ISPRS Journal of Photogrammetry and Remote Sensing 218 (2024) 693–706

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

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

PolyGNN: Polyhedron-based graph neural network for 3D building reconstruction from point clouds

Zhaiyu Chen [a], Yilei Shi [b], Liangliang Nan [c], Zhitong Xiong [a], Xiao Xiang Zhu [a,d,*]

[a] Chair of Data Science in Earth Observation, Technical University of Munich, 80333, Munich, Germany
[b] School of Engineering and Design, Technical University of Munich, 80333, Munich, Germany
[c] Urban Data Science, Delft University of Technology, 2628 BL, Delft, The Netherlands
[d] Munich Center for Machine Learning, 80333, Munich, Germany

ARTICLE INFO

Keywords:
3D reconstruction
Building model
Graph neural network
Point cloud
Polyhedron

ABSTRACT

We present PolyGNN, a polyhedron-based graph neural network for 3D building reconstruction from point clouds. PolyGNN learns to assemble primitives obtained by polyhedral decomposition via graph node classification, achieving a watertight and compact reconstruction. To effectively represent arbitrary-shaped polyhedra in the neural network, we propose a skeleton-based sampling strategy to generate polyhedron-wise queries. These queries are then incorporated with inter-polyhedron adjacency to enhance the classification. PolyGNN is end-to-end optimizable and is designed to accommodate variable-size input points, polyhedra, and queries with an index-driven batching technique. To address the abstraction gap between existing city-building models and the underlying instances, and provide a fair evaluation of the proposed method, we develop our method on a large-scale synthetic dataset with well-defined ground truths of polyhedral labels. We further conduct a transferability analysis across cities and on real-world point clouds. Both qualitative and quantitative results demonstrate the effectiveness of our method, particularly its efficiency for large-scale reconstructions. The source code and data are available at https://github.com/chenzhaiyu/polygnn.

1. Introduction

Three-dimensional (3D) building models constitute an important infrastructure in shaping digital twin cities, facilitating a broad range of applications including urban planning, energy demand estimation, and environmental analysis (Biljecki et al., 2015). Therefore, efficient reconstruction of high-quality 3D building models is crucial for understanding an urban environment and has been a long-standing challenge.

Most reconstruction methods are dedicated to detailed surfaces represented by dense triangles (Kazhdan and Hoppe, 2013; Erler et al., 2020), irrespective of the ubiquitous piecewise planarity in the built environment. Alternatively, a compact polygonal representation with sparse parameters can adequately capture the geometry of urban buildings. To reconstruct compact polygonal building models, three categories of methods are commonly employed in practice. Constrained reconstruction methods (Zhou and Neumann, 2010; Li et al., 2016b) represent buildings with pre-defined templates or specific topologies. However, the limited variety of available templates or topologies constrains the expressiveness of these methods. Geometric simplification methods (Bouzas et al., 2020; Li and Nan, 2021) aim to obtain compact surfaces by simplifying dense triangle ones. These techniques, however, necessitate an input model that is precise in both its geometry and topology to ensure a faithful approximation. Primitive assembly methods (Nan and Wonka, 2017; Huang et al., 2022) produce polygonal surface models by pursuing an optimal assembly of a collection of geometric primitives. However, these methods often entail the use of handcrafted features and thus possess limited representational capacity.

Despite successes in various other applications, learning-based solutions for compact building modeling remain largely unexplored. Notably, Points2Poly (Chen et al., 2022) stands out as a pioneering effort with the primitive assembly strategy. This method utilizes an implicit representation to learn building occupancy, followed by a Markov random field (MRF) to enhance compactness. However, it learns occupancy independently of the primitive-induced hypothesis, leading to inefficiencies in large-scale applications.

In this paper, we present PolyGNN, a polyhedron-based graph neural network for reconstructing building models from point clouds. PolyGNN leverages the decomposition of a building's ambient space into a set of polyhedra as strong priors. It learns to assemble the

* Corresponding author at: Chair of Data Science in Earth Observation, Technical University of Munich, 80333, Munich, Germany.
E-mail addresses: zhaiyu.chen@tum.de (Z. Chen), yilei.shi@tum.de (Y. Shi), liangliang.nan@tudelft.nl (L. Nan), zhitong.xiong@tum.de (Z. Xiong), xiaoxiang.zhu@tum.de (X.X. Zhu).

https://doi.org/10.1016/j.isprsjprs.2024.09.031
Received 7 September 2023; Received in revised form 30 June 2024; Accepted 23 September 2024
Available online 10 October 2024

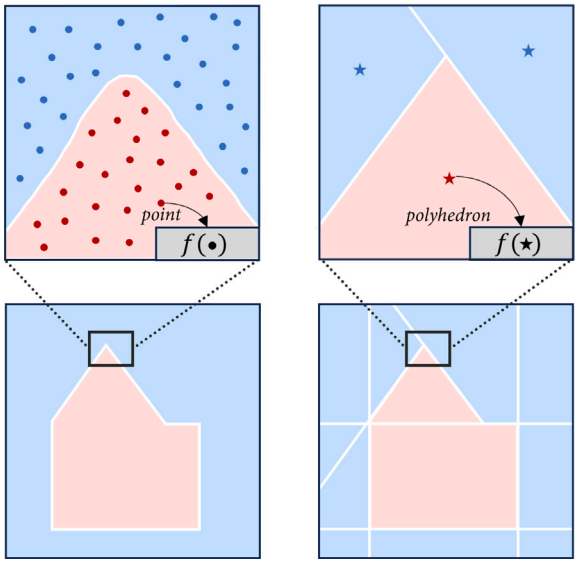

Fig. 1. Instead of learning a continuous function $f(\bullet)$ underpinned by exhaustive queries with traditional deep implicit fields, PolyGNN learns a piecewise planar occupancy function $f(\star)$ supported by polyhedral decomposition.

polyhedra to achieve a watertight and compact reconstruction, framed as end-to-end graph node classification. The neural network can be efficiently optimized, enabling reconstruction at scale.

Our key innovation entails integrating occupancy estimation with the polyhedral decomposition through primitive assembly. As illustrated in Fig. 1, instead of learning a continuous function with traditional deep implicit fields, we opt for learning a piecewise planar occupancy function from the decomposition. There, one challenge lies in consistently representing the heterogeneous geometry of arbitrary-shaped polyhedra. To address this, we propose an efficient skeleton-based strategy to sample a set of representative points within the polyhedron as queries. These queries, conditioned on the latent building shape code, collectively characterize the building occupancy. Moreover, PolyGNN is designed to accommodate variable-size input points, polyhedra, and queries using an index-driven batching technique.

We observe that existing 3D city models offer abstract representations of real-world buildings, often lacking geometric details. Thus, employing existing mesh models as ground truths is inherently inadequate due to systematic errors. To facilitate supervised learning for PolyGNN, we resort to creating a large-scale synthetic dataset comprised of simulated airborne LiDAR point clouds and corresponding building models. This dataset enables reliable one-to-one mapping between the two sources, effectively addressing the potential abstraction gap. Subsequently, we assess the transferability of our method across cities and on real-world data.

The main contributions of this paper are summarized as follows:

- We present PolyGNN, a polyhedron-based graph neural network for reconstructing compact polygonal building models from point clouds.
- We introduce a skeleton-based sampling strategy that efficiently represents arbitrary-shaped polyhedra within the neural network.
- We demonstrate the transferability of PolyGNN on cross-city synthetic point clouds and a real-world airborne LiDAR point cloud.

2. Related work

In this section, we discuss three categories of methods used for polygonal building model reconstruction: constrained reconstruction, geometric simplification, and primitive assembly. Subsequently, we introduce a line of relevant works in neural implicit representation, from which we draw inspiration for our work.

2.1. Constrained reconstruction

Constrained reconstruction methods reduce the complexity of reconstruction by incorporating constraints into the solution space, employing pre-defined templates or specific disk topologies.

With a library of roof model templates, Henn et al. (2013) applied estimation methods to derive best-fitting models from sparse LiDAR point clouds. The widely adopted Manhattan-world assumption restricts the orientation of building surfaces in three orthogonal directions, representing buildings with axis-aligned polycubes. Vanegas et al. (2012) provided a solution to reconstruct Manhattan-world building masses from LiDAR scans. Li et al. (2016a) and Li et al. (2016b) further extended the solution with integer programming and an MRF, respectively. Suveg and Vosselman (2004) proposed to integrate aerial image analysis with GIS footprints, and formulated reconstruction as a multilevel hypothesis generation and verification scheme.

Another common constraint is restricting output surfaces to specific disk topologies. Satellite data offers prospects for global building models which often lack the necessary level of detail (LoD), and thus are primarily limited to 3D models at LoD1 (Zhu et al., 2022; Sun et al., 2022). The 2.5D view-dependent representation can generate building roofs with vertical walls connecting them from LiDAR measurements (Zhou and Neumann, 2010). Peters et al. (2022) employed footprint partitioning and extrusion for an automated building reconstruction in both LoD1 and LoD2. Huang et al. (2022) proposed an LoD2 building reconstruction approach with integer programming. Furthermore, Xiong et al. (2014) and Xiong et al. (2015) exploited roof topology graphs for reconstructing LoD2 buildings from predefined primitives. Chen et al. (2017) proposed a three-stage method for LoD2 modeling that consists of primitive clustering, boundary representation, and geometric modeling. Kelly et al. (2017) proposed a data fusion technique for structured urban reconstruction from coarse meshes, street-view imagery, and GIS footprints.

