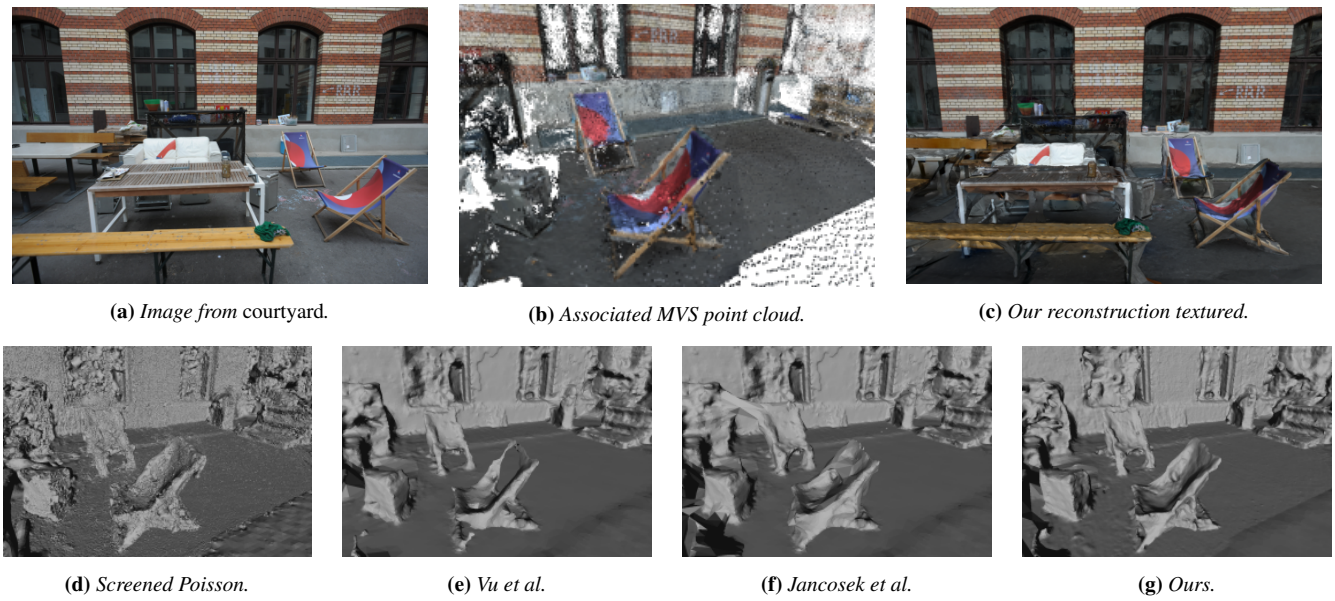
DOI: 10.1111/cgf.14364
Eurographics Symposium on Geometry Processing 2021
K. Crane and J. Digne
(Guest Editors)

Volume 40 (2021), Number 5

Scalable Surface Reconstruction with Delaunay-Graph Neural Networks

R. Sulzer[1,2] L. Landrieu[1] R. Marlet[2,3] B. Vallet[1]

[1] LASTIG, Univ Gustave Eiffel, ENSG IGN, F-94160 Saint-Mande, France
[2] LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France
[3] valeo.ai, Paris, France



(a) *Image from* courtyard. **(b)** *Associated MVS point cloud.* **(c)** *Our reconstruction textured.*

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

Figure 1: *Reconstruction of the* courtyard *scene of the ETH3D benchmark [SSG*17]. 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 [WMG14]. Bottom: we show untextured mesh reconstructions obtained by the screened Poisson algorithm in (d), the algorithms of Vu et al. [VLPK12] in (e) and Jancosek et al. [JP11] 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).*

Abstract

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 tetrahedralization whose cells are classified as inside or outside the surface by a graph neural network and an energy model solvable with a graph cut. 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. Combining the efficiency of deep learning methods and the scalability of energy-based models, our approach outperforms both learning and non learning-based reconstruction algorithms on two publicly available reconstruction benchmarks.

CCS Concepts

• ***Computing methodologies*** → ***Reconstruction;*** *Neural networks; Shape inference;*

1. Introduction

Reconstructing a surface from an unstructured point cloud is a long-standing and particularly challenging problem when applied to real-life acquisitions, due to occlusions, noise, outliers, non-uniform sampling, and misaligned scans [BTS*16].

