ISPRS Journal of Photogrammetry and Remote Sensing 194 (2022) 58–73

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

Reconstructing compact building models from point clouds using deep implicit fields

Zhaiyu Chen [a,b,1], Hugo Ledoux [a], Seyran Khademi [a], Liangliang Nan [a,*]

[a] Delft University of Technology, 2628 BL, Delft, The Netherlands
[b] Technical University of Munich, 80333, Munich, Germany

ARTICLE INFO

Keywords:
3D reconstruction
Compact building model
Point cloud
Deep neural network
Implicit field

ABSTRACT

While three-dimensional (3D) building models play an increasingly pivotal role in many real-world applications, obtaining a compact representation of buildings remains an open problem. In this paper, we present a novel framework for reconstructing compact, watertight, polygonal building models from point clouds. Our framework comprises three components: (a) a cell complex is generated via adaptive space partitioning that provides a polyhedral embedding as the candidate set; (b) an implicit field is learned by a deep neural network that facilitates building occupancy estimation; (c) a Markov random field is formulated to extract the outer surface of a building via combinatorial optimization. We evaluate and compare our method with state-of-the-art methods in generic reconstruction, model-based reconstruction, geometry simplification, and primitive assembly. Experiments on both synthetic and real-world point clouds have demonstrated that, with our neural-guided strategy, high-quality building models can be obtained with significant advantages in fidelity, compactness, and computational efficiency. Our method also shows robustness to noise and insufficient measurements, and it can directly generalize from synthetic scans to real-world measurements. The source code of this work is freely available at https://github.com/chenzhaiyu/points2poly.

1. Introduction

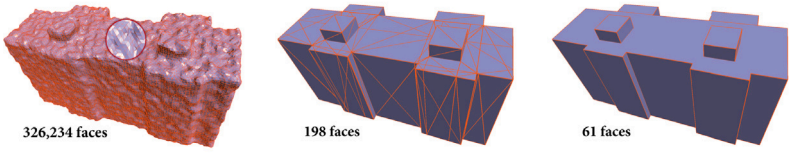
Three-dimensional (3D) building models have been playing an increasingly important role in various applications, such as urban planning (Herbert and Chen, 2015), solar potential analysis (Machete et al., 2018), noise pollution assessment (Stoter et al., 2020; Biljecki et al., 2015). Recently, with the development of augmented and virtual reality applications, the demand for high-quality 3D models of buildings has grown even faster (Blut and Blankenbach, 2021).

Most existing generic 3D reconstruction methods are dedicated to smooth surfaces represented as dense triangles, irrespective of piecewise planarity exhibited in the urban environment (Kazhdan et al., 2006; Erler et al., 2020). Compared to the dense triangle mesh representation, a compact surface model has a significantly low number of faces yet can still sufficiently describe the geometry of a building. Fig. 1 illustrates a building described as a smooth surface and piecewise-planar ones, where an arbitrary-sided polygonal surface exhibits the highest compactness. Although some works claim the possibility of reconstructing compact surface models from point clouds (Boulch et al., 2014; Mura et al., 2016; Li et al., 2016b; Nan and Wonka, 2017) or from dense triangle meshes (Bouzas et al., 2020; Li and Nan, 2021), they suffer from serious scalability issues. In this paper, we aim at a robust reconstruction of compact building surfaces directly from point clouds.

We exploit the fact that 3D shapes are not confined to explicit representations (e.g., point cloud, surface mesh, voxels), but can be encoded implicitly in a function space. For example, implicit fields are widely used to characterize 3D shapes, where the surface of a shape is implicitly interpreted as a zero-set of the signed distance field (SDF) (Kazhdan et al., 2006). A learnable indicator function of the SDF takes as input a query point and yields an indication of whether the point lies inside or outside the object. Then explicit geometry is often extracted from the field via computationally expensive iso-surfacing (Mescheder et al., 2019). Compared with explicit expressions that are heterogeneously distributed, the homogeneous functional representation is particularly favorable for geometric machine learning. Recently, Park et al. (2019) introduced a learning-based scheme with the use of the function space in 3D geometric modeling. An implicit field can be directly learned from the input point cloud and used to extract a smooth surface model of the object. However, extracting a compact polygonal model from the implicit field remains an open problem.

* Corresponding author.
E-mail addresses: zhaiyu.chen@tum.de (Z. Chen), h.ledoux@tudelft.nl (H. Ledoux), s.khademi@tudelft.nl (S. Khademi), liangliang.nan@tudelft.nl (L. Nan).
[1] Work done at Delft University of Technology.

https://doi.org/10.1016/j.isprsjprs.2022.09.017
Received 9 January 2022; Received in revised form 12 August 2022; Accepted 26 September 2022
Available online 17 October 2022



(a) Dense triangles (b) Sparse triangles (c) Polygonal surface

Fig. 1. A building in three different surface representations.

In this paper, we propose a novel framework for reconstructing compact, watertight, polygonal building meshes from point clouds by incorporating learnable implicit surface representation into explicit geometry construction. The explicit geometry provides a polyhedral embedding as the candidate set, from which the building surface can be obtained from an implicit field learned using a deep neural network (i.e., neural-guided). We formulate the surface extraction problem as a Markov random field (MRF), which can handle varying surface complexity. Through combinatorial optimization, the occupancy of a building inferred from a deep implicit field is further regularized by penalizing its surface complexity. Our reconstruction framework inherits the high efficiency and robustness in the inference of deep implicit fields, and the high fidelity of primitive assembly-based reconstruction approaches.

With our neural-guided framework, we demonstrate that high-quality building models can be obtained with significant advantages in terms of fidelity, compactness, and computational efficiency compared to state-of-the-art methods. The main contributions of this paper are as follows:

(i) A learning-based framework for compact building model reconstruction from point clouds. To the best of our knowledge, this is the first work where a deep implicit field is used for building reconstruction. Our method shows significant performance and quality advantage over state-of-the-art methods, especially for complex building models.
(ii) An MRF formulation for surface extraction from the occupancy learned by a neural network. Our formulation allows complexity control and favors compactness in the final reconstruction, and is far more efficient than the existing integer programming formulation.
(iii) The first point cloud dataset of synthetic buildings (with their corresponding surface models) for cultivating learning-based building reconstruction methods. The dataset contains 768 buildings with varying complexity and styles. Noise and scanning artifacts (e.g., occlusion) are simulated and added to the point clouds.

2. Related work

There is a large volume of literature on shape reconstruction from point clouds. In this section, we mainly review approaches aiming at obtaining compact polygonal models, namely model-based reconstruction, geometric simplification, and primitive assembly. We also discuss recent development in deep implicit fields, since it is one of the key components in our learning-based reconstruction framework.

2.1. Model-based reconstruction

Urban buildings typically demonstrate specific types of structures, such as roofs, walls, and dormers, which allows the use of strong shape priors and generic templates of such structures to ease the reconstruction. This type of approach is commonly referred to as model-based reconstruction.

The Manhattan-world assumption (Coughlan and Yuille, 2000) restricts the orientation of faces in only three orthogonal directions and represents the 3D scene with axis-aligned non-uniform polycubes (Ikehata et al., 2015; Li et al., 2016c,a). This simplification drastically reduces the geometric complexity and the solution space to explore. Another common assumption is restricting the output surface to specific disk topologies. The 2.5D view-dependent representation (Zhou and Neumann, 2010) can generate arbitrarily shaped roofs with vertical walls connecting them, from airborne light detection and ranging (LiDAR) measurements. Xiong et al. (2014, 2015) exploit roof topology graphs for reconstructing LoD2 buildings. Similarly, Li et al. (2016b) present a workflow to reconstruct building roofs for urban scenes. Kelly et al. (2017) formulate a global optimization to produce structured urban reconstruction from street-level imagery, GIS footprint, and coarse 3D mesh. The model-based approaches can maintain the uniformity of the reconstruction and is thus efficient to implement. However, they only apply to specific domains as a limited variety of the models constrains the generalization of these methods. Our reconstruction method, instead, imposes no geometric assumption except for piecewise planarity, thus remaining generic.

2.2. Geometric simplification

This type of method obtains compact surface models by the simplification of given dense triangle meshes. Dense triangle meshes can be obtained from point clouds with generic surface reconstruction techniques such as the Poisson reconstructions (Kazhdan et al., 2006; Kazhdan and Hoppe, 2013), which address the problem by establishing an indicator function underpinned by the points with their oriented normals. The output surface is acquired by extracting an iso-surface of this function. We refer to Berger et al. (2017) for a survey of generic surface reconstruction algorithms.

Dense triangles on a smooth surface can be simplified into concise polygons via various approaches. Garland and Heckbert (1997) propose to iteratively contract vertex pairs under quadric error metrics (QEM), such that the number of faces is reduced to an expected number. To preserve the piecewise-planar structure during contraction, Salinas et al. (2015) propose structure-aware mesh decimation (SAMD), which incorporates the planar proxies detected from preprocessing analysis into an adjacency graph and then approximates the mesh as well as the proxies. However, these contraction operators are inclined to more triangles than desired for piecewise-planar building representation. Alternatively, variational shape approximation (VSA) proposed by Cohen-Steiner et al. (2004) approaches the approximation through repeatedly partitioning based on an error tolerance. Furthermore, Verdie et al. (2015) propose an approach for reconstructing urban scenes with various LoD configurations from dense meshes through classification, abstraction, and reconstruction. Similarly, Zhu et al. (2018) present a simplification framework for urban scene modeling at different LODs, which is subsequently extended by Han et al. (2021) to more general scenes. Bouzas et al. (2020) incorporates structure awareness with the recovery and preservation of the initial mesh primitives and their adjacencies. Li and Nan (2021) further exploit the preservation of piecewise-planar structures and sharp features. These methods demand the input smooth surface be accurate in both geometry and topology for a faithful surface approximation. However, the requirement is rarely satisfied for real-world measurements. In contrast, our proposed reconstruction method can directly output a concise polygonal mesh without approximating an intermediate.

2.3. Primitive assembly

This type of method reconstructs compact building models by pursuing the optimal assembly of a set of basic geometric primitives (e.g., polygons, boxes).

Connectivity-based methods address the primitive assembly by extracting proper geometric primitives from an adjacency graph built on planar shapes (Chen and Chen, 2008; Schindler et al., 2011; Van Kreveld et al., 2011). Though the graph analysis can be efficiently executed, these methods are sensitive to the quality of the adjacency graph. Incorrect connectivity contaminated by linkage errors is prone to an incomplete reconstruction. Arikan et al. (2013) propose an interactive solution that enables the user to complete the surface through an optimization-based snapping, which requires laborious human interventions for complex scenes. Labatut et al. (2009), Lafarge and Alliez (2013) propose a mixed strategy where the confident areas are represented by polygonal shapes and the complex regions by dense triangles.