The constrained reconstruction approaches simplify the reconstruction and are thus efficient to implement. However, they only apply to specific domains as the limited variety of templates or topologies restricts the expressiveness of these methods. Our reconstruction method, instead, does not rely on templates or specific topologies, thus remaining generic.

2.2. Geometric simplification

Another category of approaches generates compact 3D models through the simplification of existing dense triangle meshes typically obtained from photogrammetric surface reconstruction methods or learning-based alternatives. We refer to Berger et al. (2017) for the former, and Sulzer et al. (2023) for the latter.

Garland and Heckbert (1997) introduced a mesh simplification method based on iterative vertex contraction using quadratic error metrics. Salinas et al. (2015) further incorporated planar proxies detected from a pre-processing analysis to preserve piecewise planar structures. The variational shape approximation technique (Cohen-Steiner et al., 2004) optimizes a set of geometric proxies to construct an approximating polygonal mesh.

Several methods specifically focus on the simplification of urban scenes. Verdie et al. (2015) proposed an approach that incorporates various LoD configurations through classification, abstraction, and reconstruction. Bouzas et al. (2020) incorporated a structure graph to encode planar primitives and formulated geometric simplification as mesh polygonization. Li and Nan (2021) utilized planar regions to constrain edge collapse operations, achieving feature-preserving building mesh simplification. Gao et al. (2022) proposed a three-stage approach involving carving visual hulls to generate low-poly building meshes.

While geometric simplification methods hold promise for compact building modeling, they often require high-quality input meshes for faithful surface approximation. In practice, these methods may introduce uncertainty and additional burdens without guaranteeing accuracy. In contrast, our proposed method can directly generate a compact polygonal mesh without approximating an intermediate.

2.3. Primitive assembly

Primitive assembly methods produce compact polygonal surface models by optimizing the assembly of a set of geometric primitives. In practice, planar primitives can be extracted with RANSAC (Schnabel et al., 2007), region growing (Rabbani et al., 2006), or other optimization methods (Yu and Lafarge, 2022; Li et al., 2019, 2023). High-quality primitives are critical to primitive assembly methods.

Connectivity-based approaches (Chen and Chen, 2008; Van Kreveld et al., 2011; Schindler et al., 2011) address the assembly by extracting proper geometric primitives from an adjacency graph built on planar shapes. While efficient in analyzing the graph, these methods are sensitive to the quality of the graph. Linkage errors contaminating the connectivity can compromise the reconstruction. A hybrid strategy proposed by Lafarge and Alliez (2013) and Holzmann et al. (2018) represented high-confidence areas by polygons and more complex regions by dense triangles. Arikan et al. (2013) presented an interactive optimization-based snapping solution, which requires labor-intensive human involvement in handling complex structures.

Slicing-based approaches are more robust to imperfect data with the hypothesis-and-selection strategy. With the primitives, these approaches (Chauve et al., 2010; Mura et al., 2016; Nan and Wonka, 2017; Bauchet and Lafarge, 2020) partition the 3D space into polyhedral cells by extending the primitives to supporting planes, transforming the reconstruction into a labeling problem where the polyhedral cells are labeled as either inside or outside the shape or equivalently by labeling other primitives. Li and Wu (2021) extended PolyFit (Nan and Wonka, 2017) to leverage the inter-relation of the primitives for procedural modeling. Huang et al. (2022) further extended PolyFit by introducing a new energy term to encourage roof preferences and two additional hard constraints to ensure correct topology and enhance detail recovery. Fang and Lafarge (2020) proposed a hybrid approach for reconstructing 3D objects by successively connecting and slicing planes identified from 3D data. Further, Xie et al. (2021) proposed an approach combining rule-based and hypothesis-based strategies.

Our method inherits primitive assembly while realizing the selection by a graph neural network with the polyhedra from convex decomposition. The selection is therefore completely data-driven and requires no handcrafted features.

2.4. Implicit neural representation

Recent advances in deep implicit fields have revealed their potential for 3D surface reconstruction (Park et al., 2019; Peng et al., 2020; Erler et al., 2020), and also specifically for buildings (Stucker et al., 2022). The crux of these methods is to learn a continuous function to map the input, such as a point cloud, to a scalar field. The surface can then be extracted using iso-surfacing techniques like Marching Cubes (Lorensen and Cline, 1987). To learn a more regularized field, Rella et al. (2022) and Yang et al. (2023) both proposed to learn the displacements from queries towards the surface and model shapes as vector fields. However, these methods still require iso-surfacing to extract the final surfaces. Although iso-surfacing is effective in extracting smooth surfaces, it struggles to preserve sharp features, and introduces discretization errors. Consequently, deep implicit fields alone are unsuitable for reconstructing compact polygonal models. Notably, while Chen et al. (2020) and Deng et al. (2020) both employed implicit fields for reconstructing convex shapes obtained via binary space partitioning, transferring them for compact building reconstruction from point clouds remains elusive.

Points2Poly (Chen et al., 2022) is a pioneering learning-based effort for polygonal building reconstruction. The key enabler is a learned implicit function that indicates the occupancy of a building, followed by an MRF for a favorable geometric complexity. By its design, Points2Poly is composed of two separate parts, hence cannot be optimized end-to-end. The occupancy learning is agnostic of the hypothesis. This limits its exploitation of deep features, and in turn, limits its efficiency. The prohibitive complexity hinders its application at scale. In contrast, our method directly learns to classify the polyhedra with an end-to-end neural architecture, underpinning great efficiency.

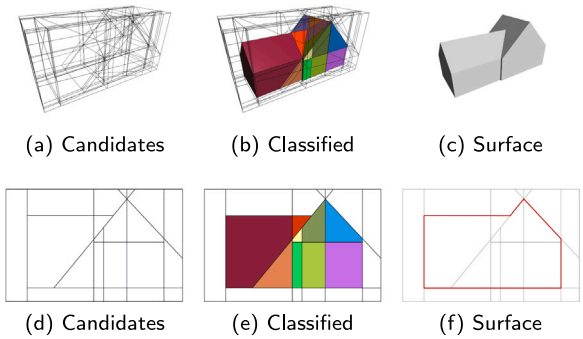

Fig. 2. Reconstruction by polyhedra classification. Candidate polyhedra (a) are generated by polyhedral decomposition and are classified by PolyGNN into *interior* ones and *exterior* ones (b). The surface (c) is extracted in between pairs of polyhedra of different classes. (d) (e) (f) are illustrations of 2D cross sections of (a) (b) (c), respectively.

3. Methodology

3.1. Overview

We formulate building reconstruction as a graph node classification problem. As illustrated in Fig. 2, we first decompose the ambient space of a building into a cell complex of candidate polyhedra following binary space partitioning. We then represent the cell complex as a graph structure and classify the polyhedral nodes into two classes: *interior* and *exterior*. Finally, the building surface model can be extracted as the boundary between the two classes of polyhedra.

The above procedure can be formulated as follows. Given an unordered point set $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ with $x_i \in \mathbb{R}^3$ as input, we first decompose the ambient space into an undirected graph embedding $\mathcal{G} = (\mathcal{V}, \mathcal{E} \mid \mathcal{X})$, where $\mathcal{V} = \{v_1, v_2, \dots, v_m\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represent non-overlapping convex polyhedra and their edges, respectively. $\mathcal{G}$ serves as a volumetric embedding, from which we seek an appropriate subset of $\mathcal{V}$ to align with the occupancy of the underlying building instance. The surface reconstruction is therefore transformed into an assignment problem which we address with a graph neural network $\tilde{f}$:

$$\tilde{f} \approx f(\mathcal{V} \mid \mathcal{X}, \mathcal{E}) = Y, \tag{1}$$

where $Y = \{y_1, y_2, \dots, y_m\} \subseteq \{0, 1\}$. Fig. 3 illustrates the architecture of PolyGNN for solving the graph node classification problem, which consists of two stages:

- **Polyhedral graph encoding**. A graph structure is constructed from polyhedral decomposition, with polyhedra being graph nodes. Node features are formed by conditioning polyhedron-wise queries on a shape latent code that encodes the occupancy of the building.
- **Polyhedral graph node classification**. With the encoded node features and inter-polyhedron adjacency, graph nodes (i.e., polyhedra) are classified for building occupancy estimation.

In the following, we elaborate on the two main components of the network.