A successful approach for dealing with large point clouds is to (i) tessellate the convex hull of the point cloud using a 3D Delaunay tetrahedralization (3DT), (ii) label the resulting cells as *inside* or *outside*, and (iii) extract the surface as the interface between cells with different labels [JP11, LPK09, VLPK12]. This guarantees to produce non-self-intersecting and watertight surfaces, a useful requirement for downstream engineering applications.

The methods used for classifying 3D tetrahedra typically rely on an energy formulation with handcrafted unary and binary potentials. However, 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. For this reason, we opt for a learning-based approach to learn data-adaptive potentials while remaining scalable.

In this paper, 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 Fig. 2). We then train a graph neural network (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 [PNM*20].

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 [SSG*17] (see Fig. 1).

2. Related Work

2.1. Graph-Cut-Based Surface Reconstruction

Visibility-based surface reconstruction from LiDAR scans and/or MVS data is traditionally formulated as a graph-cut optimization problem [BRV16, CBV17, JP11, JP14, VLPK12, ZSH19]. The 3D space is discretized into the cells of a 3DT of captured points [LPK09] or the cells of an arrangement of detected planes [CLP10]. A graph $(\mathcal{T},\mathcal{E})$ is formed for which the vertices 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:

$$E(l) = \sum_{t\in\mathcal{T}} U_t(l_t) + \lambda \sum_{(s,t)\in\mathcal{E}} B_{s,t}(l_s,l_t)\ , \tag{1}$$

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 [BdLGM14].

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 [VLPK12, JP11, JP14, ZSH19], or by using other modalities [BRV16].

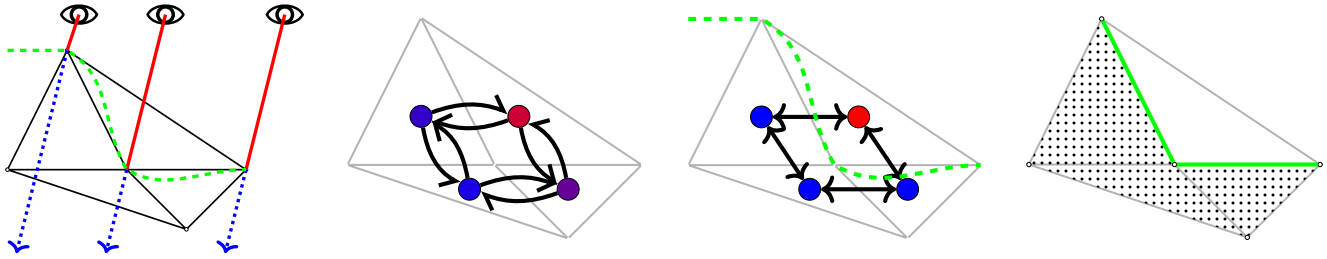
Binary potentials are used to force neighbouring cells crossed by the same line of sight to have the same labeling. Additionally, they can incorporate low-area [BRV16, CBV17] or other shape-based priors [JP11, LPK09, VLPK12]. Instead of hand-tuning the visibility model, we propose to learn it by training a neural network to produce unary potentials from local visibility and local geometric information.

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.

Surface-based approaches rely on transforming a discretized 2D surface, such as 2D patches or spheres [GFK*18, YFST18, SO20, LZS20], meshes [GJM19, HMGCO20, DN19], charts [WSS*18], or learned primitives [DGF*19], 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.

Volumetric approaches learn a continuous mapping from the input space $\mathbb{R}^3$ either to $\mathbb{R}$, defining the signed distance to the surface [GYH*20, PFS*19, AL20, AL21, CLI*20], or directly to an occupancy value $\{0,1\}$ [MON*19, MLT20, PNM*20]. The network training can be either unsupervised [LSCL19], aided by geometric regularization [GYH*20], or supervised by ground-truth surface information [MON*19, MLT20, PNM*20]. Some continuous methods [PFS*19, MON*19, PNM*20] predict the occupancy or signed distance conditionally to a latent shape representation, and



(a) *Delaunay triangulation and ray casting.*

(b) *Local and contextual learning with gnn.*

(c) *Global optimization with graph cuts.*

(d) *Surface reconstruction.*

Figure 2: ***Pipeline: 2D representation of the different steps of our method.*** *(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.*

thus learn a dataset-specific shape distribution. This can lead to difficulty in generalizing to shapes from unseen classes.

Even though volumetric approaches define a surface in continuous space with implicit functions, they often rely on a discretization of 3D space to learn these functions [MON*19, MLT20, PNM*20]. Recent works propose to scale these methods to larger scenes using an octree structure [MLT20] or a sliding window strategy [PNM*20].