Slicing-based methods show stronger robustness to imperfect data with the divide-and-conquer strategy (Chauve et al., 2010; Mura et al., 2016; Nan and Wonka, 2017). They partition the 3D space into polyhedral cells using the supporting planes of the detected planar primitives, where the polyhedral cells consist of polygonal faces. The reconstruction is therefore transformed into a labeling problem where the polyhedral cells are labeled as either inside or outside the shape or equivalently with labeling other primitives. The main limitation of slicing-based methods is the scalability of their data structure. Specifically, the pairwise intersection of supporting planes results in an over-complex tessellation, which is computationally expensive to compute, commonly via a binary tree updated upon each primitive's insertion (Murali and Funkhouser, 1997). When many planar primitives contribute to the intersection, the resulting tessellation may hamper the surface extraction. Moreover, since many anisotropic cells are generated regardless of their spatial hierarchy, the resulting surface is inclined to geometric artifacts. For instance, PolyFit (Nan and Wonka, 2017) formulates polygonal surface reconstruction as a binary integer program with hard constraints ensuring the generated surface is watertight and manifold. However, it suffers from scalability issues due to unnecessary pairwise intersections, and it thus can only process simple models with limited complexity. In this work, we address the computation bottleneck by exploiting an adaptive space partitioning strategy, which significantly reduces the algorithmic complexity. Our adaptive strategy is similar to the kinetic data structure in KSR (Bauchet and Lafarge, 2020) that creates a partition by growing planar primitives at a constant speed until they collide and form polyhedra, avoiding exhaustive partitioning of the space. Instead of relying on a sophisticated kinetic data structure, our method only requires a simple data structure that stems from binary space partitioning (BSP). Moreover, our occupancy indicator does not rely on normal information that is required by KSR, which enables a broader spectrum of inputs.

Recently, Li and Wu (2021) extend PolyFit to further exploit the inter-relation of the primitives into procedural modeling for building reconstruction in the CityGML format. Similarly, Xie et al. (2021) propose to combine the rule-based and the hypothesis-based strategies for efficient building reconstruction. Fang and Lafarge (2020) introduce a hybrid approach for reconstructing 3D objects by successively connecting and slicing planes detected from 3D data. In contrast to these works that use hand-crafted features, our method exploits automatically learned deep features.

2.4. Deep implicit field

Recent advances in deep implicit fields have revealed their potential for 3D reconstruction. The crux of these methods is to learn a mapping from the input (e.g., a point cloud) to a continuous scalar field. Then, the surface of the object can be extracted via iso-surfacing techniques such as Marching Cubes (Lorensen and Cline, 1987). Iso-surfacing is powerful in extracting smooth surfaces but has the limitation of preserving sharp features. Inevitably, it introduces discretization errors. Deep implicit fields are thus not natively suitable for reconstructing compact polygonal models.

By incorporating constructed solid geometry (CSG), Chen et al. (2020) introduce an end-to-end neural network, BSP-Net, to reconstruct a shape from a set of convexes obtained via binary space partitioning. Similarly, Deng et al. (2020) propose an architecture to represent a low-dimensional family of convexes. These two methods both learn to divide and conquer the 3D space with implicit fields. However, the inputs to these two neural networks are either images or voxels, instead of point clouds that our work aims to address.

The single latent feature vector used by most deep implicit fields methods implies strong priors dependent on the training data. While this allows plausible surface reconstruction even with highly contaminated data, it significantly limits the generalization ability of these methods. With one feature vector encoding the whole shape, the feature space inevitably overfits the shapes in the training set, which may fail for shapes from unseen categories. Erler et al. (2020) propose the Points2Surf architecture to estimate an SDF with both local and global feature vectors, which has shown great generalization capabilities in implicit field learning. In this work, we utilize the Points2Surf architecture as an initialization step for 3D reconstruction.

2.5. Summary

Table 1 summarizes the existing works related to ours, including generic reconstruction methods (Kazhdan et al., 2006; Erler et al., 2020), model-based approaches (Zhou and Neumann, 2010; Li et al., 2016c), geometry simplification methods (Garland and Heckbert, 1997; Cohen-Steiner et al., 2004; Salinas et al., 2015), and primitive assembly (Nan and Wonka, 2017; Bauchet and Lafarge, 2020). Verdicts are based on whether each method can produce compact and watertight surfaces, whether it consumes unorganized points as input (i.e., without normal information and geometric approximation or reconstruction), whether it applies to generic 3D objects, and its scalability. Among these competitors, PolyFit and KSR are both capable of reconstructing compact, watertight, generic surfaces from point clouds and thus are considered the closest to ours. Our learning-based reconstruction framework combines the strengths of primitive assembly and deep implicit fields. It not only inherits the theoretical guarantee of primitive assembly to obtain compact building models but also integrates the data-driven features of deep implicit fields such that various types of buildings can be represented with strong robustness.

3. Methodology

3.1. Overview

Our neural-guided approach for building reconstruction from point clouds utilizes the learned implicit representation as an occupancy indicator for extracting the final surface model. The indicator can be intuitively interpreted as a shape-conditioned binary classifier for which the decision boundary represents the outer surface of a building. With the learned implicit field, a compact surface model is extracted using an MRF formulation that accounts for reconstruction fidelity and compactness. Fig. 2 demonstrates the workflow of our neural-guided reconstruction framework that consists of three steps:

- **Adaptive space partitioning**. We partition the ambient 3D space to generate a linear cell complex that complies with planar primitives detected from the point cloud. The partitioning is spatially adaptive therefore being efficient and respective to the building's geometry. The non-overlapping cells in the complex serve as the candidates whose outer shell constitutes the final surface.

Table 1
Overview of the most related works. GR, MR, GS, and PA stand for generic reconstruction, model-based reconstruction, geometry simplification, and primitive assembly, respectively. All the listed works are compared with our method in the experiments.

Related work	Characteristics					
	Category	Compact	Watertight	Raw	Generic	Scalable
Poisson (Kazhdan et al., 2006)	GR	✗	✗	✗	✓	✓
Points2Surf (Erler et al., 2020)	GR	✗	✗	✓	✓	✗
2.5D DC (Zhou and Neumann, 2010)	MR	✗	✓	✓	✗	✓
Manhattan-world (Li et al., 2016c)	MR	✓	✓	✗	✗	✓
QEM (Garland and Heckbert, 1997)	GS	✓	✗	✗	✓	✗
VSA (Cohen-Steiner et al., 2004)	GS	✓	✗	✗	✓	✗
SAMD (Salinas et al., 2015)	GS	✓	✗	✗	✓	✗
FPS (Li and Nan, 2021)	GS	✓	✗	✗	✓	✗
PolyFit (Nan and Wonka, 2017)	PA	✓	✓	✓	✓	✗
KSR (Bauchet and Lafarge, 2020)	PA	✓	✓	✗	✓	✓
Ours	PA	✓	✓	✓	✓	✓

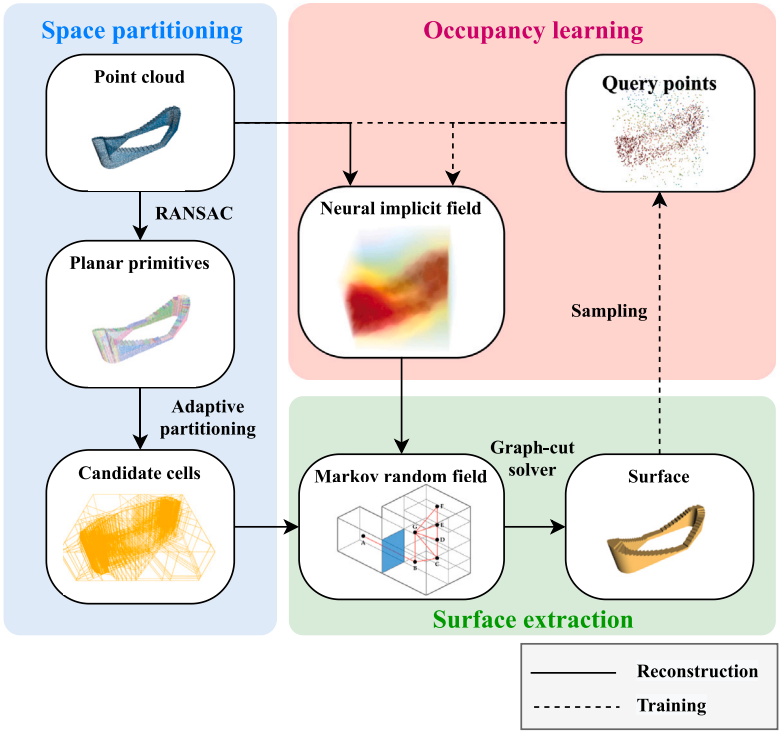


Fig. 2. Overview of our framework. The framework comprises three functional blocks: within the explicit block (in blue), a linear cell complex is generated from a point cloud by adaptive space partitioning; within the implicit block (in red), a neural implicit field is learned to indicate spatial occupancy of the building; within the optimization block (in green), an MRF is formulated to extract the final surface. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- **Occupancy learning**. We utilize a deep neural network to learn a shape-conditioned implicit field that characterizes the building object represented by the point cloud. The implicit field describes the spatial occupancy of the object given any query point in the 3D space.
- **Surface extraction**. This step takes the learned implicit field as an occupancy indicator and outputs the boundary representation of the building's surface. We formulate surface extraction as a binary classification problem and solve it using MRF optimization that encourages compactness and guarantees the final model to be watertight.

3.2. Adaptive space partitioning

We introduce an adaptive partitioning algorithm that adaptively subdivides the 3D space into a set of cells (i.e., a linear cell complex), where each cell is a convex polyhedron. Our algorithm partitions the space by extracting planes from the point cloud followed by BSP using the refined planar primitives. Specifically, we apply the RANSAC algorithm of Schnabel et al. (2007) to detect planes. Considering noise and outliers in the data, we perform a refinement procedure that iteratively merges planes under specific proximity conditions (see Algorithm 3.1). During the partitioning, a binary tree structure is dynamically and locally updated upon insertion of a primitive, and the cell adjacency information is maintained. A typical space partitioning step is illustrated in Fig. 3.

To avoid redundant partitioning, our adaptive strategy only allows intersecting spatially correlated primitives, as illustrated in Fig. 4(b). We describe the spatial correlation by intersection test between the axis-aligned bounding box (AABB) of a primitive and the cells in the leaf nodes of the BSP tree. Compared to exhaustive partitioning strategies (e.g., the pairwise intersection of planar primitives in Nan and Wonka (2017)), our adaptive partitioning can avoid unnecessary intersections between primitives and generate fewer yet more meaningful cells, which would otherwise result in computational overhead or potential artifacts in the final model. Fig. 4 shows an illustrative

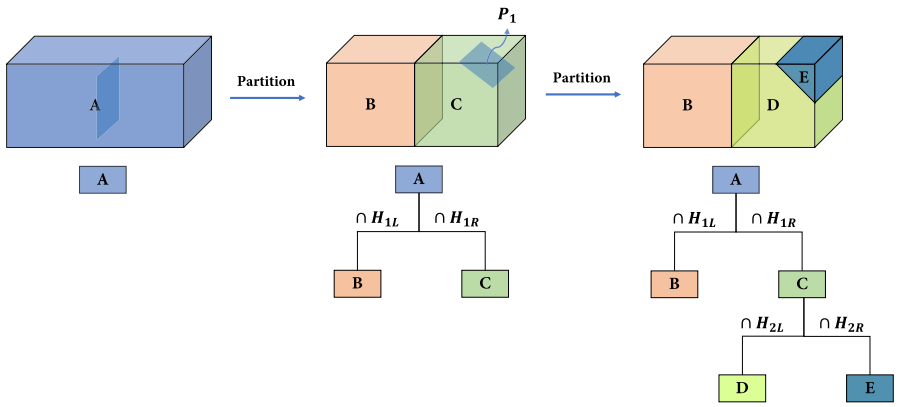


Fig. 3. An illustration of one step in space partition. After inserting a planar primitive P_1, cell C is split into two cells D and E.