3.2. Polyhedral graph encoding

3.2.1. Polyhedral graph construction

We adopt the adaptive binary space partitioning approach introduced by Chen et al. (2022). We first identify a set of planar primitives from the input point cloud that comprises the building, and subsequently partition the ambient 3D space to generate a linear cell complex of non-overlapping polyhedra that complies with the primitives. Given our focus on primitive assembly, we assume that high-quality planar

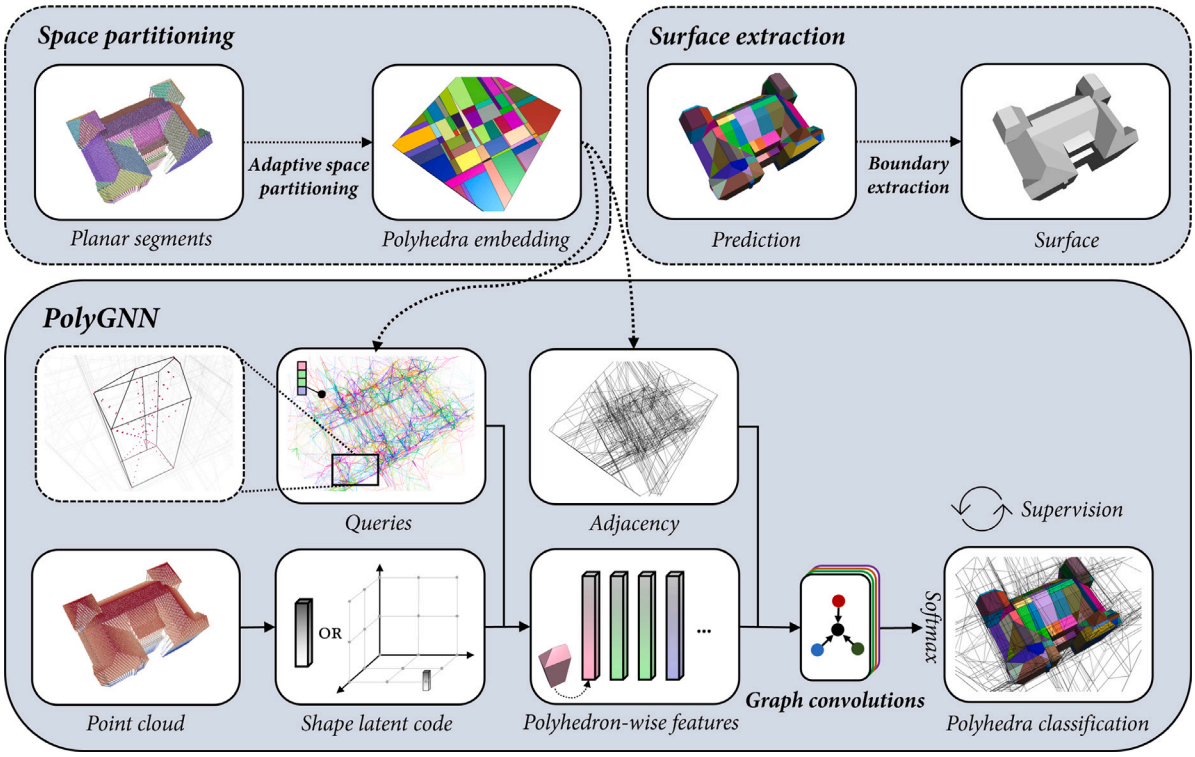

Fig. 3. Architecture of PolyGNN. Given an input point cloud, a graph topology is constructed from polyhedral decomposition, with polyhedra being graph nodes. Node features are formed by conditioning polyhedron-wise queries on a shape latent code. With the encoded node features and inter-polyhedron adjacency, graph nodes are classified for building occupancy estimation. Space partitioning and surface extraction are two external solvers.

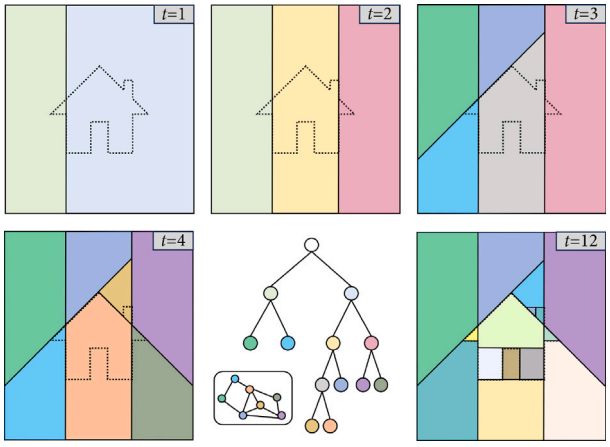

Fig. 4. Illustration of adaptive binary space partitioning. During partitioning, a binary tree is dynamically constructed to analyze inter-polyhedron adjacency. t denotes iteration.

primitives have been extracted. As illustrated in Fig. 4, vertical primitives and primitives with larger areas are given higher priority. The tessellation is spatially adaptive therefore being efficient and respective to the building's geometry. The partitioning also involves the construction of a binary tree, which records the hierarchical information of polyhedra and their inter-polyhedron adjacency.

3.2.2. Point cloud encoding

By adaptive binary space partitioning, the polyhedral embedding $\mathcal{G}$ is produced in the Euclidean space, from which we seek an appropriate subset of $\mathcal{V}$ to align with the occupancy of the underlying building instance. In addition, we obtain the shape latent code $\mathbf{z}$ embedded in the feature space, via a graph neural network.

We transform $\mathcal{X}$ into a neural feature representation, a process which, in principle, can be achieved through any point cloud encoder. We choose a lightweight plain encoder and a more modern convolutional encoder, and demonstrate the performance of PolyGNN with the two encoders.

With the plain encoder, point features are encoded with layers of dynamic edge convolutions (Wang et al., 2019):

$$g_{x_i}^{(1)} = \sum_{x_j \in \mathcal{N}(x_i)} h_\Theta \left(\left[h_{x_i}, h_{x_j}, e_{x_i,x_j} \right] \right), \tag{2}$$

where $g_{x_i}^{(1)}$ is the feature representation of point x_i. $\mathcal{N}(x_i)$ denotes the set of neighboring points of x_i in the K-nearest-neighbor graph constructed from the input point cloud. e_{x_i,x_j} is the edge feature between x_i and x_j. h_{x_i} and h_{x_j} are the features of points x_i and x_j, respectively. h_Θ is a multi-layer perceptron (MLP) that maps the concatenated input to a new feature space. Features from L layers are further concatenated and aggregated by a max pooling operator, followed by another MLP γ_Θ to form the global latent code denoted as $\mathbf{z}^{(1)}$:

$$\mathbf{z}^{(1)} = \gamma_\Theta \max_{i=1}^{n} \left(\left[g_{x_i}^{(1)1}, g_{x_i}^{(1)2}, \ldots, g_{x_i}^{(1)L} \right] \right). \tag{3}$$

Alternatively with the convolutional encoder (Peng et al., 2020), point features are first encoded with a shallow PointNet (Qi et al., 2017) with local max pooling:

$$g_{x_i}^{(2)} = \max_{x_j \in \mathcal{N}(x_i)} h_\Theta \left(h_{x_j}, x_j - x_i \right). \tag{4}$$

The point-wise features are then projected onto three feature planes and form the latent code $\mathbf{z}^{(2)}$:

$$\mathbf{z}^{(2)} = \mu_\Theta \left(\texttt{project}_u \left(g^{(2)} \right) \right), \tag{5}$$

where $u \in \{\mathrm{XY}, \mathrm{XZ}, \mathrm{YZ}\}$ represents the three orthogonal planes. μ_Θ denotes a U-Net with weights shared across the planes, and $\texttt{project}_u$ indicates projection onto plane u.

3.2.3. Query sampling

To encode an arbitrary-shaped polyhedron, one challenge lies in consistently describing the heterogeneous polyhedral geometry. To address this, we propose sampling representative points from inside the polyhedron and coercing the geometry into fixed-length queries $\mathbf{s}$ of size k: $\mathbf{s} = \{s_1, s_2, \ldots, s_k\}$. It is clear that the more representative the sampled points are, the more information they convey about the polyhedron. We propose skeleton sampling that picks samples from both vertices and principal axes, as described in Algorithm 3.1 and Fig. 5. Vertices, because of their prominence in describing sharp geometry, are prioritized over points along the axes when a low value of k is given.

Algorithm 3.1: SKELETON SAMPLING ($\mathcal{V}$, C, k)

Input: Vertices $\mathcal{V}$, centroid C, and #samples k
Output: Representative points $\mathbf{s}$

$\mathbf{s} \leftarrow$ init $\emptyset$;
if $k \leq |\mathcal{V}|$ **then**
  $\mathcal{V}_s \leftarrow \text{sample}_k(\mathcal{V})$;
  $\mathbf{s} \leftarrow \mathcal{V}_s$;
else
  $k_m \leftarrow \lfloor \frac{k}{|\mathcal{V}|} \rfloor$;
  $k_l \leftarrow k \bmod |\mathcal{V}|$;
  $\mathbf{s}_m \leftarrow \text{sample}_{k_m}\{(\mathcal{V}_i, C) : i = 1, 2, ..., |\mathcal{V}| - 1\}$;
  $\mathbf{s}_l \leftarrow \text{sample}_{k_l}(\mathcal{V}_{|\mathcal{V}|}, C)$;
  $\mathbf{s} \leftarrow \{\mathbf{s}_m, \mathbf{s}_l\}$;
return $\mathbf{s}$

We also evaluate two other sampling strategies, namely volume sampling and boundary sampling, as illustrated in Fig. 5. The volume variant randomly takes points inside the volume of a polyhedron, carrying relatively the least amount of geometric information about the polyhedron. The boundary variant samples points on the boundary of a polyhedron with area-induced probability. This variant can better depict polyhedral occupancy with boundary information. Since the skeleton variant picks samples from both vertices and principal axes, it provides arguably the most efficient description of a polyhedron among the three variants.

The representative points obtained by any of the three sampling strategies reduce the complexity of an arbitrary-shaped polyhedron to a fixed-size feature vector that can be consumed by the neural network while preserving geometric information to different extents. These representative points then serve as queries against $\mathbf{z}$, leading to the formation of polyhedron-wise features. Intuitively, these queries are used to jointly describe the occupancy of the underlying polyhedron. Note that each variant is applied individually, and their performances are compared in Section 5.1.

3.2.4. Forming polyhedron-wise features

Inspired by the recent advance in 3D shape representation learning (Park et al., 2019; Yao et al., 2021), for the latent code $\mathbf{z}^{(1)}$, we form a shape-conditioned implicit representation $\mathbf{z}_s^{(1)}$ of the polyhedron by concatenating the coordinates of the queries $\mathbf{s}$ with $\mathbf{z}^{(1)}$:

$$\mathbf{z}_{\mathbf{s}}^{(1)} = \xi_\Theta\left(\mathbf{s} \mid \mathbf{z}^{(1)}\right) = \xi_\Theta\left(\left[s_1, s_2, \ldots, s_k, \mathbf{z}^{(1)}\right]\right), \tag{6}$$

where ξ_Θ represents an MLP.