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.

3. Methodology

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.

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, and $\mathcal{C} \in \mathbb{R}^{3\times C}$ the absolute positions of a set of cameras used to capture these points. 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* $c\text{—}p \in \mathcal{L}$ is an oriented segment from c to p. 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 $c\text{—}p$ 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 3, ${}^{c}p\rightarrow$ does not go beyond t_3.

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 $c\text{—}p$ 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:

$$\mathcal{L}_t^{\mathrm{v}} = \{(c,p) \in \mathcal{L} \mid (c\text{—}p) \cap t \neq \varnothing,\ p \in \mathcal{V}_t\} \tag{2}$$

$$\mathcal{L}_t^{\mathrm{f}} = \{(c,p) \in \mathcal{L} \mid (c\text{—}p) \cap t \neq \varnothing,\ p \notin \mathcal{V}_t\}. \tag{3}$$

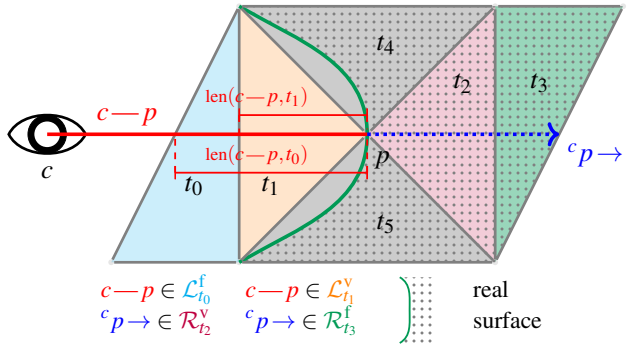


Figure 3: ***Visibility Model.*** *2D representation of our visibility features. A line-of-sight $c—p$ between a camera c and a visible 3D point p also defines a ray ${}^c p\rightarrow$. The line-of-sight $c—p$ traverses the two outside tetrahedra t_0 and t_1, while the ray ${}^c p\rightarrow$ traverses the two inside tetrahedra t_2 and t_3. Neither $c—p$, nor ${}^c p\rightarrow$ traverse t_4 or t_5; they thus do not contribute to their visibility information.*

Likewise, we denote $\mathcal{R}^{\text{v}}_t$ and $\mathcal{R}^{\text{f}}_t$ the equivalent sets for rays in $\mathcal{R}$. These definitions are illustrated on Figure 3. These sets are informative for determining the occupancy of a tetrahedron. Indeed, a tetrahedron t for which $\mathcal{R}^{\text{v}}_t$ is nonempty indicates that it is directly *behind* an element of the surface, hinting at a higher probability of insideness. A nonempty $\mathcal{R}^{\text{f}}_t$ 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 $\mathcal{L}^{\text{f}}_t$ indicates that it is traversed by a line-of-sight, indicating a high probability of outsideness. A nonempty $\mathcal{L}^{\text{v}}_t$ 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:

$$\text{count}(t) = \left[|\mathcal{L}^{\text{v}}_t|, |\mathcal{L}^{\text{f}}_t|, |\mathcal{R}^{\text{v}}_t|, |\mathcal{R}^{\text{f}}_t| \right] . \tag{4}$$

Then, to measure the *proximity* between t and the impact point p of a traversing line-of-sight $c—p$, we define $\text{len}(c—p,t)$ as the distance between p and the exit point of $c—p$ in t seen as from p. As represented in Figure 3, this corresponds to the length of the longest segment between p and the portion of $c—p$ intersecting t:

$$\text{len}(c—p,t) = \max_{y \in (c—p) \cap t} \|p - y\| . \tag{5}$$

When $(c—p) \cap t$ is empty, $\text{len}(c—p,t)$ 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:

$$\text{dist}(t) = \left[\min_{c—p \in \mathcal{L}^{\text{v}}_t} \text{len}(c—p,t), \min_{c—p \in \mathcal{L}^{\text{f}}_t} \text{len}(c—p,t), \right.$$
$$\left. \min_{{}^c p\rightarrow \in \mathcal{R}^{\text{v}}_t} \text{len}({}^c p\rightarrow,t), \min_{{}^c p\rightarrow \in \mathcal{R}^{\text{v}}_t} \text{len}({}^c p\rightarrow,t) \right] . \tag{6}$$

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.