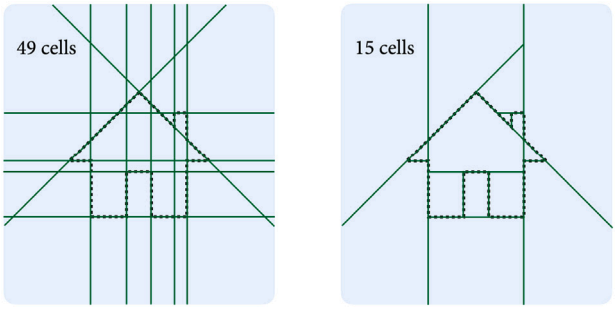


Fig. 4. Comparison of the exhaustive and adaptive partitioning strategies (a 2D example). Note the difference in the numbers of the resulted cells.

comparison of the exhaustive and our adaptive partitioning strategies. It is worth noting that in the 3D space, our adaptive space partitioning can significantly reduce the number of cells, allowing an efficient process in the subsequent steps.

During the partitioning, the BSP tree structure and adjacency information of the cells are incrementally obtained, as shown in Fig. 3. Thus the partitioning result depends highly on the order of inserting the planes. To this end, we prioritize the partitioning process such that planes conforming to building structural priors play dominant roles in partitioning the space. Considering that buildings typically demonstrate vertical walls, our space partitioning algorithm gives vertical planes a higher priority to be involved in partitioning. We define the verticality of a plane as $\mathcal{V} = 1 - |n_z|$, where n_z denotes the z coordinate of the plane's normal vector. A zero verticality value indicates a perfectly horizontal plane and a value of 1 indicates a perfectly vertical plane. Note that the verticality priority is only applied to planes whose verticality is greater than a threshold value (0.9 in our work). For the remaining planes, the priority is determined by the number of points in each plane because a larger number of points often implies a higher confidence value. The verticality priority ensures that vertically oriented planes partition the space before other planes do, which has an advantage in handling partially occluded facades. Fig. 5(b) shows such an example.

3.3. Occupancy learning

From the cell complex (denoted by C) obtained from adaptive binary space partitioning, we would like to extract a subset of the cells, i.e., a subcomplex $L \in C$, such that its occupancy O_L indicates the interior space enclosed by the outer surface of the building. To this end, we learn the occupancy of the building as a signed distance field (SDF), where the value is the distance d from a point $\mathbf{x}$ to the building surface

Algorithm 3.1: PLANE REFINEMENT (S)

Input: Raw planar segments S, angle tolerance θ and distance tolerance ϵ
Output: Refined planar segments $\widetilde{S}$

$\mathcal{Q} \leftarrow$ initialize priority queue;
for $(i, j) \in S$ **do**
  $\alpha_{i,j} \leftarrow$ compute angle between S_i and S_j;
  $\mathcal{Q} \leftarrow$ push $(\alpha_{i,j}, i, j)$ ordered by $\alpha_{i,j}$;
while $\mathcal{Q}$ *not empty* **do**
  $(\alpha_{i,j}, i, j) \leftarrow$ pop from $\mathcal{Q}$ with the smallest $\alpha_{i,j}$;
  $d_{i,j} \leftarrow$ compute distance between S_i and S_j;
  if $\alpha_{i,j} < \theta$ *and* $d_{i,j} < \epsilon$ **then**
    $m \leftarrow$ merge S_i and S_j with new plane parameters by PCA;
    $\alpha_{m,\mathbf{n}} \leftarrow$ compute angle between m and every plane $\mathbf{n}$ in $\mathcal{Q}$;
    $\mathcal{Q} \leftarrow$ push $(\alpha_{m,\mathbf{n}}, m, \mathbf{n})$ ordered by $\alpha_{m,\mathbf{n}}$;
  else
    $\widetilde{S} \leftarrow$ extract planar segments from $\mathcal{Q}$;
    break;
  return $\widetilde{S}$

and its sign indicates whether the point lies inside (with a positive sign) or outside (with a negative sign) the outer surface of the building:

$$SDF(\mathbf{x}) = d \ : \ \mathbf{x} \in \mathbb{R}^3, d \in \mathbb{R}. \tag{1}$$

Fig. 6 illustrates an example of the signed distance field at a series of cross-sections of a 3D building model.

Inspired by the work of Erler et al. (2020), we exploit a deep learning-based approach to learn a signed distance field from a point cloud. Specifically, we train a neural network that can predict the signed distance value for any point $\mathbf{x} \in \mathbb{R}^3$, i.e.,

$$f(\mathbf{x}) \approx \tilde{f}(\mathbf{x}) = s_\theta(\mathbf{x} \mid \mathbf{z}), \ \text{with } \mathbf{z} = e_\phi(P), \tag{2}$$

where $\mathbf{z}$ is a latent representation of the building surface encoded from the input point cloud P by an encoder e, and s represents the neural network. The encoder e and neural network s are parameterized by θ and ϕ, respectively. Following the neural network architecture of Erler et al. (2020), we decompose the signed distance field into two components: the absolute distance f^d and its sign f^s.

- **The absolute distance value**. The estimated absolute distance $\tilde{f}^d(\mathbf{x})$ can be determined from only the neighborhood of the query point:

$$\tilde{f}^d(\mathbf{x}) = s_\theta^d\left(\mathbf{x} \mid \mathbf{z}_\mathbf{x}^d\right), \ \text{with } \mathbf{z}_\mathbf{x}^d = e_\phi^d\left(\mathbf{p}_\mathbf{x}^d\right) \tag{3}$$

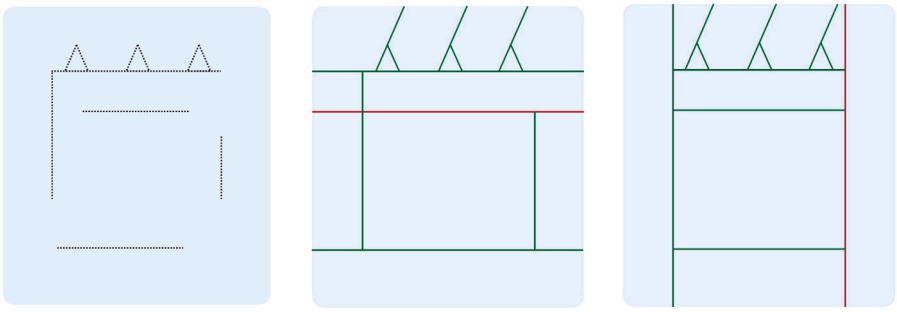


Fig. 5. An illustration of the effect of the verticality priority in space partitioning. (a) Incomplete input. (b) Without the verticality priority. Inserting the planar primitive denoted by the red line results in a missing wall in the upper-right part of the building. (c) With the verticality priority. By first inserting the vertical planar primitive (in red), a plausible vertical wall is inferred. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

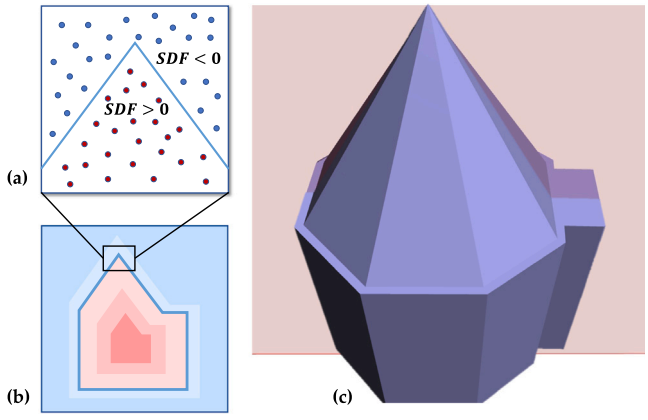


Fig. 6. An example of a signed distance field defined on a cross-section of a building model. (a) A visualization of the signed distance field, where the red color indicates the building's interior and the blue color indicates its exterior. (b) The signs of the signed distance field at a set of point samples. The color intensity reflects the distance values at a few discrete levels. (c) The 3D surface model. The cutting plane indicates where the signed distance field is defined. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where $\mathbf{p}_{\mathbf{x}}^{d} \in P$ is a set of neighboring points around query point $\mathbf{x}$.

- **The sign**. To estimate the sign $\tilde{f}^{s}(\mathbf{x})$ at $\mathbf{x}$, local sampling does not suffice because the occupancy information cannot be reliably estimated from the local neighborhood only. Therefore, a global uniform sub-sample $\mathbf{p}_{\mathbf{x}}^{s} \in P$ is taken as input:

$$\tilde{f}^{s}(\mathbf{x}) = \operatorname{sgn}\left(\tilde{g}^{s}(\mathbf{x})\right) = \operatorname{sgn}\left(s_{\theta}^{s}\left(\mathbf{x} \mid \mathbf{z}_{\mathbf{x}}^{s}\right)\right), \text{ with } \mathbf{z}_{\mathbf{x}}^{s} = e_{\psi}^{s}\left(\mathbf{p}_{\mathbf{x}}^{s}\right) \tag{4}$$

where ψ parameterizes the encoder, and $\tilde{g}^{s}(\mathbf{x})$ is a logit that expresses the confidence of $\mathbf{x}$ having a positive distance to the surface.

The two latent descriptions $\mathbf{z}_{\mathbf{x}}^{s}$ and $\mathbf{z}_{\mathbf{x}}^{d}$ share information, resulting in the formulation for signed distance learning, i.e.,

$$\left(\tilde{f}^{d}(\mathbf{x}), \tilde{g}^{s}(\mathbf{x})\right) = s_{\theta}\left(\mathbf{x} \mid \mathbf{z}_{\mathbf{x}}^{d}, \mathbf{z}_{\mathbf{x}}^{s}\right), \text{ with } \mathbf{z}_{\mathbf{x}}^{d} = e_{\phi}^{d}\left(\mathbf{p}_{\mathbf{x}}^{d}\right) \text{ and } \mathbf{z}_{\mathbf{x}}^{s} = e_{\psi}^{s}\left(\mathbf{p}_{\mathbf{x}}^{s}\right) \tag{5}$$

Fig. 7 illustrates a series of cross-sections of the signed distance field predicted by the neural network for the 3D model shown in Fig. 6. With the learned signed distance field, instead of performing Marching Cubes, we assign each cell $c_i \in C$ a signed distance value at its centroid. This sparse query can significantly reduce the computation for occupancy evaluation.

3.4. Surface extraction

With the cell complex and the occupancy of its cells obtained in the previous steps, surface reconstruction can be addressed by obtaining a consistent classification of the cells into *interior* and *exterior* categories, followed by an outer shell extraction step, as shown in Fig. 8. We address the interior/exterior cell classification using a Markov Random Field (MRF) formulation.

Given the cell complex $C = \{c_i\}$, we denote the binary label to be assigned to a cell c_i by $x_i \in \{in, out\}$. Our energy function is written as the weighted sum of two energy terms, i.e.,

$$E(x) = D(x) + \lambda V(x) \tag{6}$$

where $D(x)$ and $V(x)$ denote the data cost term and the smoothness cost term, respectively, and λ is the weight that balances the two terms.