Alternatively, the conditioned implicit representation $\mathbf{z}_s^{(2)}$ of the latent code $\mathbf{z}^{(2)}$ is constructed by interpolating $\mathbf{z}^{(2)}$ at the coordinates of $\mathbf{s}$:

$$\mathbf{z}_{\mathbf{s}}^{(2)} = \xi_\Theta\left(\mathbf{s} \mid \mathbf{z}^{(2)}\right) = \xi_\Theta \sum_u \left(\texttt{interpolate}_{\mathbf{z}^{(2)}}(\mathbf{s})\right), \tag{7}$$

where `interpolate` denotes bilinear interpolation, and $\mathbf{s}$ represents the coordinates of queries. Both Eqs. (6) and (7) enable the modeling of multiple building instances with a single neural network. Fig. 6 illustrates the different formations of polyhedron-wise features with $\mathbf{z}_s^{(1)}$ and $\mathbf{z}_s^{(2)}$.

Intuitively, $\mathbf{z}_s$ represents a discrete occupancy function that, given a polyhedron, describes its occupancy conditioned on the underlying building instance. This representation can be interpreted as a spatial classifier for which the decision boundary is the surface of the building. Notably, instead of approximating a continuous implicit function by exhaustive enumeration, our discretized formulation takes geometric priors of individual polyhedra into account, which significantly reduces computational complexity and mitigates solution ambiguity. Fig. 1 illustrates this distinction.

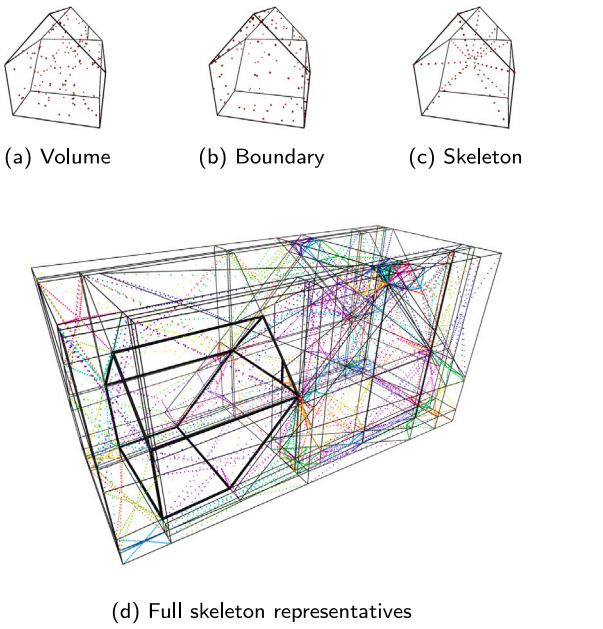

Fig. 5. Sampling representative points from a polyhedron. (a)–(c) visualize different strategies to sample the polyhedron highlighted in (d). **Volume**: Points are sampled randomly from inside the volume. **Boundary**: Points are sampled from the boundary. **Skeleton**: Points are sampled along the polyhedral skeleton as described in Algorithm 3.1. Representative points are color-coded by their parent polyhedra.

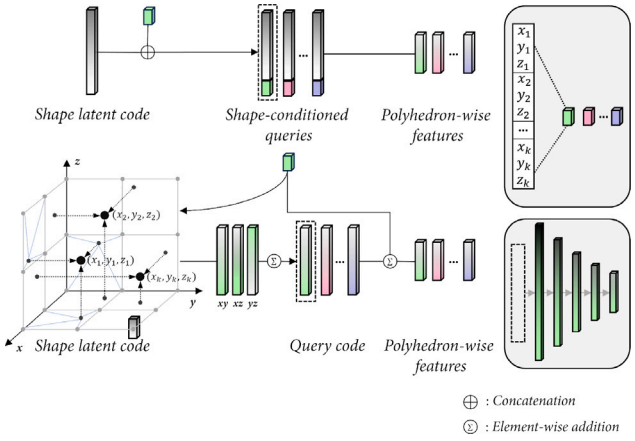

Fig. 6. Fusion of shape latent code and polyhedral queries to form polyhedron-wise features, with the plain encoder (top) and with the convolutional encoder (bottom).

3.3. Graph node classification

The polyhedron-wise features produced by Eq. (6) or Eq. (7) do not yet account for inter-polyhedron adjacency, which could provide

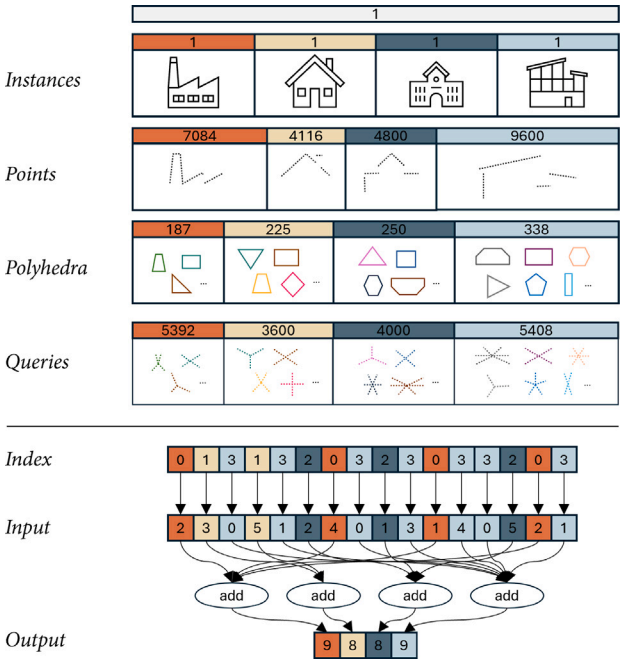

Fig. 7. PolyGNN efficiently accommodates variable-size input points, polyhedra, and queries with an index-driven batching technique (top). Batch size equals 4 in this example. An example with scattered `add` operator (bottom).

additional information for classifying individual polyhedra. To leverage this topological information and enhance classification performance, we utilize another stack of graph convolution layers for graph node classification, as outlined in Eq. (1). Specifically, we employ topology-adaptive graph convolution (Du et al., 2017) for its adaptivity to the topology of the graph and computational efficiency. It utilizes a set of fixed-size learnable filters for graph convolution, defined as follows:

$$\mathbf{G}^{l+1} = \sum_{k=0}^{K} \left(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^k \mathbf{G}^{l}, \tag{8}$$

where $\mathbf{G}^{l+1}$ and $\mathbf{G}^{l}$ denote the node features before and after the convolution at the l-th layer, respectively. $\mathbf{A}$ is the adjacency matrix implied by $\mathcal{E}$ in Eq. (1), and $\mathbf{D} = \mathrm{diag}[\mathbf{d}]$ with the i-th component being $d(i) = \sum_j \mathbf{A}_{i,j}$. K is the number of filters, whose topologies are adaptive to the topology of the graph.

Multiple graph convolution layers defined in Eq. (8) are stacked to increase the receptive field of the neural network. The feature of the i-th node G_i is then fed into a binary classification head with the softmax activation function to produce the probability of the polyhedron v_i being *interior*:

$$\hat{y}_i = \mathrm{softmax}\left(\lambda_\Theta\left(G_i\right)\right), \tag{9}$$

where λ_Θ denotes an MLP.

The number of input points, polyhedra, and queries may vary considerably among different building instances, naturally impeding parallelization with mini-batches. To mitigate this variability, we employ a batching mechanism tailored to accommodate such diversity. Our approach entails recording indices of the points, polyhedra, and queries within each batch. These indices are subsequently utilized to acquire instance-level features, as depicted in Fig. 7.

The network can be supervised by cross entropy or focal loss (Lin et al., 2017) that minimizes the discrepancy between the prediction and the ground truth. It can be optimized end-to-end without any auxiliary supervision. In the testing phase, given a building instance, we predict the occupancy of the candidate polyhedra. Then the surface lies in between pairs of polyhedra $\{v_i, v_j\}$ with different class predictions, i.e., $y_i \neq y_j$, as shown in Fig. 2.

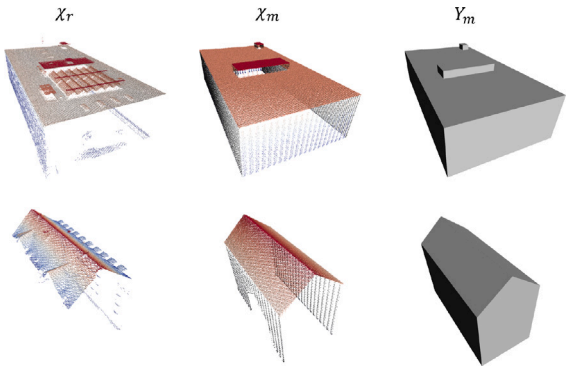

Fig. 8. Examples of abstraction gaps between real-world point clouds $\mathcal{X}_r$ and existing building models Y_m. Instead of learning f: $\mathcal{X}_r \to Y_m$, we learn an auxiliary mapping f': $\mathcal{X}_m \to Y_m$, where $\mathcal{X}_m$ is derived from Y_m by synthesizing $\mathcal{X}_r$. Point clouds are color-coded by their height fields.