3.3. Contextual Learning

We learn contextual information with a GNN using the propagation scheme GraphSAGE of Hamilton *et al.* [HYL17] with a depth of K (see Figure 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 3.2: $x^0_s = f_s$. We then apply the following update rule in two nested loops over $k = 0, \dots, K-1$ and for all $s \in \text{hop}(t, K-1)$:

$$x^{k+1}_s = \sigma\left(\text{norm}\left(W^{(k)} \left[x^k_s \,\middle\|\, \underset{u \in \mathcal{N}(s)}{\text{mean}} \left(x^k_u \right) \right] \right) \right) , \tag{7}$$

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^K_t to a vector of dimension 2 representing the inside/outside scores for tetrahedron t:

$$(i_t, o_t) = \text{MLP}(x^K_t) . \tag{8}$$

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.

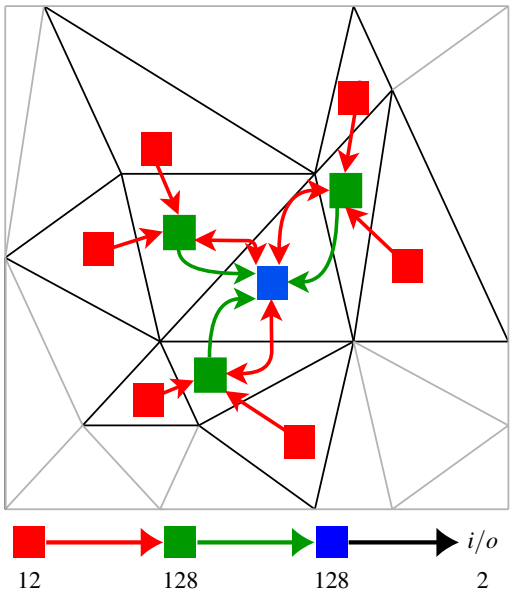


Figure 4: ***Graph Convolution Scheme.*** *Illustration of our GNN update for tetrahedron ■. Information is pooled at different hops (here $K = 2$) into an increasingly rich descriptor. A linear layer based on the last cell embedding assigns an inside/outside score to the central tetrahedron. Note that each prediction can be performed independently for each tetrahedron by considering only the K-hop subgraph.*

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 $i^g \in [0,1]^{|\mathcal{T}|}$ 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:

$$\hat{\imath}_t = \frac{\exp(i_t)}{\exp(i_t) + \exp(o_t)} \tag{9}$$

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 $\hat{\imath}_t$:

$$\mathrm{KL}_t(\hat{\imath}_t) = i_t^g \log(\hat{\imath}_t) + (1 - i_t^g) \log(1 - \hat{\imath}_t) + q\,, \tag{10}$$

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:

$$L(\hat{\imath}) = \frac{1}{\sum_{t\in\mathcal{T}} V_t} \sum_{t\in\mathcal{T}} V_t\, \mathrm{KL}_t(\hat{\imath}_t)\,. \tag{11}$$

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 1:

$$U(l_t) = i_t\,[l_t = 0] + o_t\,[l_t = 1]\,, \tag{12}$$

with $[x = y]$ the Iverson bracket, equal to 1 if $x = y$ and 0 otherwise.

Following the idea of Labatut *et al.* [LPK09], we define the binary potentials introduced in Equation 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. See the supplementary for more details. We also add a constant factor α_{vis} to o_t for tetrahedra containing a camera, indicating that they must lie outside the surface. The energy $E(l)$ in Eq. (1) with unary and binary potentials as defined above can be minimized efficiently by constructing a flow graph and using a min-cut solver [BK04].

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 [Cer20], 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. 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 capacity for generalization.

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 [CXG*16]. 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.* [HSG18]. We then synthetically scan the models with different degrees of outliers and noise as described in the supplementary material, and build corresponding 3DTs. To obtain the ground truth occupancy, we randomly sample 100 points in each tetrahedron, and determine the