- **Data cost**. We define the data cost term in a way such that it reflects the confidence of the classification. Specifically, it is defined to measure how much the classification differs from the previously estimated occupancy of the cells in the cell complex, i.e.,

$$D(X) = \frac{1}{|C|} \sum_{c_i \in C} \left| x_i - occupancy(c_i) \right|, \tag{7}$$

where $occupancy(c_i)$ denotes the learned occupancy of cell c_i, which can be computed as

$$occupancy(c_i) = sigmoid(SDF(c_i) \cdot volume(c_i)), \tag{8}$$

where $SDF(c_i)$ is the signed distance value of the query point at the centroid of c_i, predicted by the neural network, and $volume(c_i)$ is the volume of c_i. Intuitively, a cell with a larger volume should weigh higher regardless of its predicated signed distance. The sigmoid function $sigmoid(x) = \frac{1}{1+e^{-x}}$ normalizes the signed distance to the range $(0, 1)$.

- **Smoothness cost**. This energy term encourages assigning similar labels to the adjacent cells. We design this term in a way such that it penalizes the complexity of the output building surface model. Since the final surface will be eventually extracted as the outer shell of the interior cells, lowering its complexity is equivalent to limiting its surface area. Thus, our smoothness cost is defined as

$$V(X) = \frac{1}{A} \sum_{\{c_i, c_j\} \in C} a_{ij} \cdot \mathbb{1}(c_i, c_j), \tag{9}$$

where $\{c_i, c_j\} \in C$ represents a pair of adjacent cells in the complex. a_{ij} denotes the surface area of the common face of the two cells. A is a normalization factor that is chosen as the maximum area of all faces in the cell complex. $\mathbb{1}(c_i, c_j)$ is an

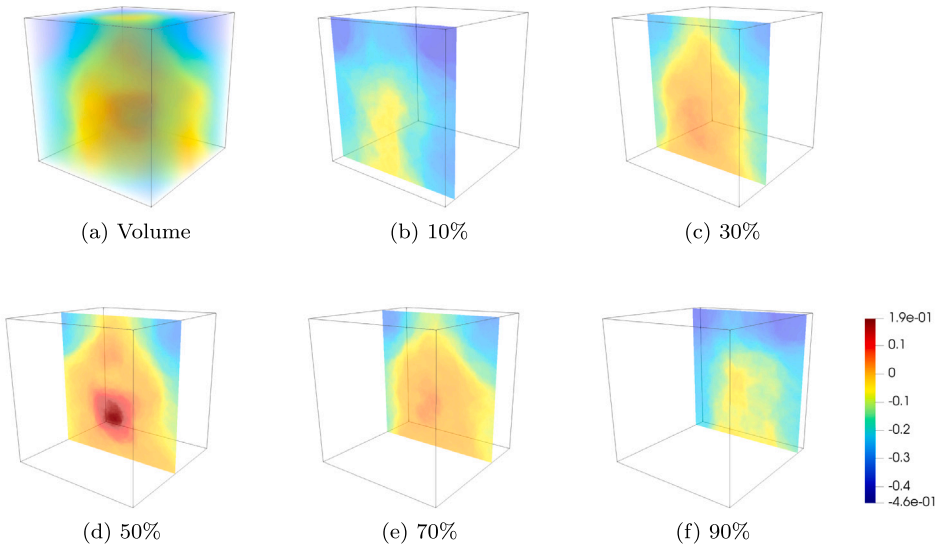


Fig. 7. A series of cross-sections of a signed distance field learned for the building shown in Fig. 6. (a) Volume rendering of the signed distance field. (b)–(f) are a few cross-sections of (a) where the percentages represent relatively where the sections are taken. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

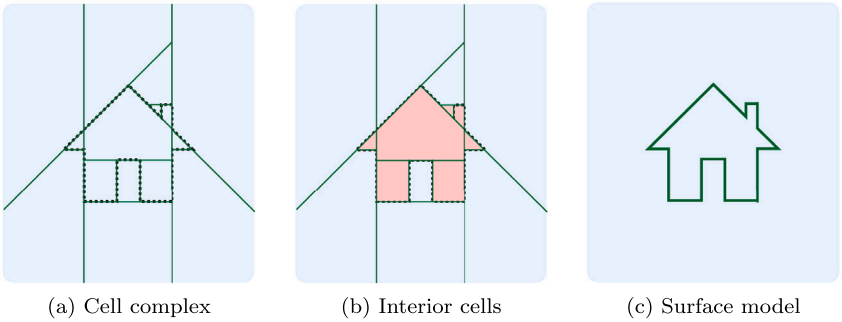


Fig. 8. An illustration of the surface extraction step. (a) The cell complex with the learned occupancy (the red and blue colors indicate interior and exterior, respectively). (b) The union of the interior cells. (c) The surface model was obtained by extracting the outer shell of the union of the interior cells. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

indicator function, which has the value of 1 if c_i and c_j receive different labels, i.e,

$$\mathbb{1}(c_i, c_j) = \begin{cases} 0, & x_i = x_j \\ 1, & x_i \neq x_j \end{cases} \tag{10}$$

Intuitively, the smoothness cost serves as a regularization term that penalizes zigzag artifacts on the final surface model, whose effect is illustrated in Fig. 9.

By minimizing the energy function given in Eq. (6) using the graph cut algorithm (Boykov and Funka-Lea, 2006), the interior cells can be obtained, as shown in Fig. 8(b). The final surface model is then obtained by extracting the outer surface of the union of the interior cells, which is illustrated in Fig. 8(c). As our adaptive binary space partitioning produces a valid polyhedral embedding, the surface is inherently guaranteed to be watertight.

4. Datasets

Sufficient high-quality point clouds and the corresponding surface models of real-world buildings are necessary for training and evaluating learning-based reconstruction methods. Due to the lack of such data, we have created a synthetic dataset on which we train our neural network for occupancy learning. We then evaluate the performance of the proposed method on both the synthetic dataset and a real-world dataset.

4.1. Synthetic dataset

To create the synthetic dataset for training our neural network and further evaluating our reconstruction method, we have created a synthetic dataset by simulating the scanning process based on the Helsinki LoD2 CityGML models (City of Helsinki, 2021). Specifically, we pick 678 watertight building meshes for training, 45 for validation, and another 45 for testing. Because of our patch-based architecture for occupancy learning, a large number of diverse patches are produced from each mesh as training samples.

The point clouds are generated by simulated scanning on the building mesh models. In pre-processing, the building meshes are translated to the origin and scaled uniformly to a unit length. To simulate the acquisition of a point cloud P on a building S, a LiDAR sensor is configured from random perspectives. We use Blensor (Gschwandtner et al., 2011) to simulate the scanning, intentionally with various levels of Gaussian noise and artifacts such as occlusions and light reflections.

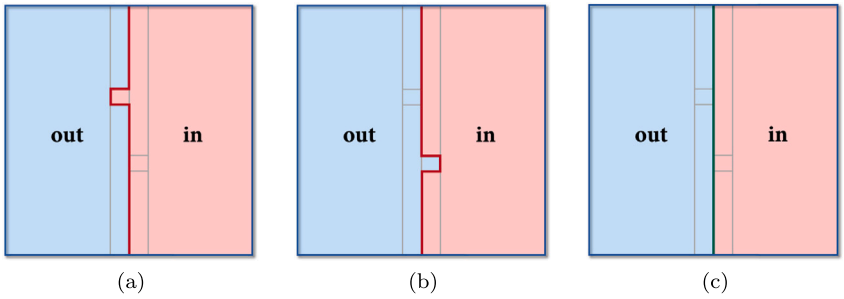


Fig. 9. An illustration of the effect of the smoothness cost energy term. (a) and (b) Interior–exterior classification result without the smoothness cost energy term. (c) Interior–exterior classification result with the smoothness cost energy term. The smoothness cost energy term penalizes zigzag artifacts and encourages a compact surface model.

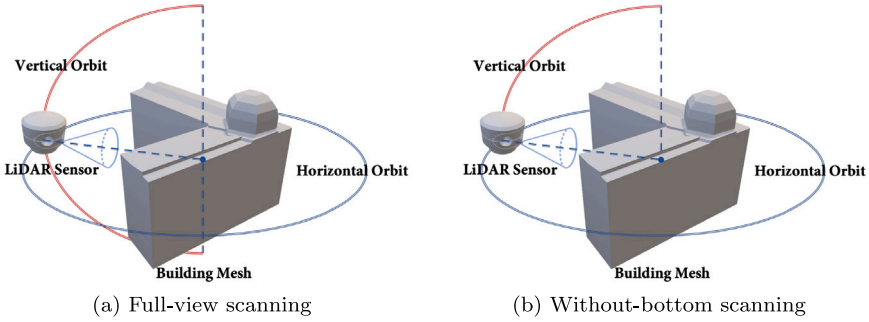


Fig. 10. Simulation of point cloud acquisition. (a) A LiDAR scanner rotates on a sphere around the building mesh to generate a full-view point cloud. (b) The scanner's position is constrained on the upper half of the sphere to simulate real-world scanning where the bottom of a building will not be captured.

To handle noise, we created multiple versions of the Helsinki dataset with different levels of noise, which is achieved by intentionally adding artificial Gaussian noise to the depth values during the virtual scanning process to augment the training data. For the training of the neural network, we obtain point clouds with Gaussian noise whose standard deviation is randomly chosen from the range $U[0, 0.005R]$, where R is the largest side of the building's bounding box, and U stands for uniform distribution. We also create multiple versions of point clouds for the same building sets, where the Gaussian noise is set to four different levels $\{0, 0.001R, 0.005R, 0.010R, 0.050R\}$, for evaluating the robustness of the proposed method and its competitors. In addition to the simulated point clouds, the training set also contains a set of query points X_S for each building. The query points are pre-sampled with known signed distance to the building's surfaces. Since query points with smaller absolute distances are of higher importance for detailed surface reconstruction, we randomly sample 1000 points on the surface and then apply perturbation in their normal direction by a random displacement in $U[-0.02R, 0.02R]$. In addition, we sample 1000 query points randomly distributed in the bounding box of the building, mounting up to 2000 query points in total per building. To enhance the robustness of the learned SDF, we randomly drop out 1000 query points and use the other 1000 samples for training. We denote this full-view dataset as *Helsinki full-view*.

To mimic the real-world scanning process and simulate occlusions in the dataset, we also constrain the scanner's position such that no bottom view is used (see Fig. 10(b)), to create another set of point clouds, denoted as *Helsinki no-bottom*. Similar to *Helsinki full-view*, we generate the point clouds with various Gaussian noise along with the query points for training, and a series of point clouds with fixed noise levels for evaluation.

4.2. Real-world dataset

Besides the synthetic data, we further evaluate our method on the real-world dataset consisting of photogrammetric point clouds of six buildings (Xie et al., 2021). These point clouds are obtained from aerial images using multi-view stereo techniques. The original images were captured by Unmanned Aerial Vehicles (UAV) from top and lateral perspectives. The main bodies of buildings are visible with well-captured roof structures, while the facades of the buildings are only partially visible with missing areas due to occlusions and poor lighting conditions in the lower part of the buildings. The point clouds in this dataset are demonstrated in the first column of Fig. 13.