4. Experimental settings

4.1. Datasets

Unlike other applications that adhere to rigorous definitions of ground truths, reconstructing large-scale polygonal buildings presents a challenge due to the abstraction of existing building models (Wang et al., 2023; Wichmann et al., 2018), resulting in inevitable deviations from actual measurements, as shown in Fig. 8. This abstraction would impede a supervised learning algorithm due to its inherent biases. To overcome this abstraction gap, we created a synthetic dataset comprised of simulated airborne LiDAR point clouds and their corresponding building mesh models. This dataset enables a reliable mapping between the two sources, enabling a fair evaluation of the proposed method for primitive assembly.

Formally, let $\mathcal{X}_r$ and Y_m be a real-world point cloud and its corresponding building model, respectively. Due to the abstraction gap, the mapping f: $\mathcal{X}_r \to Y_m$ cannot be accurately learned by a neural network. Instead, we opt to learn an auxiliary mapping f': $\mathcal{X}_m \to Y_m$ where $\mathcal{X}_m$ is derived from Y_m by synthesizing $\mathcal{X}_r$. Once f' is learned, it can be applied to $\mathcal{X}_r$ to obtain the corresponding output $Y'_r = f'(\mathcal{X}_r)$. Conditioned on good transferability to real-world point clouds, using synthetic data in our task offers two-fold advantages. First, it enables the learning of the desired mapping by circumventing the abstraction, allowing the classifier to be trained and evaluated independently of potential data discrepancies. Moreover, it facilitates the exploration of a large volume of "free" training data, which benefits the learning algorithm in general.

We utilize the *Helios++* simulation toolkit (Winiwarter et al., 2022) to simulate airborne LiDAR scanning. LoD2 building models from Bavaria, Germany are used as references for their high quality and coverage (State of Bavaria, 2022). Artifacts such as noise and inter-building occlusions are intentionally included in the scanning process, to assimilate the distribution of $\mathcal{X}_m$ and $\mathcal{X}_r$, thereby enhancing the robustness of the neural network against real-world measurements. The virtual sensor closely emulates the characteristics of *Leica HYPERION2+*, utilizing an oscillating optics system with a pulse frequency of 1.5 MHz and a scan frequency of 150 Hz. We simulate an airborne survey performed by a *Cirrus SR22* aircraft flying at an altitude of 400 m with a strip interval of 160 m. Our training dataset comprises 281,571 buildings with a total of 6,532,880,764 points extracted from the city of Munich, Germany, with an additional 10,000 buildings reserved for evaluation. On average, each building in the dataset is associated with 22,406 points. To assess the cross-city transferability of PolyGNN, we also synthesize data from 220,127 buildings in Nuremberg, Germany. In addition to the synthetic data, we apply the trained model directly to a real-world airborne

LiDAR point cloud dataset containing 1452 buildings captured with *Leica HYPERION2+*. For an individual building, we normalize it to unit scale, generate a set of polyhedra with inter-polyhedron adjacency as pre-processing (see Section 3.2.1), and use ray tracing to determine the ground truth occupancy label for every polyhedron.

4.2. Evaluation metrics

We utilize multiple criteria to evaluate the performance of the reconstruction. The classification accuracy directly impacts the fidelity of the reconstruction and is therefore evaluated. Furthermore, since the ground truths are reliably defined in our setting, we quantify the surface discrepancy between the reconstructed surface and the ground truth by calculating the Hausdorff distance H:

$$H = \max\left\{\sup_{a\in A}\inf_{b\in B} d(a,b), \sup_{b\in B}\inf_{a\in A} d(a,b)\right\}, \tag{10}$$

where $d(a,b)$ represents the distance between points a and b. We randomly sample 10,000 points from both the reconstructed surface and the ground truth and calculate both the absolute and relative distances. We quantify the success rate S by the proportion of 10,000 samples that are solvable. Typical unsolvable cases include empty reconstruction due to the absence of *interior* polyhedra, or timeout. For a fair comparison in the context of large-scale reconstruction, in the event of an unsolvable reconstruction, we assign the length of the largest side of the bounding box as the absolute distance, and 100% as the relative distance. We also employ RMSE for evaluating fidelity of the reconstruction on real-world data. Additionally, we measure the geometric complexity of the reconstructed building models in terms of the number of faces they comprise, denoted as N_F, and measure computational efficiency in terms of running time t with a 5 min timeout for an individual building.

4.3. Implementation details

We implemented adaptive space partitioning with robust Boolean spatial operations from *SageMath* (The Sage Developers, 2021). For query sampling, while a larger value of k, representing the number of samples per polyhedron, could enhance the polyhedral representation, particularly for polyhedra with a large number of faces, this increased representativeness would come with additional computational costs. For all of our experiments, we set k to 16 to balance the representativeness and computational complexity.

Although our implementation with the index-driven batching accommodates variable-length input point clouds, unless otherwise specified, the input point clouds are downsampled to 4096 points. Point clouds are normalized before being fed into the network and rescaled for computing the Hausdorff distance. All experiments are optimized by the Adam with a base learning rate 10^{-3} and weight decay 10^{-6}, with batch size 64. The network variants were trained for 50 epochs for the ablation experiments, whereas they continued training until 150 epochs for the best model in other experiments.

5. Results and analysis

5.1. Alternative and ablation experiments

As shown in Table 1, among the three sampling strategies presented in Fig. 5, skeleton sampling achieves the best classification and geometric accuracy, followed by boundary sampling. This finding aligns with the fact that both skeleton sampling and boundary sampling leverage more explicit geometric information compared to the volume counterpart, with the skeleton of a polyhedron capturing the most critical information conveyed by its vertices and principal axes.

The individual contributions of the classification head and the adjacency information to the reconstruction performance are analyzed through another ablation experiment, as presented in Table 2. Replacing the classification head with a regression head and the use of the L_2 loss cause the network to collapse completely, leading to the prediction of every polyhedron as an exterior one. In this case, the classification accuracy represents the high dominance of exterior polyhedra (87.3%). Furthermore, the results provide clear evidence that incorporating inter-polyhedron adjacency information significantly enhances the reconstruction performance compared to relying solely on monotonic polyhedral information (95.5% vs. 93.7%). This improvement suggests that PolyGNN effectively exploits neighborhood information for occupancy estimation. Additionally, Fig. 9 visually demonstrates the effectiveness of such information where the network utilizes adjacency information to achieve a more regularized reconstruction. This regularization is analogous to the MRF employed in Chen et al. (2022), while with PolyGNN it is integrated into the feature space, avoiding additional computational overhead.

PolyGNN is designed to be agnostic to the number of points, allowing for point clouds with varying sizes as inputs with advanced mini-batching. In Table 3, we compare three point cloud sampling options: random sampling, coarse grid sampling with a resolution of 0.05 within a unit cube, and fine grid sampling with a resolution of 0.01 within the same unit cube. The results demonstrate that random sampling outperforms grid sampling with both resolutions in terms of accuracy. It is worth noting that our random sampling strategy is dynamic, where different random points are selected in different epochs during training. This dynamic random sampling can also be considered a form of data augmentation. Although computationally more efficient, coarse grid sampling does not entail sufficient details for input point clouds. Interestingly, fine grid sampling leads to significantly longer training times, yet it yields lower accuracy compared to random sampling, possibly due to the inherent difficulty of encoding shape latent codes from variable-length inputs.

Table 4 compares the performance of the two encoders, demonstrating the superiority of the convolutional encoder over the plain

Table 1
Impact of query sampling strategy on model performance.

Query sampling	Accuracy (%) ↑	H (m) ↓
Random	94.5	1.20
Boundary	94.7	1.12
Skeleton	**95.5**	**1.08**

Table 2
Impact of classification head and adjacency information on model performance. "–" indicates complete failure where no model is reconstructed.

Classification	Adjacency	Accuracy (%) ↑	H (m) ↓
✗	✗	87.3	–
✓	✗	93.7	1.80
✓	✓	**95.5**	**1.08**

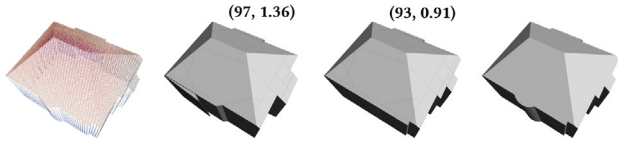

Fig. 9. Impact of adjacency in PolyGNN reconstruction. From left to right: input point cloud color-coded by height field, reconstructed model w/o adjacency, reconstructed model w/ adjacency, ground truth. $(\bullet,\bullet)$ denotes (N_F, H).

Table 3
Impact of point cloud sampling strategy on model performance and per-epoch training time. "res" represents grid resolution relative to a unit cube.