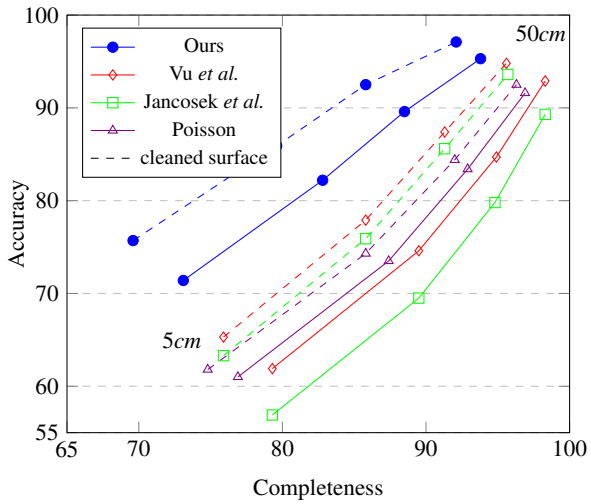


Figure 5: ***Performance analysis on ETH3D.*** *Each point corresponds to the accuracy and completeness at a given error threshold, respectively at 5, 10, 20 and 50 cm. Dashed lines represent the performance of meshes cleaned by post-processing. Our method produces meshes with a higher accuracy but a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

percentage of these sampled points lying inside the ground-truth model.

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 [IS15] 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 [KB15] 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 [BLN*13]. 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** [PNM*20] 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.
- **IGR** [GYH*20] is a deep model which we retrain for each object using the official implementation.
- Screened **Poisson** [KH13] 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.
- **Labatut *et al.*** [LPK09] 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 $\lambda = 5$).
- **Vu *et al.*** [VLPK12] is an extension of Labatut *et al.* [LPK09] to MVS data. We use its OpenMVS [Cer20] implementation with default parameters.
- **Jancosek *et al.*** [JP14] also exploits visibility in a graph-cut formulation, with special attention to weakly-supported surfaces. We use the OpenMVS [Cer20] implementation with default parameters.

For scene reconstruction, we compare all methods with and without mesh cleaning, with the same parameters over all experiments. See the supplementary for details. 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 1 and Table 2.

4.2. 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.* [BLN*13]. 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 MVS scans.

Results. The results are presented in Table 1 and illustrated in Figure 6 (*dancing children* shape) and in the supplementary material (the other shapes). We observe that the other learning-based methods have a hard time with this dataset. ConvONet [PNM*20] does not generalize well from the simple models of ShapeNet to the more challenging objects evaluated here. As for IGR [GYH*20], 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.* [LPK09], 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.

Table 1: ***Quantitative results on Berger et al.'s dataset.*** *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 [BLN*13]), 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 [GYH*20] "trained" on each of the 5 variants (LR, HR, HRN, HRO, HRNO) of the 5 shapes, screened Poisson reconstruction [KH13] with an octree of depth 10, Labatut et al. [LPK09] with σ set according to an estimation of the scan noise and ConvONet [PNM*20] and our method trained on the ShapeNet subset from [CXG*16].*

Method		Chamfer distance (per-point ave. %) [↓] LR	HR	HRN	HRO	HRNO	Mean	Volumetric IoU (%) [↑] LR	HR	HRN	HRO	HRNO	Mean
ConvONet	[PNM*20]	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	[GYH*20]	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	[KH13]	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.	[LPK09]	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** [↓] LR	HR	HRN	HRO	HRNO	Mean	**Number of non-manifold edges** [↓] LR	HR	HRN	HRO	HRNO	Mean
ConvONet	[PNM*20]	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	[GYH*20]	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	[KH13]	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.	[LPK09]	**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

4.3. 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 [SSG*17]. 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 [BSFG09] implemented in OpenMVS [Cer20]. The point clouds and associated camera poses are used as inputs for all mesh reconstruction methods evaluated in Section 4.1. Additionally, as input for the Screened Poisson and IGR algorithm, we estimate surface normals using Jets [CP05] 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 [PNM*20] 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 the supplementary 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 3. 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.

Table 2: ***Quantitative results on ETH3D.*** *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.*

Method	F1	CC	Area
Poisson [KH13]	66.8 (67.2)	83 (23131)	82 (116)
Vu *et al.* [VLPK12]	70.6 (70.8)	17 (560)	17 (125)
Jan. *et al.* [JP14]	70.0 (67.7)	14 (667)	14 (78)
Ours	**73.1 (72.1)**	23 (253)	11 (24)

Evaluation. We use the ETH3D [SSG*17] multi-view evaluation procedure. This protocol accounts for incomplete ground truth by segmenting the evaluation space into *occupied*, *free* and *unobserved* regions. As the procedure was designed to evaluate point reconstruction, we uniformly sample random points from the reconstructed meshes. We then evaluate the methods on extrinsic quality parameters, namely (i) accuracy, (ii) completeness, and (iii) F1-score (harmonic mean). Accuracy is defined as the fraction of sam-