4.3. Summary

Table 2 summarizes the characteristics of the datasets used for evaluation. Since global shape priors are dataset-dependent, the neural network trained on one dataset may not capture the characteristics of another. To this end, we train our neural network for SDF estimation on the *Helsinki full-view* dataset and *Helsinki no-bottom* dataset with various levels of noise, respectively. The trained model on the former is used for evaluation on the *full-view* set, while that of the latter on the *no-bottom* set.

5. Results and analysis

5.1. Reconstruction results

With the neural network trained on the *Helsinki full-view* point clouds, the proposed method can reconstruct buildings of various architectural styles. Fig. 11 presents the reconstruction results of eight buildings from the *Helsinki full-view* set. The SDF can accurately describe the occupancy of the buildings even though their styles may be distinct from the data on which the neural network was trained. The errors occur only when subtle structures exist, which is due to the uncertainty in planar primitive detection and the regularization imposed to surface extraction. Nonetheless, all building surfaces can be effectively retrieved from the point clouds with plausible visual quality and symmetric mean Hausdorff (SMH) distance of less than 0.3%.

Table 2
Datasets overview.

Name	Type	Perspective			Quantity	Usage
		Top	Bottom	Lateral		
Helsinki full-view	Simulated LiDAR	✓	✓	✓	768	Training + evaluation
Helsinki no-bottom	Simulated LiDAR	✓	✗	✓	768	Training + evaluation
Shenzhen	Real-world MVS	✓	✗	✓	6	Evaluation

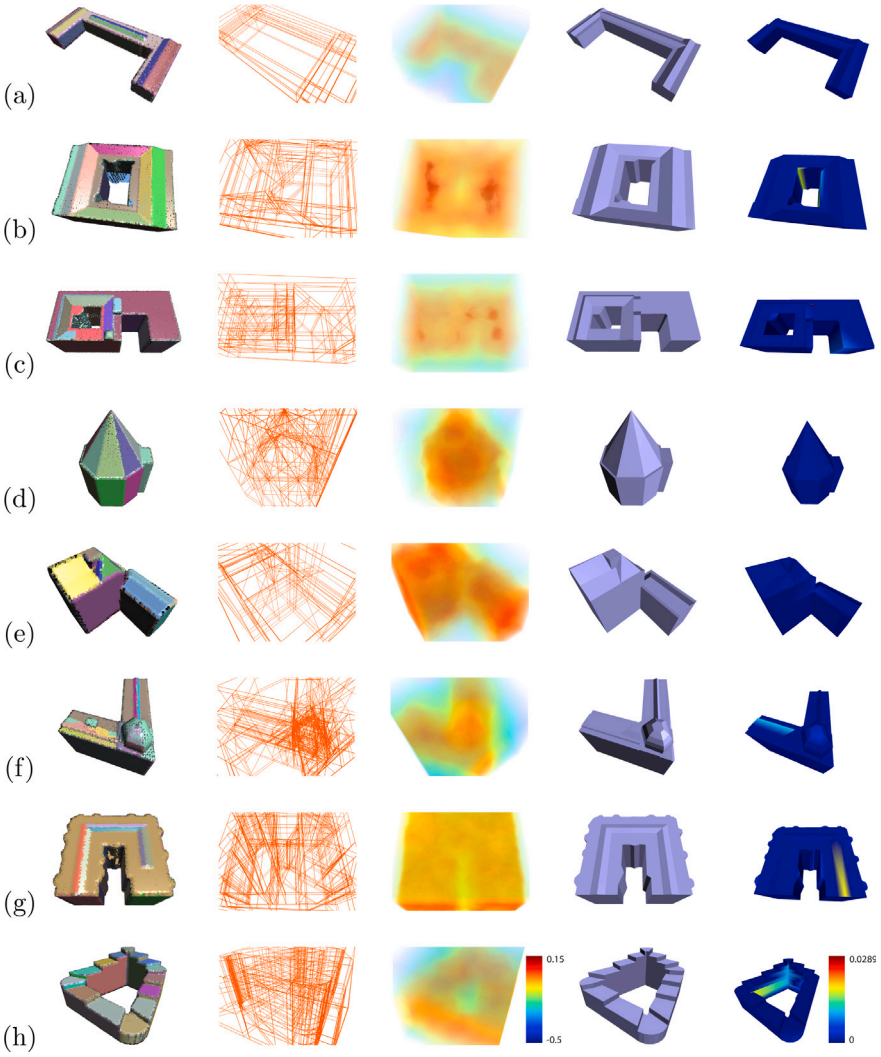


Fig. 11. Reconstruction results on the *Helsinki full-view* point clouds. From left to right: input point cloud (colored randomly per planar primitive), wireframe of the cell complex, volume rendering of the SDF, reconstructed building model, and error map. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We additionally train the neural network and evaluate it on the *Helsinki no-bottom* point clouds. As shown in Fig. 12, our method can still correctly infer the occupancy of the entire buildings regardless of the missing ground planes, resulting in complete reconstruction results. Unlike PolyFit (Nan and Wonka, 2017) that relies on a complete set of planes, our method can make use of the additional faces of the AABB of the point clouds, which guarantees to complete the missing surface models.[2] Besides, with the learned SDF, facades with few supporting points can be reliably reconstructed.

Since local features are used for occupancy estimation, and the training data is augmented with intentionally added noise and occlusions, our reconstruction can reasonably generalize from synthetic data to real-world point clouds. Fig. 13 shows the reconstruction results on the *Shenzhen* point clouds directly using the neural network trained on the *Helsinki no-bottom* dataset. The inferred distance fields still conform to the real-world buildings with styles unseen by the network during training. Though lower parts of the buildings are insufficiently measured, our method can still successfully reconstruct complete buildings. For the test on the *Shenzhen* dataset, since no ground truth surface models are available for quantitative evaluation

[2] This is specially designed for point clouds where no ground is captured.

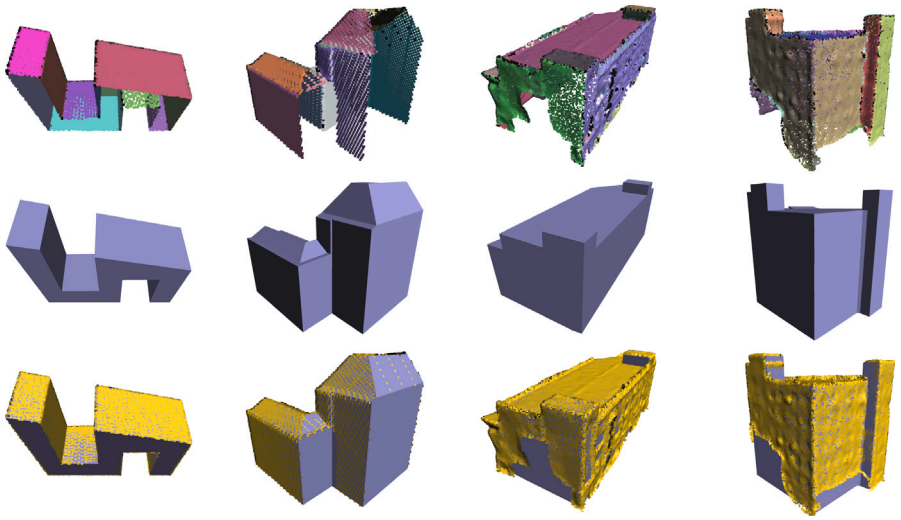


Fig. 12. Surface reconstruction from incomplete point clouds. Our method can reconstruct complete building models even if the input point clouds have missing regions on the bottom or facades. From top to bottom: input point clouds colored with the extracted planar primitives, reconstructed models, and an overlay of the two. The left two buildings are from *Helinki no-bottom*, and the right two are from *Shenzhen*. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

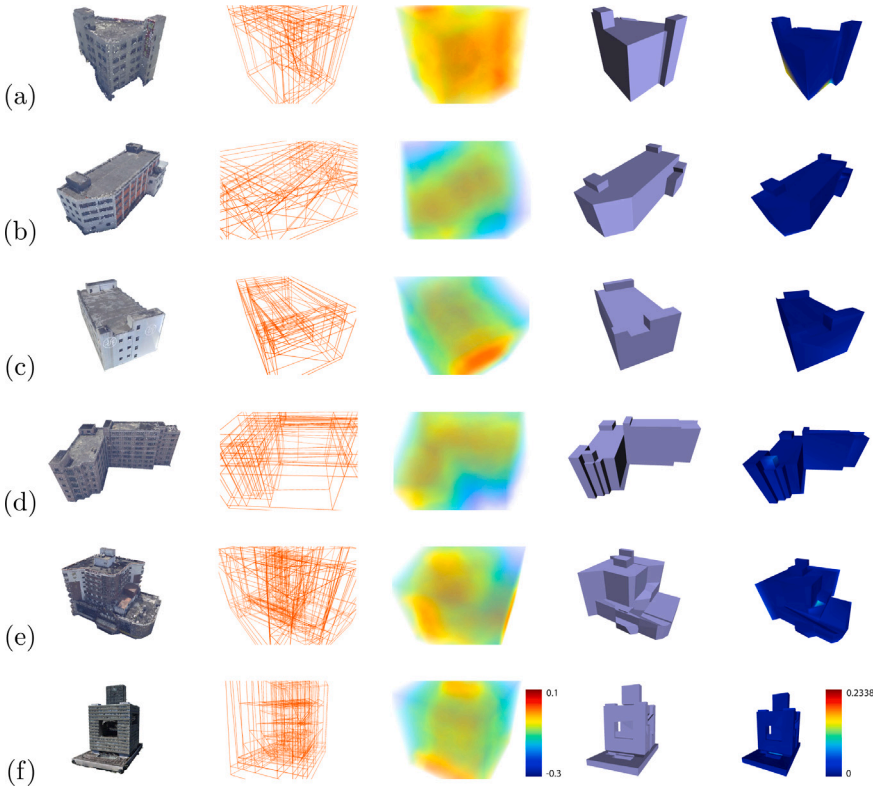


Fig. 13. Reconstruction results on the *Shenzhen* point clouds. From left to right: input point cloud (colored randomly per planar primitive), wireframe of the cell complex, volume rendering of the SDF, reconstructed building model, and error map. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of the reconstruction, we quantify reconstruction accuracy by measuring the Hausdorff distances from the reconstructed surfaces to the corresponding input point clouds. It is worth noting that the most prominent error seen from each error map lies in where the measurement is missing (i.e., some lower parts of the building were occluded in data acquisition). In addition, the complex roof structures occasionally introduce tangible errors when approximated with piecewise-planar abstraction. Fig. 13(d) shows such an example, where the protuberance on top of the building is approximated with a superfluous prism.

5.2. Evaluation

5.2.1. Fidelity and complexity

We have compared our method with various generic reconstruction, model-based approaches, geometry simplification, and primitive assembly methods.

