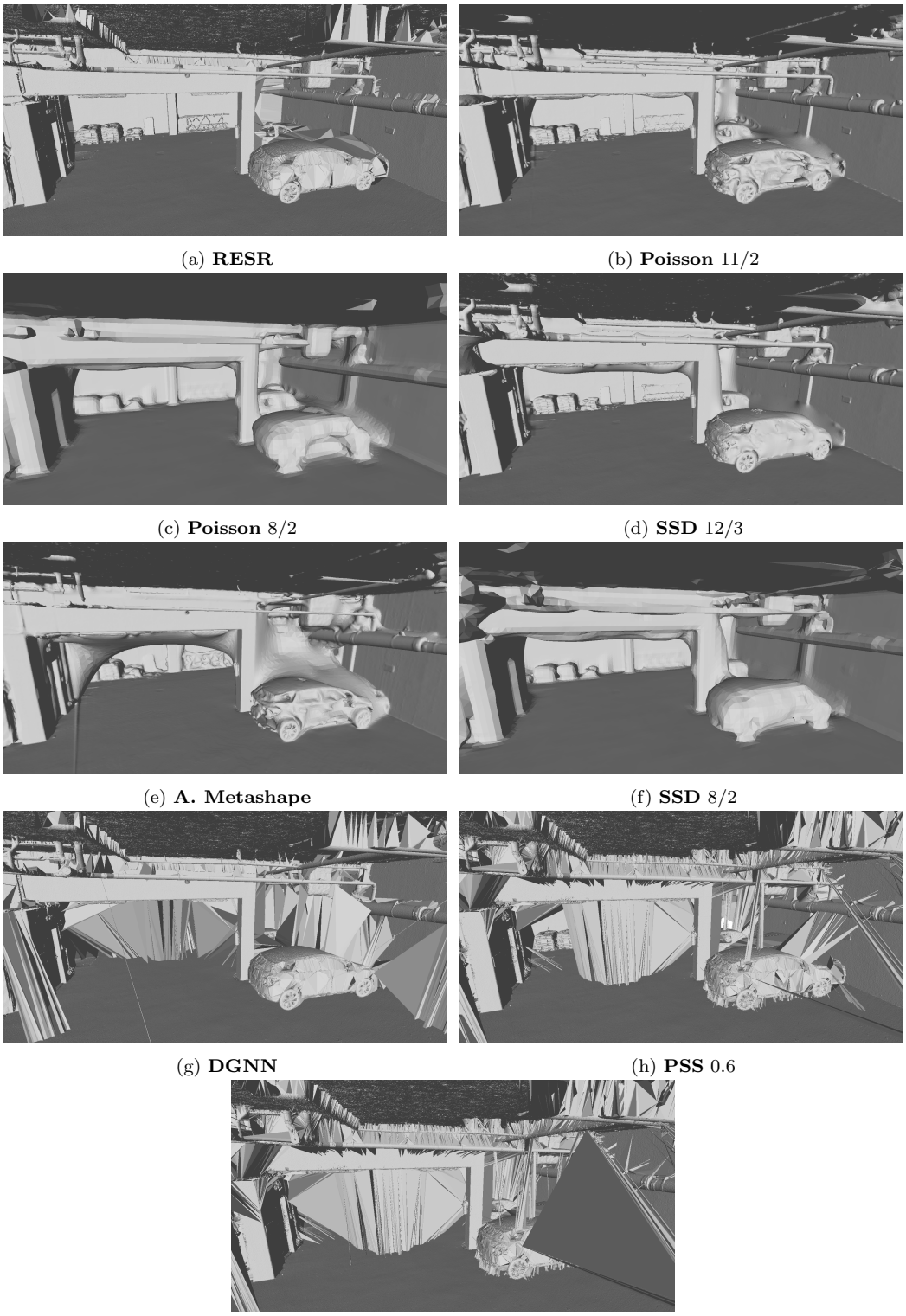


Figure 13: Cumulative distances over the three scenes from *STRAS* dataset for $d_{max} = 50\,cm$.



(a) **RESR** (b) **Poisson** 11/2

(c) **Poisson** 8/2 (d) **SSD** 12/3

(e) **A. Metashape** (f) **SSD** 8/2

(g) **DGNN** (h) **PSS** 0.6

(i) **PSS** 0.1

Figure 14: Reconstructed meshes from the **ENSG** Parking Lot scene.

References

[1] R. Lange, P. Seitz, Solid-state time-of-flight range camera, IEEE Journal of Quantum Electronics 37 (3) (2001) 390–397. doi:10.1109/3.910448.

[2] J. Geng, Structured-light 3D surface imaging: a tutorial, Advances in Optics and Photonics 3 (2) (2011) 128–160. doi:10.1364/AOP.3.000128.