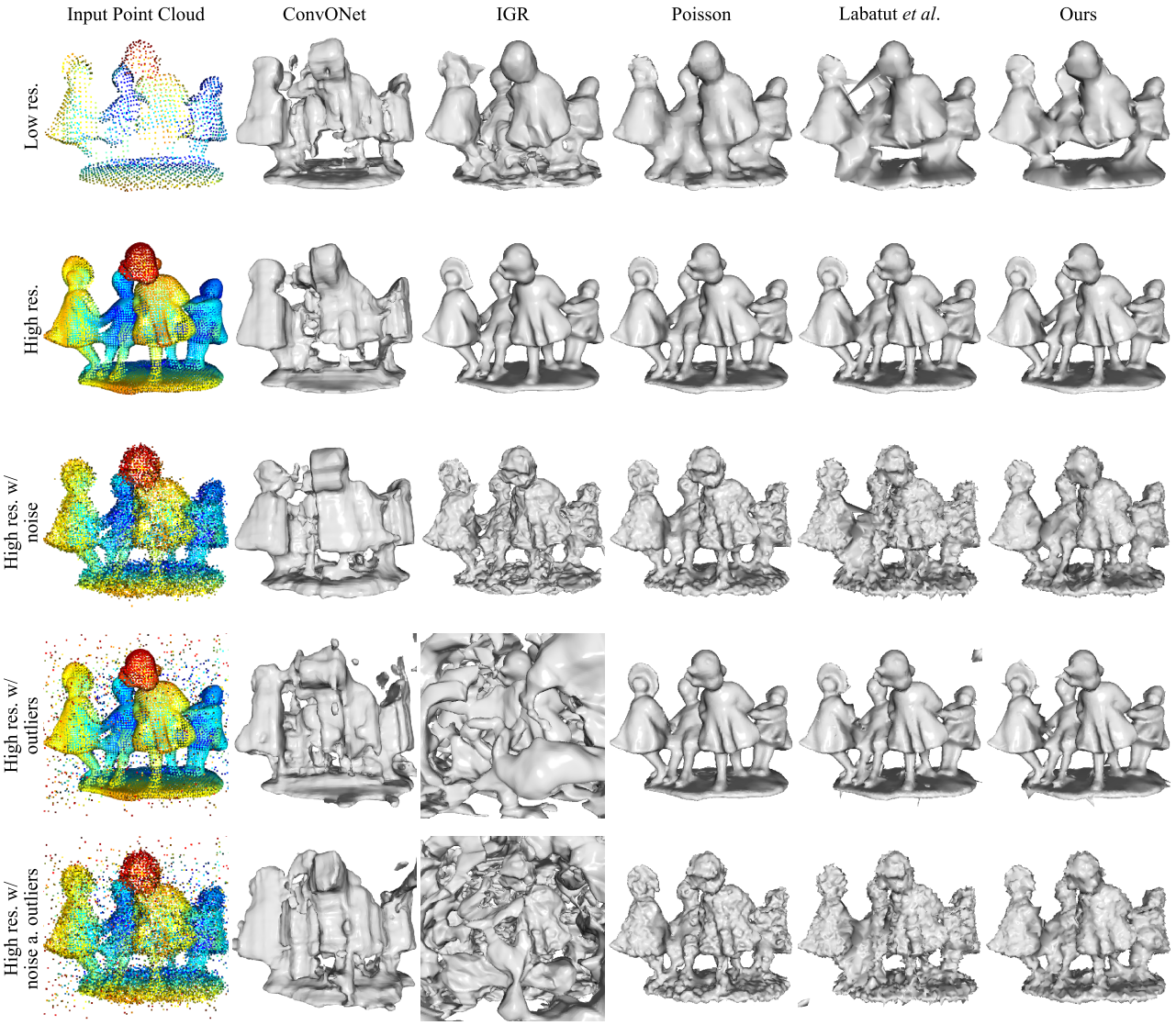


Figure 6: ***Qualitative Results on Berger et al.'s dataset.*** *Reconstruction of the* dancing children *shape of Berger et al.'s benchmark [BLN*13] with different levels of noise and outliers to emulate challenging MVS settings. Note that in contrast to ConvONet [PNM*20], our method generalizes much better to unseen objects, is highly resilient to outliers, and does not produce the floating artifacts of the IGR [GYH*20] and Labatut [LPK09] algorithms. The Screened Poisson reconstruction [KH13] is visually similar to ours, but occasionally produces unwanted surface parts.*

pled points on the output mesh within distance τ to any point in the ground truth. Reconstructed points in the unobserved space are thereby ignored. 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.