Fig. 14 demonstrates the comparison with two generic reconstruction methods, namely Poisson surface reconstruction (Kazhdan et al.,

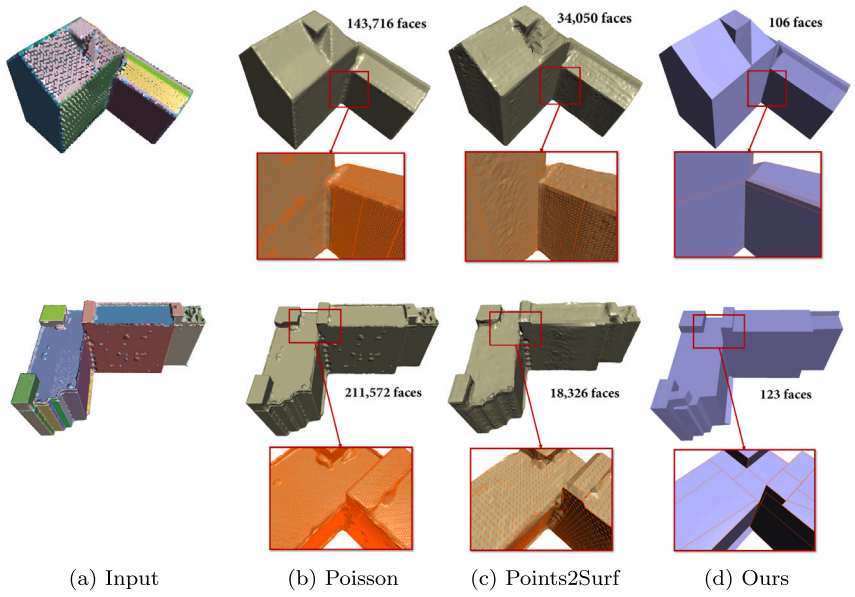


Fig. 14. Comparison between our method and two generic reconstruction methods, namely, Poisson surface reconstruction (Kazhdan et al., 2006) and Points2Surf (Erler et al., 2020). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

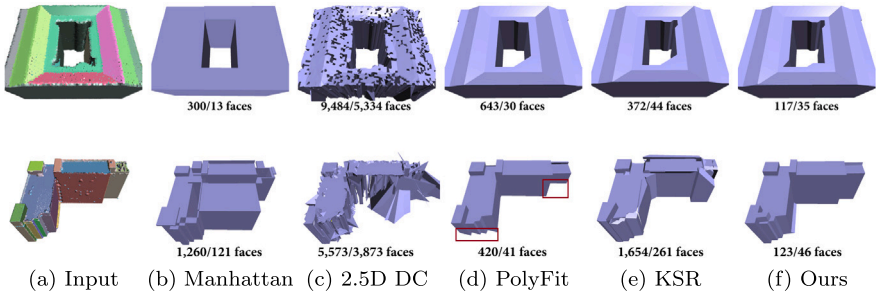


Fig. 15. Comparison between our method and two model-based methods, namely, Manhattan-world building reconstruction (Li et al., 2016c) and 2.5D Dual Contouring (Zhou and Neumann, 2010), and two primitive assembly based methods PolyFit (Nan and Wonka, 2017) and KSR (Bauchet and Lafarge, 2020). The building in the top row is from the *Helsinki full-view* dataset and the one in the bottom row is from the *Shenzhen* dataset. Two sets of face numbers (•/•) are given, which are the as-is face number from the algorithm and the face number after merging coplanar faces, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2006) and the learning-based method Points2Surf (Erler et al., 2020). Both methods produce surface models with a massive number of triangle faces and chamfered edges, which not only leads to extra memory consumption but also potentially artifacts such as bumps and holes. In contrast, our approach describes the building geometry as a lightweight watertight polyhedron with sharp edges.

In Fig. 15, we demonstrate the comparison of our results with two model-based building reconstruction methods, namely, Manhattan-world reconstruction (Li et al., 2016c) and 2.5D Dual Contouring (Zhou and Neumann, 2010), and two primitive assembly methods PolyFit (Nan and Wonka, 2017) and KSR (Bauchet and Lafarge, 2020), on two buildings from the *Helsinki full-view* and *Shenzhen* datasets, respectively. From the results, we can see that neither Manhattan-world reconstruction nor 2.5D Dual Contouring can deliver visually plausible building models. The former constrains faces to be axis-aligned, and it thus cannot faithfully recover planar faces of arbitrary orientation, while the latter produces a prohibitive number of faces that do not properly address the piecewise planarity of urban buildings but with undesirable discontinuity. Among these methods, the results from PolyFit and KSR are most comparable with ours in terms of compactness. PolyFit, KSR, and our method fall in the *hypothesizing-and-selection* strategy. Specifically, the three methods tessellate the ambient space into a candidate set with detected planar primitives and then seek a proper arrangement of the candidates with combinatorial analysis. The optimization goal in PolyFit is hard coded by the weights for data fitting, surface coverage, and model complexity objectives. In contrast, our optimization is essentially guided by the implicit field estimated from the input point cloud using the neural network and further regularized by the MRF in surface extraction. KSR's voting scheme requires the input point cloud to have high-quality normals associated, while our neural-guided approach enables directly consuming unorganized point clouds, which improves robustness on noisy or incomplete data.

On *Helsinki fullview*, the quantitative analysis presented in Table 3 reveals a marginal fidelity advantage of our method while using much fewer faces to represent the buildings. Also, our method can scale to buildings with higher complexity (for those PolyFit fails).

Compared with the exhaustive partitioning strategy adopted by PolyFit, our adaptive space partitioning can produce building models with lower complexity. This can be concluded by comparing the number of faces in the reconstructed surface models from these methods. In fact, the many polygonal faces originating from the same plane can be merged into a single face by a post-processing step but we did not do so in our paper. It is also worth noting that our adaptive space partitioning cannot be plugged into PolyFit because it would otherwise break PolyFit's pre-condition that every edge is shared by four candidate faces (except for border edges). Nevertheless, PolyFit performs

Table 3
Quantitative evaluation among Poisson (Kazhdan et al., 2006), PolyFit (Nan and Wonka, 2017), KSR (Bauchet and Lafarge, 2020), and ours. (•, •) denotes (Hausdorff distance, number of faces of a model from each original algorithm). "-" indicates failure in obtaining the result within 3-hour running of an algorithm.

Index	Poisson	PolyFit	KSR	Ours	Index	Poisson	PolyFit	KSR	Ours
1	(0.1992, 417546)	(0.2004, 33)	(0.2007, 60)	(0.2013, 23)	24	(0.1022, 245028)	–	(0.1056, 1382)	(0.0714, 458)
2	(0.0433, 63440)	(0.0433, 269)	(0.0435, 408)	(0.0433, 145)	25	(0.0263, 67000)	(0.0167, 54)	(0.0167, 64)	(0.0180, 41)
3	(0.0173, 178608)	(0.0165, 42)	(0.0168, 60)	(0.0177, 32)	26	(0.1247, 216590)	(0.1247, 45)	(0.1246, 62)	(0.1247, 26)
4	(0.0481, 226278)	(0.2130, 60)	(0.0483, 74)	(0.0478, 37)	27	(0.0828, 300112)	(0.0829, 68)	(0.0830, 68)	(0.0829, 51)
5	(0.0573, 163084)	(0.0422, 231)	(0.0463, 180)	(0.0420, 75)	28	(0.0938, 215042)	–	(0.0933, 610)	(0.1105, 304)
6	(0.1518, 405996)	–	(0.1464, 832)	(0.2179, 249)	29	(0.0797, 234884)	(0.1024, 56)	(0.0795, 74)	(0.2099, 53)
7	(0.1184, 302234)	(0.1252, 442)	(0.2016, 270)	(0.1367, 336)	30	(0.1564, 174826)	(0.1573, 25)	(0.1574, 38)	(0.1574, 14)
8	(0.0889, 133070)	(0.0887, 318)	(0.0888, 208)	(0.0890, 164)	31	(0.0454, 176860)	(0.0454, 127)	(0.0464, 136)	(0.0452, 53)
9	(0.0746, 277654)	(0.1563, 643)	(0.2468, 358)	(0.0457, 148)	32	(0.0396, 140286)	(0.0484, 220)	(0.0410, 174)	(0.0952, 64)
10	(0.0611, 198032)	–	(0.0601, 1088)	(0.0635, 349)	33	(0.0267, 252528)	(0.0518, 344)	(0.0520, 232)	(0.1956, 141)
11	(0.0390, 116550)	–	(0.1846, 442)	(0.0370, 137)	34	(0.2245, 287140)	(0.2287, 180)	(0.2220, 322)	(0.2203, 127)
12	(0.0622, 214442)	(0.0601, 816)	(0.1899, 340)	(0.0599, 186)	35	(0.0153, 155978)	(0.0306, 283)	(0.0297, 210)	(0.0302, 110)
13	(0.0167, 107360)	(0.0183, 66)	(0.0170, 88)	(0.0167, 46)	36	(0.0472, 458574)	(0.0461, 37)	(0.0460, 50)	(0.0457, 35)
14	(0.1307, 398462)	(0.1313, 311)	(0.1316, 474)	(0.1434, 124)	37	(0.0152, 145496)	–	(0.0610, 6218)	(0.0970, 918)
15	(0.0702, 181472)	–	(0.0714, 828)	(0.0697, 294)	38	(0.2209, 143716)	(0.2222, 271)	(0.2220, 222)	(0.2218, 105)
16	(0.0491, 96932)	–	(0.0499, 706)	(0.0486, 329)	39	(0.1427, 126976)	–	(0.1402, 848)	(0.1445, 315)
17	(0.0723, 124446)	(0.0668, 58)	(0.0673, 104)	(0.0676, 48)	40	(0.0828, 129036)	(0.0718, 548)	(0.0748, 362)	(0.1849, 99)
18	(0.0937, 118946)	(0.0935, 25)	(0.0933, 40)	(0.0934, 20)	41	(0.0325, 339120)	(0.0285, 71)	(0.0297, 284)	(0.0280, 36)
19	(0.0166, 157064)	(0.0277, 90)	(0.0286, 124)	(0.0278, 84)	42	(0.0654, 78496)	–	(0.0662, 626)	(0.0731, 308)
20	(0.0845, 148874)	–	(0.0822, 300)	(0.0821, 157)	43	(0.0787, 107480)	(0.0785, 64)	(0.0800, 64)	(0.0780, 35)
21	(0.0196, 93014)	(0.0688, 76)	(0.0172, 104)	(0.1456, 44)	44	(0.0164, 168376)	(0.1454, 65)	(0.0187, 64)	(0.1631, 37)
22	(0.0193, 245844)	(0.0216, 852)	(0.0254, 576)	(0.0213, 228)	45	(0.0432, 104310)	(0.0449, 158)	(0.0453, 150)	(0.0457, 47)
23	(0.1023, 160638)	–	(0.1019, 888)	(0.1025, 292)					

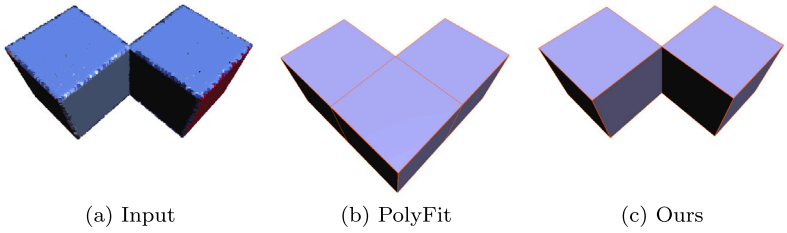


Fig. 16. Comparison between PolyFit (Nan and Wonka, 2017) and our method on a building that demonstrates a non-manifold structure. Our method faithfully reconstructs the surface, while PolyFit impertinently enforces manifoldness, resulting in incorrect reconstruction.

pairwise intersection of all input planes and thus the partitioning by the vertical plane is always guaranteed because every pair of the support planes will intersect and result in four candidate faces. In this sense, verticality is not a concern for PolyFit.