Point sampling	Accuracy (%) ↑	Time *train* (h.) ↓
Grid (res. 0.05)	94.6	**0.7**
Grid (res. 0.01)	94.8	24.6
Random	**95.5**	2.1

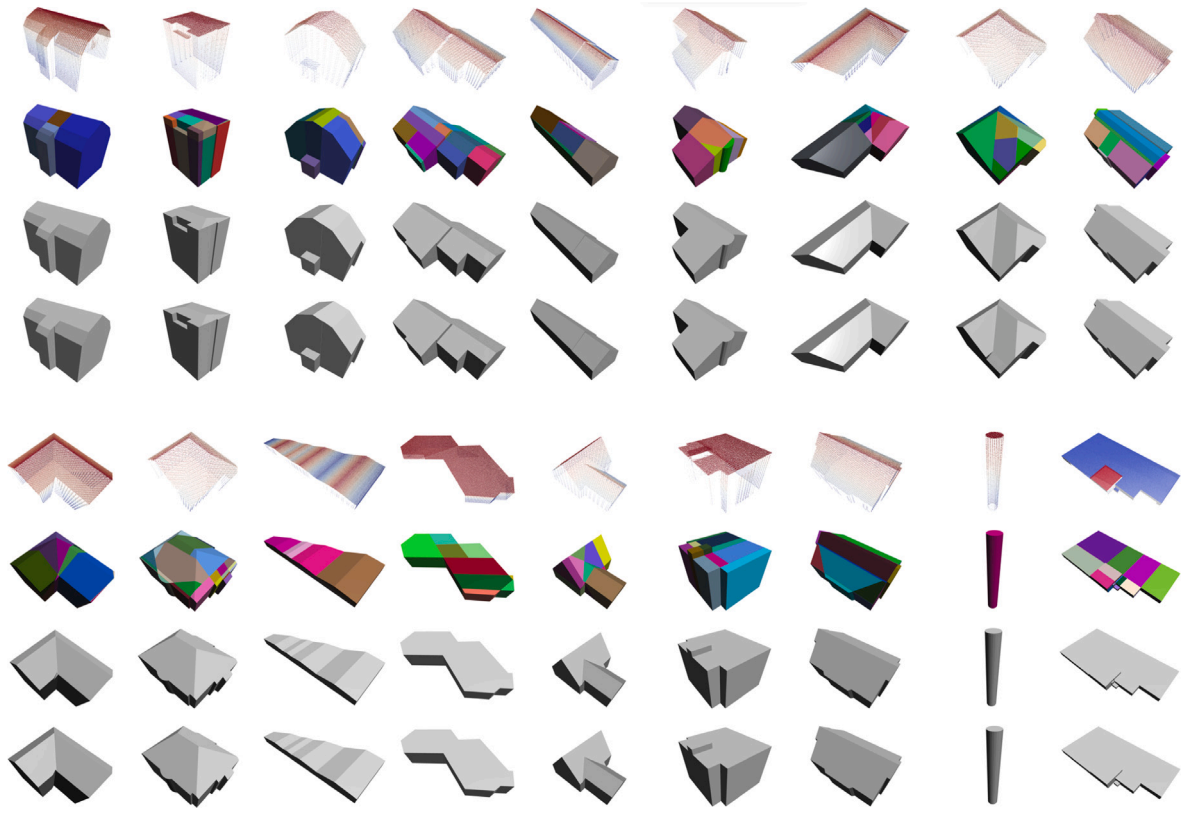

Fig. 10. Reconstruction examples on the Munich data with PolyGNN. From top to bottom: input point cloud color-coded by height field, polyhedra classified as building components, reconstructed model, and ground truth model. Point clouds are rendered by their height fields. Polyhedra are randomly color-coded.

Table 4
Impact of encoder on model performance and per-epoch training time.

Encoder	Accuracy (%) ↑	Time *train* (h.) ↓
Plain	95.5	2.1
Convolutional	**96.5**	**0.4**

encoder in terms of both accuracy and efficiency. This advantage reveals that the latent code generated by Eq. (5) preserves more local information compared to the shape latent code described in Eq. (3). Unless otherwise stated, the subsequent experimental analysis utilized the convolutional encoder.

5.2. Performance and transferability

PolyGNN achieves an average error of 0.33 m on the held-out Munich evaluation set. Notably, we observe that when building instances demonstrate similar levels of geometric complexity, accurate classification often leads to lower geometric errors. The reconstructed building models, as shown in Fig. 10, demonstrate conformity to the distribution of the point clouds while maintaining compactness for potential downstream applications. Buildings with simpler geometry are of more regularity in the reconstruction.

To assess the transferability of PolyGNN, we applied the model trained on the Munich data to buildings in Nuremberg. Fig. 11 showcases the reconstruction of a downtown area of Nuremberg. The Hausdorff distance measures 0.40 m. The comparable accuracy demonstrates the strong cross-city transferability of our approach when confronted with buildings that may vary in architectural styles. The inference with the convolutional encoder takes 15 s for 4185 buildings in the area, highlighting its efficiency for large-scale reconstruction.

Fig. 12 depicts the reconstruction results obtained by further applying PolyGNN trained exclusively on the synthetic data to real-world point clouds in Munich. Fig. 13 shows detailed examples. As expected, a domain gap exists between the two datasets, resulting in suboptimal reconstructions for certain buildings, especially those with architectural styles that are less represented in the training data. Nevertheless, it is noteworthy that the majority of the reconstructed buildings align well with the distribution of the input point clouds. Fig. 14 further demonstrates cases where we apply the trained model with extracted planar primitives by RANSAC (Schnabel et al., 2007). By learning the underlying mapping, the reconstruction may approximate the point cloud distribution closer than the ground truth does, which validates the effectiveness of our strategy of learning the auxiliary mapping.

5.3. Comparison with state-of-the-art methods

Table 5 presents a quantitative comparison between our method and state-of-the-art methods in urban reconstruction, while Fig. 15 showcases examples for qualitative comparison as well. The 2.5D DC method (Zhou and Neumann, 2010) outputs only facades and roofs, and therefore cannot be fairly compared to other reconstructions quantitatively by Hausdorff distance. Nevertheless, it was unable to represent building models with a concise set of parameters. In contrast, all the other methods exhibit compact reconstructions. Compared to the traditional optimization-based approach City3D (Huang et al., 2022), our method demonstrates the capability to handle more complex buildings commonly found in large-scale urban scenes. City3D adopts exhaustive partitioning, leading to a large solution space for exploration. Thus, it only managed to reconstruct 9886 buildings within a 5 min timeout using its Gurobi solver (Gurobi Optimization, LLC, 2023), resulting in inferior reconstruction accuracy as measured by the balanced Hausdorff distance. In contrast, our approach utilizes adaptive space partitioning, resulting in a more compact candidate space that enhances both efficiency and overall accuracy. Compared to Geoflow (Peters et al., 2022), which explicitly confines reconstruction within the

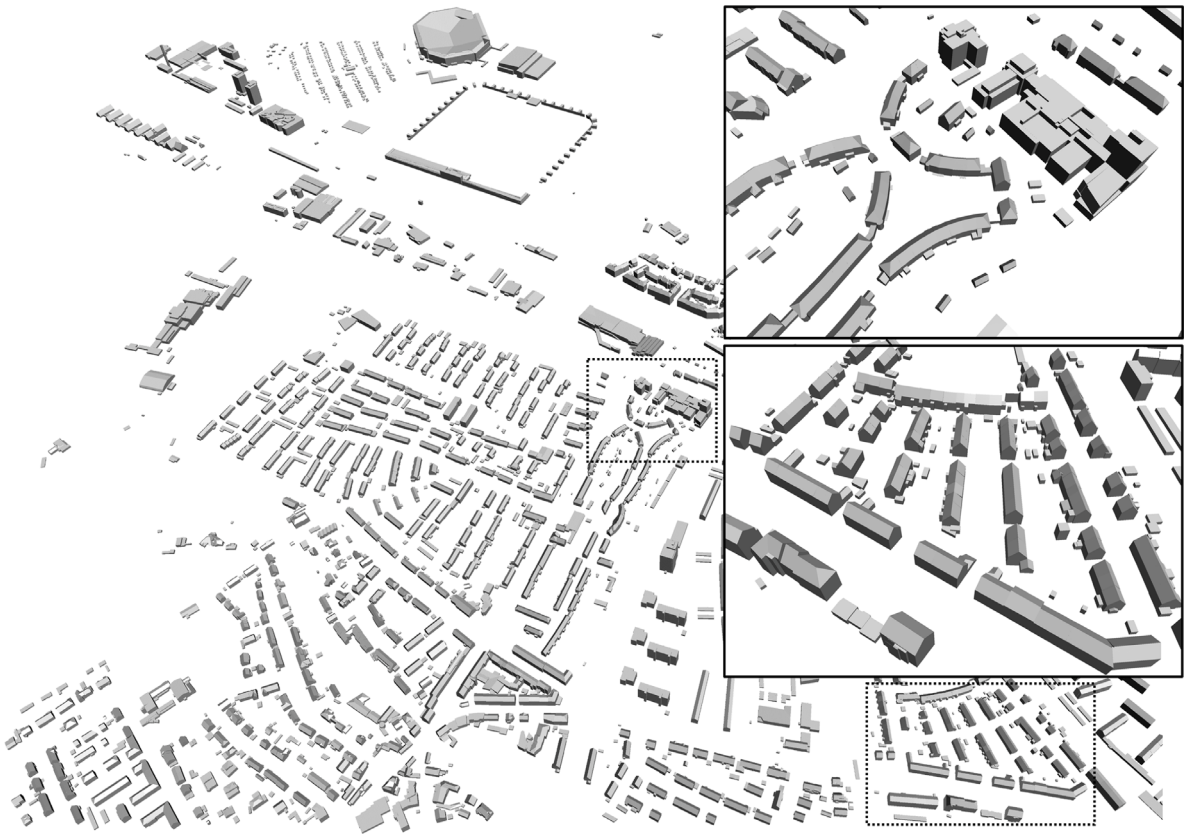

Fig. 11. Reconstruction of Nuremberg downtown buildings with PolyGNN.