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.

Results. In Figure 5, 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 1 and Figure 7, where fine details that are hard to reconstruct are

(a) *Dense MVS input.* **(b)** *Our textured mesh.* **(c)** *Details Image.* **(d)** *[JP14].* **(e)** *Ours.*

Figure 7: ***Qualitative Results on ETH3D.*** *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 [WMG14]). We represent: in (c), a cropped image of a detail from the* terrace *scene of the ETH3D benchmark [SSG*17]; in (d), the reconstruction by Jancosek and et al. [JP14]; 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.*

better preserved. In Table 2, 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. More results can be found in the appendix.

We would like to stress that, while our model is learning-based, its training set [CFG*15] is very different from the one used to evaluate the performance [SSG*17]: 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.

Speed and Memory. In Table 3, we report the speed and GPU memory requirements of the different competing methods. Our approach compares favorably to ConvONet for all space-time tradeoffs, on top of improved reconstruction metrics. While the added GNN inference step of our method results in a slower overall prediction compared to [VLPK12], 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.

4.4. 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.

Table 3: ***Time 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. [VLPK12]	-	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 [PNM*20]	1	5 s	-	145 s	7.9 GB	14 s	164 s

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 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 [BVZ01]. 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.

Graph Convolution. We tried replacing our GNN scheme with the Dynamic Edge Conditioned Convolution of Simonovsky *et al.* [SK17] 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 information should be combined with geometric information, which is not typically done in traditional approaches.

Limitations. As 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. 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.

5. 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 for deep-learning techniques to successfully tackle hard problems of computational geometry at a large scale.

Acknowledgments

This work was partially funded by the ANR-17-CE23-0003 BIOM grant.

References

[AL20] ATZMON M., LIPMAN Y.: SAL: Sign agnostic learning of shapes from raw data. In *CVPR* (2020). 2

[AL21] ATZMON M., LIPMAN Y.: SALD: sign agnostic learning with derivatives. In *ICLR* (2021). 2

[BdLGM14] BOULCH A., DE LA GORCE M., MARLET R.: Piecewise-planar 3D reconstruction with edge and corner regularization. *Computer Graphic Forum* (2014). 2

[BK04] BOYKOV Y., KOLMOGOROV V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI* (2004). 5

[BLN*13] BERGER M., LEVINE J. A., NONATO L. G., TAUBIN G., SILVA C. T.: A benchmark for surface reconstruction. *ACM Transaction on Graphics.* (2013). 6, 7, 8

[BRV16] BÓDIS-SZOMORÚ A., RIEMENSCHNEIDER H., VAN GOOL L.: Efficient volumetric fusion of airborne and street-side data for urban reconstruction. In *ICPR* (2016). 2

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics* (2009). 7

[BTS*16] BERGER M., TAGLIASACCHI A., SEVERSKY L., ALLIEZ P., GUENNEBAUD G., LEVINE J., SHARF A., SILVA C.: A survey of surface reconstruction from point clouds. *Computer Graphics Forum* (2016). 2

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *PAMI* (2001). 9

[CBV17] CARAFFA L., BRÉDIF M., VALLET B.: 3D watertight mesh generation with uncertainties from ubiquitous data. In *ACCV* (2017). 2

[Cer20] CERNEA D.: OpenMVS: Multi-view stereo reconstruction library, 2020. URL: https://cdcseacave.github.io/openMVS. 5, 6, 7

[CFG*15] 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.: *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep., Stanford University, Princeton University, Toyota Technological Institute at Chicago, 2015. 9

[CLI*20] CHABRA R., LENSSEN J. E., ILG E., SCHMIDT T., STRAUB J., LOVEGROVE S., NEWCOMBE R.: Deep local shapes: Learning local SDF priors for detailed 3D reconstruction. In *ECCV* (2020). 2

[CLP10] CHAUVE A.-L., LABATUT P., PONS J.-P.: Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *CVPR* (2010). 2

[CP05] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design 22*, 2 (2005), 121–146. 7

[CXG*16] CHOY C. B., XU D., GWAK J., CHEN K., SAVARESE S.: 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *ECCV* (2016). 5, 7

[DGF*19] DEPRELLE T., GROUEIX T., FISHER M., KIM V., RUSSELL B., AUBRY M.: Learning elementary structures for 3D shape generation and matching. In *NeurIPS* (2019). 2