Our experiments reveal that introducing the vertical priority leads to slightly higher accuracy, i.e., 0.9% less Hausdorff distance on average. We would like to point out that this priority has not been exploited in the previous work. We have also observed in our experiments that non-manifold structures can significantly affect surface reconstruction, as exemplified by Fig. 16. Since our method constrains only water tightness, the reconstructed surface respects faithfully the geometry of the point cloud that exhibits a non-manifold structure, which PolyFit fails due to its manifoldness hard constraint.

In addition to the methods from the reconstruction category, we further compared our method with a few commonly used shape simplification methods, namely, QEM (Garland and Heckbert, 1997), SAMD (Salinas et al., 2015), VSA (Cohen-Steiner et al., 2004), and FPS (Li and Nan, 2021). For these methods, we set their expected number of vertices to be comparable to the models generated by our method, which ensures that the quality of the generated building models is compared at the same complexity level. Notice that, however, the resulting number of faces is not guaranteed to be identical to ours because no such constraints can be asserted with the three surface approximation methods. As can be seen from Fig. 17, VSA produces surfaces with the worst quality given the same complexity. SAMD, though claiming to be structure-aware, fails to deliver surfaces with compact planes, partially because of its strong compliance to the detected primitives, which on the contrary limits its compactness when the primitives are of low quality. QEM generates fractured surfaces because its quadric error metric contracts edges without attention to the piecewise planarity of the buildings. Nevertheless, the results from QEM are more plausible than VSA and SAMD. FPS generates the most visually pleasing results among the four methods because of its capability of preserving piecewise-planar structures and sharp features. However, the results are not as regular as ours. In contrast, our method produces surface models directly from point clouds, which recovers stronger piecewise planarity and meanwhile preserves sharp features.

5.2.2. Computational efficiency

Our adaptive space partitioning can significantly reduce the computations for cell complex creation, compared to an exhaustive partitioning strategy. With exhaustive partitioning, a massive number of candidate polyhedra are produced and the running time increases accordingly. A comparison between our adaptive space partitioning and the exhaustive partitioning in terms of the number of resulting cells and the running time is presented in Fig. 18. The excessive number of cells not only hinders computation but also inclines to defective surfaces on subtle structures where inaccurate labels are more likely to be assigned. Instead, the adaptive strategy avoids redundant partitioning and thus produces compact surfaces efficiently.

We further compared our method with PolyFit (Nan and Wonka, 2017) and KSR (Bauchet and Lafarge, 2020) in terms of scalability. The results are shown in Fig. 19. PolyFit is based on the exhaustive partitioning strategy. Specifically, with pairwise intersection from the initial set of planar primitives, PolyFit generates a prohibitively large number of candidate faces, from which an optimal subset of the faces is to be selected using optimization based on its linear integer programming. In our experiments, PolyFit failed to reconstruct the surface when

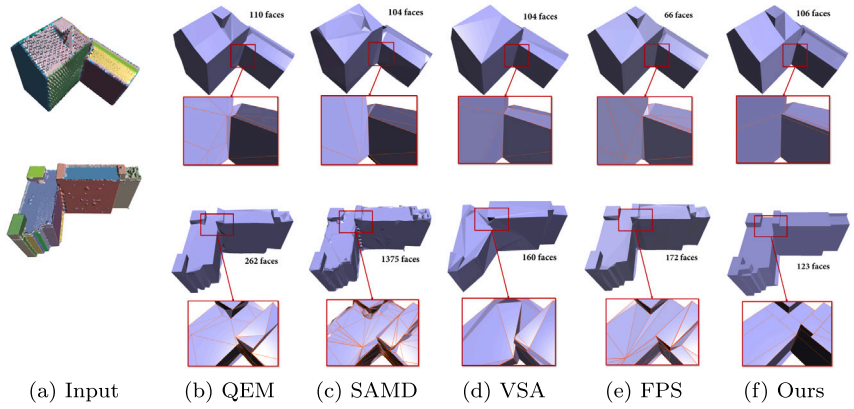


Fig. 17. Comparison between the results of our method and three geometry simplification methods, namely, QEM (Garland and Heckbert, 1997), SAMD (Salinas et al., 2015), VSA (Cohen-Steiner et al., 2004), and FPS (Li and Nan, 2021). Note that QEM, SAMD, VSA, and FPS all output triangle surfaces while our results are arbitrary-sided polygonal surfaces. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

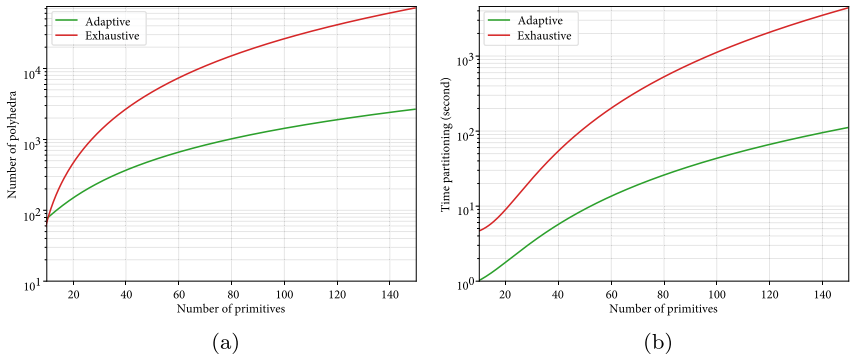


Fig. 18. Comparison between our adaptive space partitioning and exhaustive partitioning in terms of the number of resulting cells (a) and the running time for partitioning (b).

the number of planar primitives exceeds 100. In contrast, our adaptive binary space partitioning resulted in a significantly smaller number of cells given the same number of input planar primitives, which allows our method to process input point clouds with more than one thousand planar primitives. For this building with 260 detected primitives, PolyFit generated 762,439 candidate faces from the planar primitives and failed to solve the resulted large linear integer program. In contrast, our method successfully produced a high-quality surface model in 150 s, of which the combinatorial optimization takes only 0.13 s. KSR has higher efficiency than ours when the number of input primitives is smaller than 150. When scaling to inputs of higher complexity, our method excels since the employed adaptive strategy implies a strong locality, as opposed to the globally kinetic structure in KSR. It is also worth noting the potential implementation-wise performance gap, to which the computation overhead may be ascribed especially with a lower number of input primitives.[3]

5.2.3. Robustness to noise and incomplete input

To cope with noisy data, our network was trained on the synthetic point clouds with intentionally added different levels of noise. To evaluate the robustness of our method against noise, we tested our method on point clouds with increasing levels of Gaussian noise. Fig. 20 demonstrates such an example. Though our method was trained on point clouds with low noise levels in the range $[0, 0.005R]$, it produced reliable reconstruction for point clouds with surprisingly higher levels of noise (i.e., until $0.05R$, which is equivalent to measurement errors of as high as 5 meters for a 100 meter-sided building).

[3] KSR and PolyFit are implemented in C++, while ours is in Python.

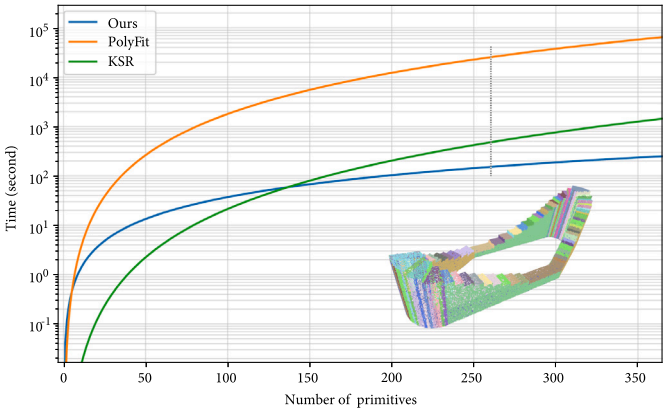
Since our neural network takes the global subsample of the point cloud into account in occupancy learning, it captures the global structures of the buildings, making it robust to incomplete input. Examples of such reconstruction on the *Helsinki no-bottom* point clouds are shown in Fig. 12 and on the *Shenzhen* point clouds shown in Fig. 13. In these examples, the ground planes in the input point clouds are completely missing, and the lower parts of some buildings in the *Shenzhen* dataset are also occluded. With such incomplete input, our method successfully produced complete 3D models.

5.2.4. Effect of parameter λ

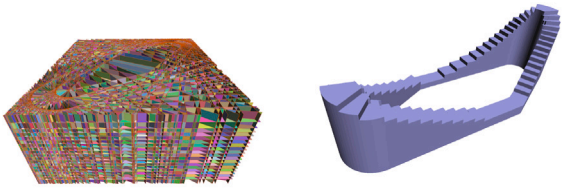
The parameter λ in Eq. (6) weights the complexity term formulated in the MRF for surface extraction from the cell complex, which controls the complexity of the final surface model. Specifically, increasing λ leads to a more compact surface with fewer faces, while too large values of λ may result in over-simplified surface models. Fig. 21 demonstrates the effect of λ on the final reconstruction. In all other experiments, λ was set to 0.001, which was chosen by trial and error.

5.3. Limitations

Similar to existing piecewise-planar object reconstruction methods such as PolyFit (Nan and Wonka, 2017), our work focuses on assembling the initially detected planar primitives into compact polygonal building models, rather than on detecting the planar primitives. We thus have assumed that dominant planes (e.g., walls and roofs) can be detected from input point clouds. However, this is not always possible for noisy or incomplete scans. In our work, though a few steps are designed to mitigate the inaccuracy from primitive detection, including



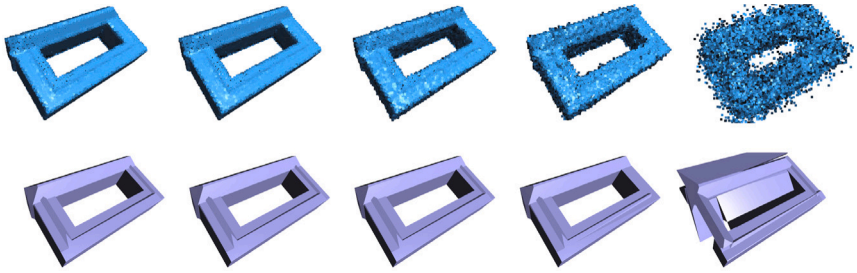
(a) Performances of reconstruction. The complex building with 260 planar primitives is reconstructed by KSR and ours in 480 seconds and 150 seconds, respectively. Statistics were drawn from ten models with various complexity.



(b) PolyFit generates 762,439 candidate faces and fails to solve the resulting integer program.

(c) Our method generates 2,856 candidate cells and produces a faithful reconstruction.

Fig. 19. Scalability comparison between our method, PolyFit (Nan and Wonka, 2017), and KSR (Bauchet and Lafarge, 2020).



(a) $0R$ (b) $0.001R$ (c) $0.005R$ (d) $0.010R$ (e) $0.050R$

Fig. 20. Robustness to noise. Note that our neural network was trained on point clouds with different levels of noise in the range $[0, 0.005R]$, where R denotes the radius of the bounding sphere of the input point cloud.

primitive refinement and the complexity term in the MRF formulation, it may still fail when the provided primitives are not complete or have large errors. Fig. 22 shows an example of how the quality of planar primitives impacts the reconstruction.