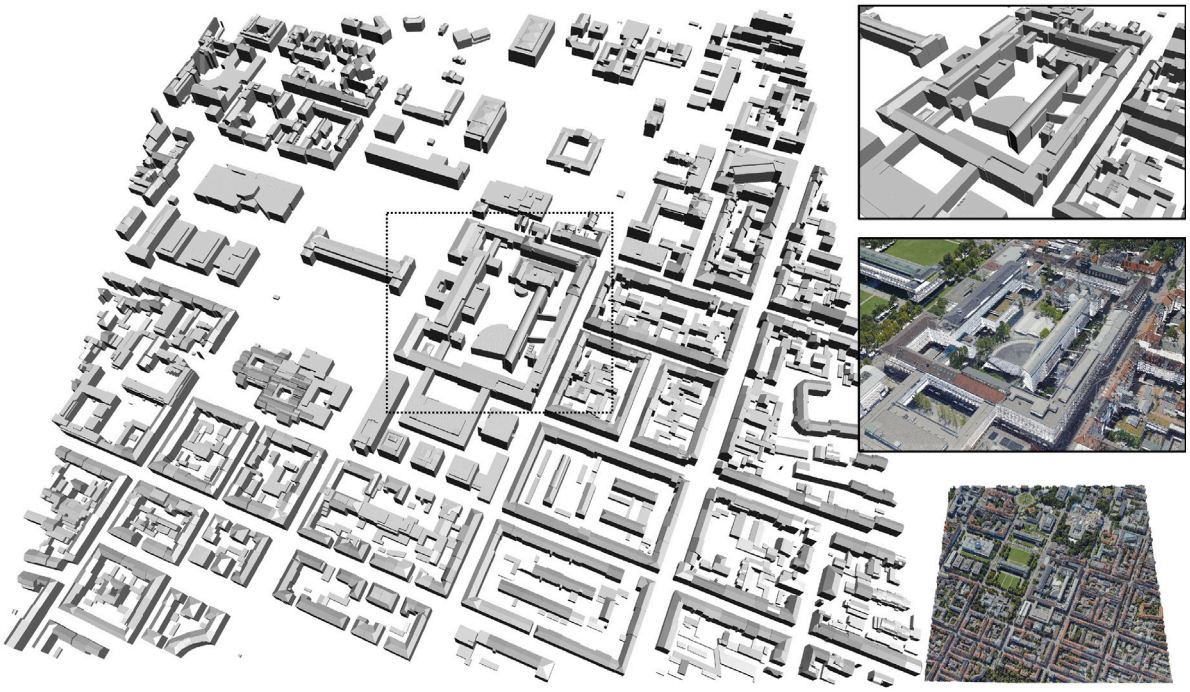

Fig. 12. Reconstruction from real-world point clouds with PolyGNN trained on the synthetic dataset.

footprint, our method demonstrates superior performance in terms of accuracy and compactness, even though it is not restricted to the 2.5D disk topology. Furthermore, in comparison to the learning-based method Points2Poly (Chen et al., 2022), PolyGNN excels in efficiency while achieving comparable or higher geometric accuracy. Fig. 13 also demonstrates that PolyGNN transfers better than Points2Poly from synthetic to real data, with lower RMSE.

Both Figs. 15 and 16 present comparisons with KSR (Bauchet and Lafarge, 2020), another primitive assembly method. When provided with identical input primitives, KSR necessitates additional wall points

Table 5
Quantitative performance comparison with state-of-the-art methods. "Learning" indicates learning-based methods. S, F_N, and H denote success rate, number of faces, and Hausdorff distance, respectively. Statistics were derived from 10,000 held-out samples, while F_N was derived from successfully reconstructed samples.

Method	Learning	S (%) ↑	F_N ↓	H (m) ↓	H (%) ↓
2.5D DC (Zhou and Neumann, 2010)	✗	**100.00**	613.63	–	–
City3D (Huang et al., 2022)	✗	98.86	48.87	1.10	6.0
Geoflow (Peters et al., 2022)	✗	99.97	137.66	0.42	2.8
Points2Poly (Chen et al., 2022)	✓	99.76	**27.94**	0.83	4.7
PolyGNN (ours w/ plain)	✓	99.76	36.37	0.81	4.7
PolyGNN (ours w/ conv.)	✓	**100.00**	28.82	**0.33**	**2.2**

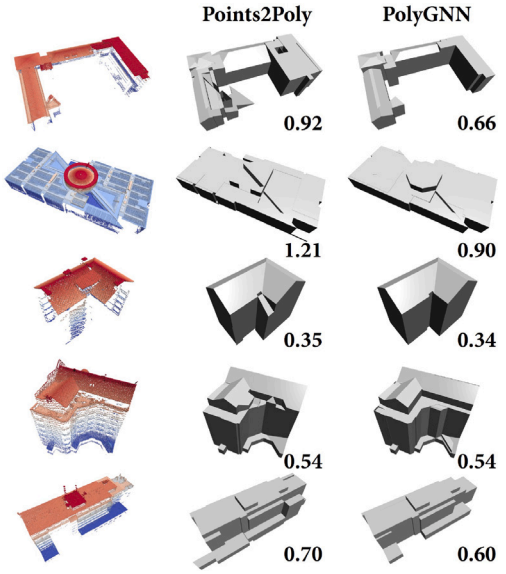

Fig. 13. Reconstruction from real-world point clouds. Numbers represent RMSE. Both Points2Poly (Chen et al., 2022) and PolyGNN (ours) were trained only on the synthetic data. The models produced by PolyGNN demonstrate lower RMSE.

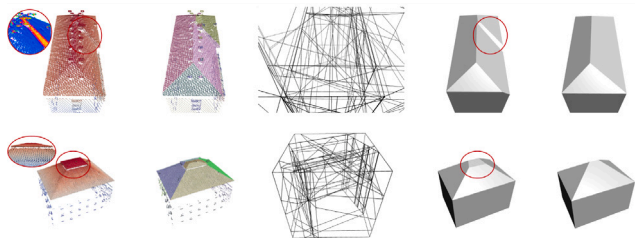

Fig. 14. Reconstruction from real-world point clouds with planar primitives extracted by RANSAC. From left to right: input point cloud color-coded by height field, the same input color-coded by primitives, cell complex, reconstructed model, and the ground truth. As revealed by the close-up views on the input point cloud, the reconstruction recovered more detailed structures than the ground truth.

for achieving non-trivial results, along with pre-computed point normals. In contrast, PolyGNN can directly estimate building occupancy from unorganized and incomplete point clouds. Furthermore, Fig. 17 shows a comparison with LowPolyBuildings (Gao et al., 2022), a simplification approach operating on dense smooth surfaces. Notably, PolyGNN's reconstruction exhibits greater regularity with the same level of complexity. Additionally, Fig. 18 showcases a comparison with Manhattan reconstruction (Li et al., 2016b), a model-based approach that utilizes polycubes as the model library. Here, PolyGNN's reconstruction demonstrates superior flexibility in describing arbitrarily oriented building geometry.

Fig. 19 presents the running time comparison among different methods, highlighting the superior efficiency of our approach. City3D, which relies on an integer programming solver, encounters computational bottlenecks as the number of planar primitives increases. Consequently, for certain complex buildings, the reconstruction cannot even be solved within a feasible time frame of 24 h. Points2Poly requires approximately 4 days for one epoch of training, whereas PolyGNN only takes 24 min (240× faster). The longer inference time of Points2Poly, on the other hand, comes mostly from two factors. Firstly, more efforts are required for its occupancy estimation. Table 6 presents the comparison with the learning-based method Points2Poly, for reconstructing the building in Fig. 3 with 60 planar segments. Points2Poly enumerates queries with signed distance values to learn a smooth boundary, while ours only requires discrete binary-class queries directly describing the piecewise planar surface. Meanwhile, the adaptive strategy significantly reduces the number of queries for both training and testing of our method as fewer polyhedra need to be considered as candidates. Secondly, the interface computation, which is necessary for assigning graph edge weights, contributes to the longer running time of Points2Poly. In contrast, PolyGNN can reconstruct a building directly by inferring the polyhedral occupancy, leveraging GPU parallelization for improved efficiency.

5.4. Robustness analysis

To analyze the robustness of PolyGNN against variations in point cloud density, we randomly drop input points. As shown in Fig. 20, even though PolyGNN was not explicitly trained against various point densities, it manages to achieve reasonable reconstruction with sparsely subsampled point clouds. When uniformly dropping 85% of the points, the overall shape can still be maintained as long as planar primitives remain accurate. Additionally, while preserving the decomposition, we truncated different portions of input points to analyze the robustness of PolyGNN against heterogeneously missing points, a scenario for which it was also not explicitly trained. The reconstruction remains feasible even when up to 30% of the points are truncated, as shown in Fig. 20.

We also apply different levels of noise to perturb the initial planes. As shown in Fig. 21, the reconstruction starts to deviate from the ground truth when the noise level reaches 5%. Increasing the noise level results in irregular cell complexes and thus poses challenges for primitive assembly. With 20% noise, the method fails to reconstruct the main body. Since PolyGNN learns a discrete decision boundary, it is inherently sensitive to the quality of planar primitives. We believe that introducing different point distributions and noises into the training data would further enhance the model's robustness.

5.5. Limitations

We assume the availability of high-quality planar primitives extracted from point clouds. This assumption may not always be fulfilled with real-world data that contains significant noise and occlusions, and therefore is considered a limitation of the proposed method and similar approaches that rely on primitive assembly. Additionally, since PolyGNN operates on individual buildings, a preliminary step of building instance segmentation is required prior to reconstruction, especially when dealing with point clouds of entire scenes.