[DN19] DAI A., NIESSNER M.: Scan2Mesh: From unstructured range scans to 3D meshes. In *CVPR* (2019). 2

[GFK*18] GROUEIX T., FISHER M., KIM V. G., RUSSELL B. C., AUBRY M.: AtlasNet: A papier-mâché approach to learning 3D surface generation. In *CVPR* (2018). 2

[GJM19] GKIOXARI G., JOHNSON J., MALIK J.: Mesh R-CNN. In *CVPR* (2019). 2

[GYH*20] GROPP A., YARIV L., HAIM N., ATZMON M., LIPMAN Y.: Implicit geometric regularization for learning shapes. *Machine Learning and Systems* (2020). 2, 6, 7, 8

[HMGCO20] HANOCKA R., METZER G., GIRYES R., COHEN-OR D.: Point2Mesh: A self-prior for deformable meshes. *ACM Transaction on Graphics* (2020). 2

[HSG18] HUANG J., SU H., GUIBAS L.: Robust watertight manifold surface generation method for ShapeNet models. *arXiv preprint arXiv:1802.01698* (2018). 5

[HYL17] HAMILTON W. L., YING R., LESKOVEC J.: Inductive representation learning on large graphs. In *NeurIPS* (2017). 4

[IS15] IOFFE S., SZEGEDY C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML* (2015). 6

[JP11] JANCOSEK M., PAJDLA T.: Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR* (2011). 1, 2

[JP14] JANCOSEK M., PAJDLA T.: Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces. *International Scholarly Research Notices* (2014). 2, 6, 7, 9

[KB15] KINGMA D. P., BA J.: ADAM: A method for stochastic optimization. In *ICLR* (2015). 6

[KH13] KAZHDAN M., HOPPE H.: Screened Poisson surface reconstruction. *ACM Transaction on Graphics.* (2013). 6, 7, 8

[LPK09] LABATUT P., PONS J. P., KERIVEN R.: Robust and efficient surface reconstruction from range data. *Computer Graphics Forum* (2009). 2, 5, 6, 7, 8

[LSCL19] LIU S., SAITO S., CHEN W., LI H.: Learning to infer implicit surfaces without 3D supervision. In *NeurIPS* (2019). 2

[LZS20] LIU M., ZHANG X., SU H.: Meshing point clouds with predicted intrinsic-extrinsic ratio guidance. In *ECCV* (2020). 2

[MLT20] MI Z., LUO Y., TAO W.: SSRNet: Scalable 3D surface reconstruction network. In *CVPR* (2020). 2, 3

[MON*19] MESCHEDER L., OECHSLE M., NIEMEYER M., NOWOZIN S., GEIGER A.: Occupancy networks: Learning 3D reconstruction in function space. In *CVPR* (2019). 2, 3

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR* (2019). 2

[PNM*20] PENG S., NIEMEYER M., MESCHEDER L., POLLEFEYS M., GEIGER A.: Convolutional occupancy networks. In *ECCV* (2020). 2, 3, 6, 7, 8, 9

[SK17] SIMONOVSKY M., KOMODAKIS N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR* (2017). 9

[SO20] SHARP N., OVSJANIKOV M.: PointTriNet: Learned triangulation of 3D point sets. In *ECCV* (2020). 2

[SSG*17] SCHÖPS T., SCHÖNBERGER J. L., GALLIANI S., SATTLER T., SCHINDLER K., POLLEFEYS M., GEIGER A.: A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR* (2017). 1, 2, 7, 9

[VLPK12] VU H. H., LABATUT P., PONS J. P., KERIVEN R.: High accuracy and visibility-consistent dense multiview stereo. *PAMI* (2012). 1, 2, 6, 7, 9

[WMG14] WAECHTER M., MOEHRLE N., GOESELE M.: Let there be color! large-scale texturing of 3D reconstructions. In *ECCV* (2014). 1, 9

[WSS*18] WILLIAMS F., SCHNEIDER T., SILVA C., ZORIN D., BRUNA J., PANOZZO D.: Deep geometric prior for surface reconstruction. In *CVPR* (2018). 2

[YFST18] YANG Y., FENG C., SHEN Y., TIAN D.: FoldingNet: Point cloud auto-encoder via deep grid deformation. In *CVPR* (2018). 2

[ZSH19] ZHOU Y., SHEN S., HU Z.: Detail preserved surface reconstruction from point cloud. *Sensors* (2019). 2