[3] B. Lohani, S. Ghosh, Airborne LiDAR technology: a review of data collection and processing systems, Proceedings of the National Academy of Sciences, India Section A: Physical Sciences 87 (4) (2017) 567–579, publisher: Springer.

[4] Y. Furukawa, C. Hernández, Multi-view stereo: A tutorial, Foundations and Trends® in Computer Graphics and Vision 9 (1-2) (2015) 1–148, publisher: Now Publishers Inc. Hanover, MA, USA.

[5] O. Ozyesil, V. Voroninski, R. Basri, A. Singer, A survey of structure from motion, arXiv preprint arXiv:1701.08493.

[6] Y. Marchand, B. Vallet, L. Caraffa, Evaluating Surface Mesh Reconstruction of Open Scenes, ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 43 (2021) 369–376.

[7] W. Xiao, B. Vallet, M. Brédif, N. Paparoditis, Street environment change detection from mobile laser scanning point clouds, ISPRS Journal of Photogrammetry and Remote Sensing 107 (2015) 38–49, publisher: Elsevier.

[8] M. Van Kreveld, T. Van Lankveld, R. C. Veltkamp, Watertight scenes from urban lidar and planar surfaces, in: Computer Graphics Forum, Vol. 32, Wiley Online Library, 2013, pp. 217–228, issue: 5.

[9] M. Kazhdan, H. Hoppe, Screened poisson surface reconstruction, ACM Transactions on Graphics 32 (3) (2013) 1–13. doi:10.1145/2487228.2487237.

[10] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, C. T. Silva, A Survey of Surface Reconstruction from Point Clouds, Comput. Graph. Forum 36 (1) (2017) 301–329. doi:10.1111/cgf.12802.
URL https://doi.org/10.1111/cgf.12802

[11] A. Khatamian, H. Arabnia, Survey on 3D Surface Reconstruction, Journal of Information Processing Systems 12 (2016) 338–357. doi:10.3745/JIPS.01.0010.

[12] M. Kazhdan, Reconstruction of solid models from oriented point sets, in: Proceedings of the third Eurographics symposium on Geometry processing, Eurographics Association, 2005, p. 73.

[13] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proceedings of the fourth Eurographics symposium on Geometry processing, Eurographics Association, 2006, pp. 61–70.

[14] S. Peng, C. M. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, A. Geiger, Shape As Points: A Differentiable Poisson Solver, in: Advances in Neural Information Processing Systems (NeurIPS), 2021.

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

[16] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, A. Geiger, Occupancy networks: Learning 3d reconstruction in function space, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4460–4470.

[17] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, A. Geiger, Convolutional occupancy networks, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16, Springer, 2020, pp. 523–540.

[18] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, M. Aubry, A Papier-Mâché Approach to Learning 3D Surface Generation, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[19] P. Labatut, J.-P. Pons, R. Keriven, Robust and efficient surface reconstruction from range data, in: Computer graphics forum, Vol. 28, Wiley Online Library, 2009, pp. 2275–2290, issue: 8.

[20] F. Lafarge, P. Alliez, Surface reconstruction through point set structuring, in: Proc. of Eurographics, Girona, Spain, 2013.

[21] L. Caraffa, M. Brédif, B. Vallet, 3D Watertight Mesh Generation with Uncertainties from Ubiquitous Data, in: S.-H. Lai, V. Lepetit, K. Nishino, Y. Sato (Eds.), Computer Vision – ACCV 2016, Lecture Notes in Computer Science, Springer International Publishing, 2016, pp. 377–391.

[22] R. Kolluri, J. R. Shewchuk, J. F. O'Brien, Spectral Surface Reconstruction from Noisy Point Clouds, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '04, ACM, New York, NY, USA, 2004, pp. 11–21, event-place: Nice, France. doi:10.1145/1057432.1057434.
URL http://doi.acm.org/10.1145/1057432.1057434

[23] C. Holenstein, R. Zlot, M. Bosse, Watertight surface reconstruction of caves from 3D laser data, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2011, pp. 3830–3837.

[24] R. Sulzer, L. Landrieu, R. Marlet, B. Vallet, Scalable Surface Reconstruction with Delaunay-Graph Neural Networks, in: Computer Graphics Forum, Vol. 40, Wiley Online Library, 2021, pp. 157–167, issue: 5.

[25] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, H.-P. Seidel, Multi-level partition of unity implicits, ACM Transactions on Graphics 22 (3) (2003) 463–470. doi:10.1145/882262.882293.

[26] F. Calakli, G. Taubin, SSD: Smooth signed distance surface reconstruction, in: Computer Graphics Forum, Vol. 30, Wiley Online Library, 2011, pp. 1993–2002, issue: 7.

[27] J. J. Park, P. Florence, J. Straub, R. Newcombe, S. Lovegrove, Deepsdf: Learning continuous signed distance functions for shape representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 165–174.