6. Conclusion

We have presented a novel efficient method for urban building reconstruction by exploiting the learned implicit representation as an occupancy indicator for explicit geometry extraction. With our occupancy learning strategy and the MRF formulation, we demonstrate that high-quality building models can be reconstructed with significant advantages in terms of fidelity, compactness, and computational efficiency. To the best of our knowledge, this is the first work where a deep implicit field is explored for building reconstruction.

In future work, we would like to extend the current method to an end-to-end pipeline by incorporating the construction of explicit geometry into a neural network. In addition, our MRF-based surface extraction is efficient enough to allow interactive editing. We plan to integrate user interactions into the reconstruction pipeline to further improve the usability of the method for challenging reconstruction scenarios.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

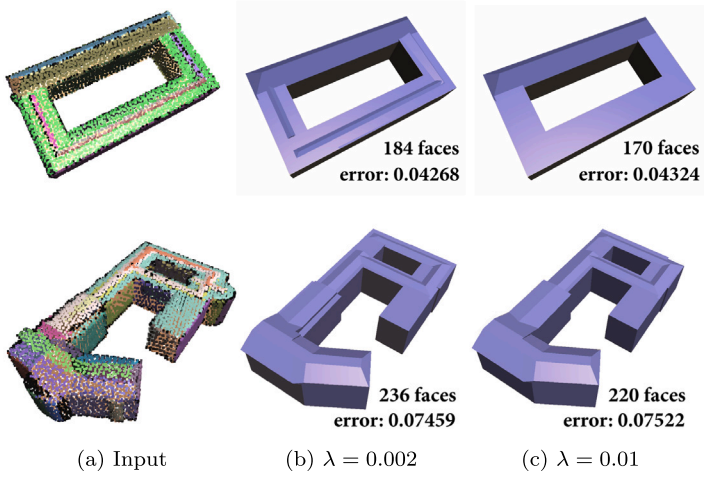


Fig. 21. The effect of parameter λ on the final reconstruction. Increasing λ leads to more compact surface models with fewer faces but potentially larger geometric errors.

Fig. 22. The impact of the quality of detected planar primitives on the final reconstruction. Left: given accurately detected planar primitives, our method reconstructs high-quality building models. Right: the reconstruction result from inaccurately detected planar primitives.

References

Arikan, M., Schwärzler, M., Flöry, S., Wimmer, M., Maierhofer, S., 2013. O-Snap: Optimization-based snapping for modeling architecture. ACM Trans. Graph. 32 (1), 1–15.

Bauchet, J.-P., Lafarge, F., 2020. Kinetic shape reconstruction. ACM Trans. Graph. 39 (5), 1–14.

Berger, M., Tagliasacchi, A., Seversky, L.M., Alliez, P., Guennebaud, G., Levine, J.A., Sharf, A., Silva, C.T., 2017. A survey of surface reconstruction from point clouds. In: Computer Graphics Forum, Vol. 36, no. 1. Wiley Online Library, pp. 301–329.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., Çöltekin, A., 2015. Applications of 3D city models: State of the art review. ISPRS Int. J. Geo-Inf. 4 (4), 2220–9964.

Blut, C., Blankenbach, J., 2021. Three-dimensional citygml building models in mobile augmented reality: A smartphone-based pose tracking system. Int. J. Digit. Earth 14 (1), 32–51.

Boulch, A., de La Gorce, M., Marlet, R., 2014. Piecewise-planar 3D reconstruction with edge and corner regularization. Computer Graphics Forum 33 (5), 55–64.

Bouzas, V., Ledoux, H., Nan, L., 2020. Structure-aware building mesh polygonization. ISPRS J. Photogramm. Remote Sens. 167, 432–442.

Boykov, Y., Funka-Lea, G., 2006. Graph cuts and efficient ND image segmentation. Int. J. Comput. Vis. 70 (2), 109–131.

Chauve, A.-L., Labatut, P., Pons, J.-P., 2010. Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE, pp. 1261–1268.

Chen, J., Chen, B., 2008. Architectural modeling from sparsely scanned range data. Int. J. Comput. Vis. 78 (2–3), 223–236.

Chen, Z., Tagliasacchi, A., Zhang, H., 2020. BSP-Net: Generating compact meshes via binary space partitioning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 45–54.

City of Helsinki, 2021. Helsinki 3D+. URL: https://kartta.hel.fi/3d/.

Cohen-Steiner, D., Alliez, P., Desbrun, M., 2004. Variational shape approximation. In: ACM SIGGRAPH 2004 Papers. pp. 905–914.

Coughlan, J.M., Yuille, A.L., 2000. The Manhattan world assumption: Regularities in scene statistics which enable Bayesian inference. In: NIPS, Vol. 2. p. 3.

Deng, B., Genova, K., Yazdani, S., Bouaziz, S., Hinton, G., Tagliasacchi, A., 2020. CvxNet: Learnable convex decomposition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 31–44.

Erler, P., Guerrero, P., Ohrhallinger, S., Mitra, N.J., Wimmer, M., 2020. Points2Surf: Learning implicit surfaces from point clouds. In: European Conference on Computer Vision. Springer, pp. 108–124.

Fang, H., Lafarge, F., 2020. Connect-and-Slice: An hybrid approach for reconstructing 3D objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13490–13498.

Garland, M., Heckbert, P.S., 1997. Surface simplification using quadric error metrics. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. pp. 209–216.

Gschwandtner, M., Kwitt, R., Uhl, A., Pree, W., 2011. Blensor: Blender sensor simulation toolbox. In: International Symposium on Visual Computing. Springer, pp. 199–208.

Han, J., Zhu, L., Gao, X., Hu, Z., Zhou, L., Liu, H., Shen, S., 2021. Urban scene LOD vectorized modeling from photogrammetry meshes. IEEE Trans. Image Process. 30, 7458–7471.

Herbert, G., Chen, X., 2015. A comparison of usefulness of 2D and 3D representations of urban planning. Cartogr. Geogr. Inf. Sci. 42 (1), 22–32.

Ikehata, S., Yang, H., Furukawa, Y., 2015. Structured indoor modeling. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1323–1331.

Kazhdan, M., Bolitho, M., Hoppe, H., 2006. Poisson surface reconstruction. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, Vol. 7.

Kazhdan, M., Hoppe, H., 2013. Screened Poisson surface reconstruction. ACM Trans. Graph. (ToG) 32 (3), 1–13.

Kelly, T., Femiani, J., Wonka, P., Mitra, N.J., 2017. BigSUR: Large-scale structured urban reconstruction. ACM Trans. Graph. 36 (6).

Labatut, P., Pons, J.-P., Keriven, R., 2009. Hierarchical shape-based surface reconstruction for dense multi-view stereo. In: 2009 IEEE 12th International Conference on Computer Vision Workshops. ICCV Workshops, IEEE, pp. 1598–1605.

Lafarge, F., Alliez, P., 2013. Surface reconstruction through point set structuring. In: Computer Graphics Forum, Vol. 32, no. 2pt2. Wiley Online Library, pp. 225–234.

Li, M., Nan, L., 2021. Feature-preserving 3D mesh simplification for urban buildings. ISPRS J. Photogramm. Remote Sens. 173, 135–150.

Li, M., Nan, L., Liu, S., 2016a. Fitting boxes to Manhattan scenes using linear integer programming. Int. J. Digit. Earth 9 (8), 806–817.

Li, M., Nan, L., Smith, N., Wonka, P., 2016b. Reconstructing building mass models from UAV images. Comput. Graph. 54, 84–93.

Li, M., Wonka, P., Nan, L., 2016c. Manhattan-world urban reconstruction from point clouds. In: European Conference on Computer Vision. Springer, pp. 54–69.

Li, Y., Wu, B., 2021. Relation-constrained 3D reconstruction of buildings in metropolitan areas from photogrammetric point clouds. Remote Sens. 13 (1), 129.

Lorensen, W.E., Cline, H.E., 1987. Marching cubes: A high resolution 3D surface construction algorithm. In: ACM SIGGRAPH Computer Graphics, Vol. 21, no. 4. ACM New York, NY, USA, pp. 163–169.

Machete, R., Falcão, A.P., Gomes, M.G., Rodrigues, A.M., 2018. The use of 3D GIS to analyse the influence of urban context on buildings' solar energy potential. Energy Build. 177, 290–302.

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

Mura, C., Mattausch, O., Pajarola, R., 2016. Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. In: Computer Graphics Forum, Vol. 35, no. 7. Wiley Online Library, pp. 179–188.

Murali, T., Funkhouser, T.A., 1997. Consistent solid and boundary representations from arbitrary polygonal data. In: Proceedings of the 1997 Symposium on Interactive 3D Graphics. pp. 155–ff.

Nan, L., Wonka, P., 2017. PolyFit: Polygonal surface reconstruction from point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2353–2361.

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

Salinas, D., Lafarge, F., Alliez, P., 2015. Structure-aware mesh decimation. In: Computer Graphics Forum, Vol. 34, no. 6. Wiley Online Library, pp. 211–227.

Schindler, F., Wörstner, W., Frahm, J.-M., 2011. Classification and reconstruction of surfaces from point clouds of man-made objects. In: 2011 IEEE International Conference on Computer Vision Workshops. ICCV Workshops, IEEE, pp. 257–263.

Schnabel, R., Wahl, R., Klein, R., 2007. Efficient RANSAC for point-cloud shape detection. In: Computer Graphics Forum, Vol. 26, no. 2. Wiley Online Library, pp. 214–226.

Stoter, J., Peters, R., Commandeur, T., Dukai, B., Kumar, K., Ledoux, H., 2020. Automated reconstruction of 3D input data for noise simulation. Comput. Environ. Urban Syst. 80, 101424.

Van Kreveld, M., Van Lankveld, T., Veltkamp, R.C., 2011. On the shape of a set of points and lines in the plane. In: Computer Graphics Forum, Vol. 30, no. 5. Wiley Online Library, pp. 1553–1562.

Verdie, Y., Lafarge, F., Alliez, P., 2015. LOD generation for urban scenes. ACM Trans. Graph. 34 (ARTICLE), 30.

Xie, L., Hu, H., Zhu, Q., Li, X., Tang, S., Li, Y., Guo, R., Zhang, Y., Wang, W., 2021. Combined rule-based and hypothesis-based method for building model reconstruction from photogrammetric point clouds. Remote Sens. 13 (6), 1107.

Xiong, B., Elberink, S.O., Vosselman, G., 2014. A graph edit dictionary for correcting errors in roof topology graphs reconstructed from point clouds. ISPRS J. Photogramm. Remote Sens. 93, 227–242.

Xiong, B., Jancosek, M., Elberink, S.O., Vosselman, G., 2015. Flexible building primitives for 3D building modeling. ISPRS J. Photogramm. Remote Sens. 101, 275–290.

Zhou, Q.-Y., Neumann, U., 2010. 2.5D Dual Contouring: A robust approach to creating building models from aerial LiDAR point clouds. In: European Conference on Computer Vision. Springer, pp. 115–128.

Zhu, L., Shen, S., Gao, X., Hu, Z., 2018. Large scale urban scene modeling from MVS meshes. In: Proceedings of the European Conference on Computer Vision. ECCV, pp. 614–629.