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

[29] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for point-cloud shape detection, in: Computer graphics forum, Vol. 26, Wiley Online Library, 2007, pp. 214–226, issue: 2.

[30] P. Lancaster, Moving Weighted Least-Squares Methods, in: B. N. Sahney (Ed.), Polynomial and Spline Approximation: Theory and Applications, NATO Advanced Study Institutes Series, Springer Netherlands, Dordrecht, 1979, pp. 103–120. doi:10.1007/978-94-009-9443-0_7.
URL https://doi.org/10.1007/978-94-009-9443-0_7

[31] D. Shepard, A two-dimensional interpolation function for irregularly-spaced data, in: Proceedings of the 1968 23rd ACM national conference, ACM '68, Association for Computing Machinery, New York, NY, USA, 1968, pp. 517–524. doi:10.1145/800186.810616.
URL https://doi.org/10.1145/800186.810616

[32] Z.-Q. Cheng, Y.-Z. Wang, B. Li, K. Xu, G. Dang, S.-Y. Jin, A survey of methods for moving least squares surfaces, in: Proceedings of the Fifth Eurographics / IEEE VGTC conference on Point-Based Graphics, SPBG'08, Eurographics Association, Los Angeles, CA, 2008, pp. 9–23.

[33] D. Levin, The Approximation Power Of Moving Least-Squares, Mathematics of Computation 67. doi:10.1090/S0025-5718-98-00974-0.

[34] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. T. Silva, Point set surfaces | Proceedings of the conference on Visualization '01 (2001).
URL https://dl.acm.org/doi/abs/10.5555/601671.601673

[35] D. Levin, Mesh-Independent Surface Interpolation, Geometric Modeling for Scientific Visualization 3. doi:10.1007/978-3-662-07443-5_3.

[36] F. Ter Haar, P. Cignoni, P. Min, R. Veltkamp, A comparison of systems and tools for 3D scanning, 3D Digital Imaging and Modeling: Applications of Heritage, Industry, Medicine and Land.

[37] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, C. T. Silva, A benchmark for surface reconstruction, ACM Transactions on Graphics 32 (2) (2013) 1–17. doi:10.1145/2451236.2451246.

[38] J. Manson, G. Petrova, S. Schaefer, Streaming Surface Reconstruction Using Wavelets, in: Proceedings of the Symposium on Geometry Processing, SGP '08, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2008, pp. 1411–1420, event-place: Copenhagen, Denmark.
URL http://dl.acm.org/citation.cfm?id=1731309.1731324

[39] J. Süßmuth, Q. Meyer, G. Greiner, Surface reconstruction based on hierarchical floating radial basis functions, in: Computer Graphics Forum, Vol. 29, Wiley Online Library, 2010, pp. 1854–1864, issue: 6.

[40] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, W. Stuetzle, Surface reconstruction from unorganized points. SIGGRAPH Computer Graphics, 26(2), 71-78, Computer Graphics (Proc. SIGGRAPH 86) 20. doi:10.1145/142920.134011.

[41] L. Winiwarter, A. M. E. Pena, H. Weiser, K. Anders, J. M. Sánchez, M. Searle, B. Höfle, Virtual laser scanning with HELIOS++: A novel take on ray tracing-based simulation of topographic full-waveform 3D laser scanning, Remote Sensing of Environment 269 (2022) 112772. doi:https://doi.org/10.1016/j.rse.2021.112772.
URL https://www.sciencedirect.com/science/article/pii/S0034425721004922

[42] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma, R. Urtasun, LiDARsim: Realistic LiDAR Simulation by Leveraging the Real World, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[43] P. Cignoni, C. Rocchini, R. Scopigno, Metro: measuring error on simplified surfaces, in: Computer graphics forum, Vol. 17, Wiley Online Library, 1998, pp. 167–174, issue: 2.

[44] K. Hildebrandt, K. Polthier, M. Wardetzky, On the convergence of metric and geometric properties of polyhedral surfaces, Geometriae Dedicata 123 (1) (2006) 89–112, publisher: Springer.

[45] T. Schops, J. L. Schonberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, A. Geiger, A Multi-View Stereo Benchmark With High-Resolution Images and Multi-Camera Videos, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[46] D. Cernea, OpenMVS: Multi-View Stereo Reconstruction Library (2020).

URL https://cdcseacave.github.io/openMVS

[47] M. Jancosek, T. Pajdla, Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces, International scholarly research notices 2014, publisher: Hindawi.

[48] H.-H. Vu, P. Labatut, J.-P. Pons, R. Keriven, High accuracy and visibility-consistent dense multiview stereo, IEEE transactions on pattern analysis and machine intelligence 34 (5) (2011) 889–901, publisher: IEEE.