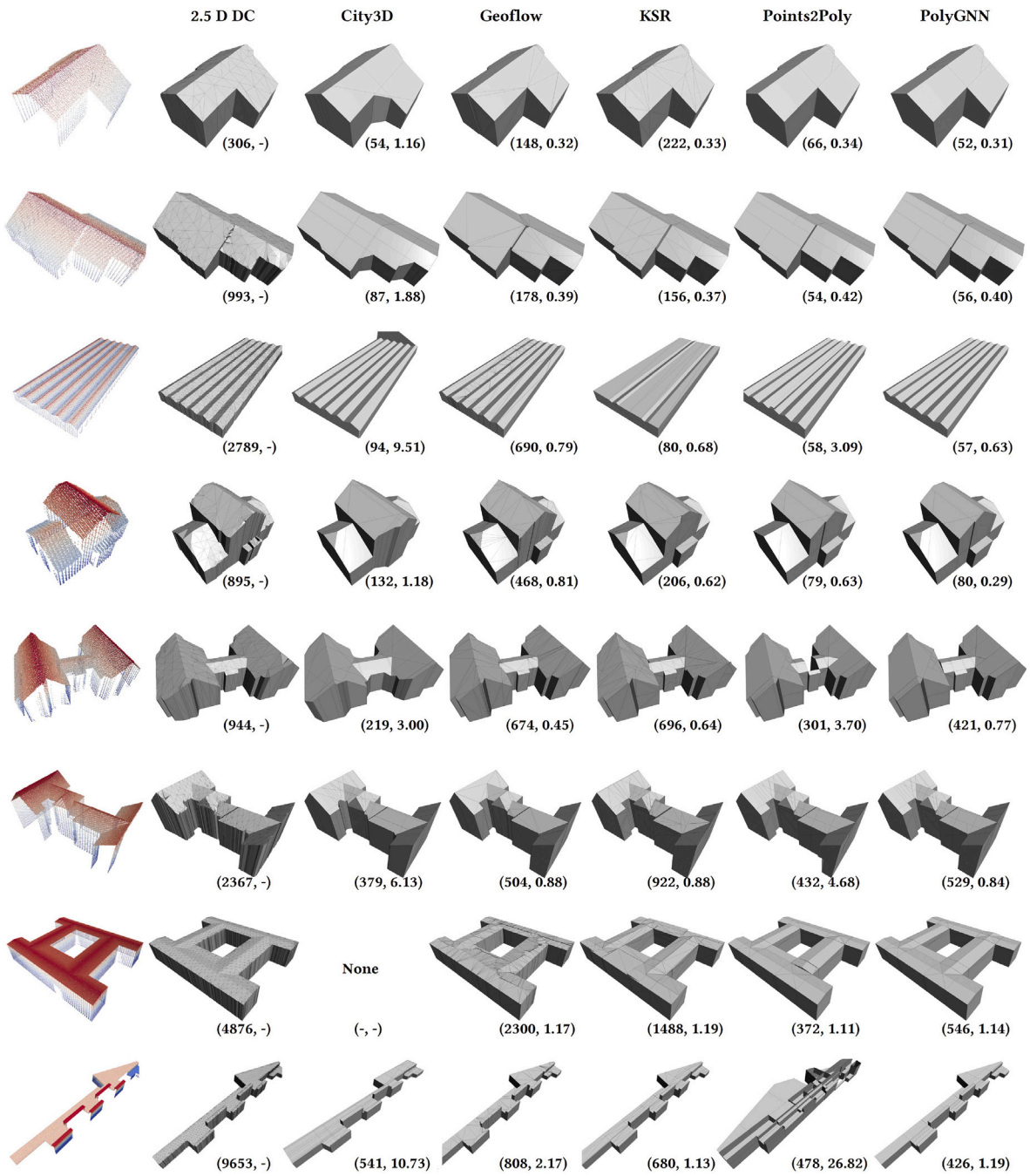

Fig. 15. Qualitative performance comparison with state-of-the-art methods: 2.5D DC (Zhou and Neumann, 2010), City3D (Huang et al., 2022), Geoflow (Peters et al., 2022), KSR (Bauchet and Lafarge, 2020), Points2Poly (Chen et al., 2022), and PolyGNN (ours). Point clouds are color-coded by their height fields. $(\bullet,\bullet)$ denotes (N_F, H). Note that additional wall points were required by KSR; otherwise, it would generates trivial results (see Fig. 16).

Table 6
Efficiency comparison between Points2Poly (Chen et al., 2022) and ours, two learning-based methods. The building in Fig. 3 with 60 planar segments is taken for calculating the number of queries. Efficiency is a derived factor based on the number of queries; the actual gain may deviate due to parallelization.

Method	Label type	#Queries *train*	#Queries *test*	Efficiency *train*	Efficiency *test*
Points2Poly w/ exh.	Class + value	2,600,000	14,146,600	1x	1x
Points2Poly w/ ada.	Class + value	2,600,000	1,127,100	1x	13x
PolyGNN (ours) w/ exh.	Class	174,112	174,112	15x	81x
PolyGNN (ours) w/ ada.	Class	13,872	13,872	**187x**	**1020x**

Fig. 22 shows some failure cases. When reconstructing buildings with complex structures, PolyGNN may encounter challenges in capturing fine details, such as intricate rooftop superstructures. These failures can be attributed to two factors. Firstly, the complexity of a building implies a larger and more intricate polyhedral embedding, which poses challenges to the network's prediction. Secondly, the training dataset

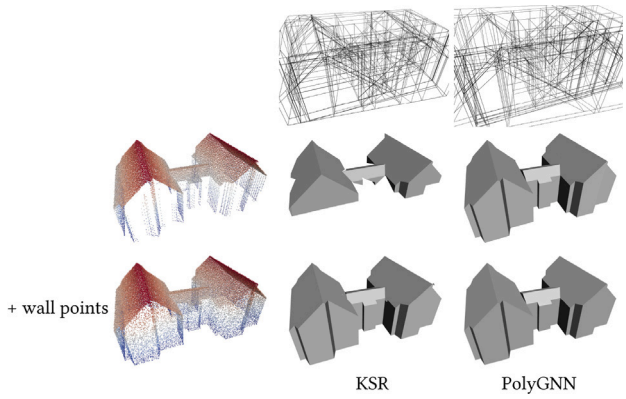

Fig. 16. Comparison with KSR (Bauchet and Lafarge, 2020). Additional wall points were required by KSR for non-trivial results, in addition to point normals. Given the same planar primitives, PolyGNN (ours) can estimate building occupancy from unorganized incomplete points, and can still deal with added wall points even though it was only trained on airborne data without wall points. Point clouds are color-coded by their height fields.

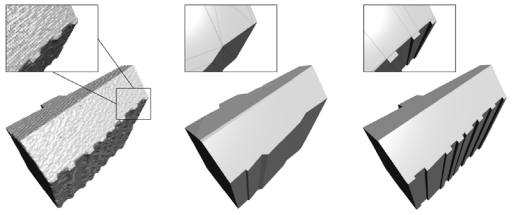

Fig. 17. Comparison with geometric simplification. Left: smooth surface mesh with Screened Poisson (Kazhdan and Hoppe, 2013) (N_F = 40,042). Middle: simplified mesh with LowPolyBuildings (Gao et al., 2022) (N_F = 187). Right: direct reconstruction with PolyGNN (ours) (N_F = 190). Our results demonstrate greater regularity.

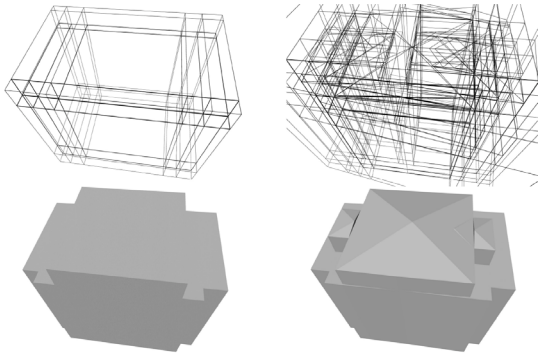

Fig. 18. Comparison between Manhattan reconstruction (Li et al., 2016b) and PolyGNN. Left: Manhattan reconstruction with polycubes as candidates. Right: PolyGNN (ours) with arbitrary polyhedra as candidates.

predominantly consists of buildings with simple shapes, leading to an under-representation of complex structures. As a result, the network may have limited exposure to and understanding of complex architectural elements with fine details.

6. Conclusion

We introduced PolyGNN, a novel framework for urban building reconstruction with a polyhedron-based graph neural network. Unlike traditional deep implicit fields that learn a continuous function, our approach learns a piecewise planar occupancy function derived from polyhedral decomposition. We proposed a skeleton-based sampling strategy for representing an arbitrary-shaped polyhedron within the

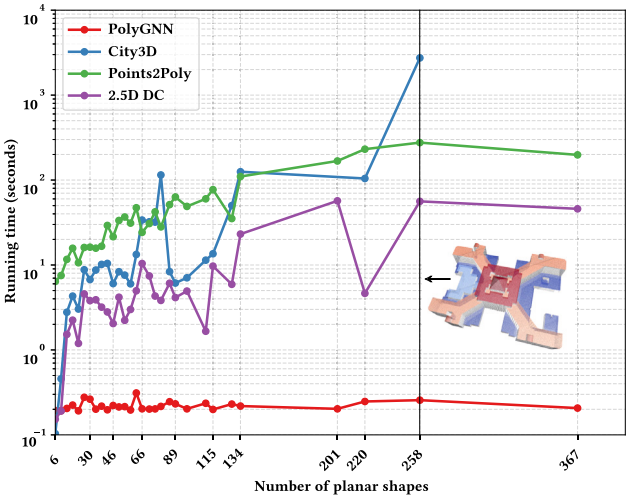

Fig. 19. Running time comparison. Statistics are derived from 30 buildings of various geometric complexity with the number of planar primitives ranging from 6 to 367.

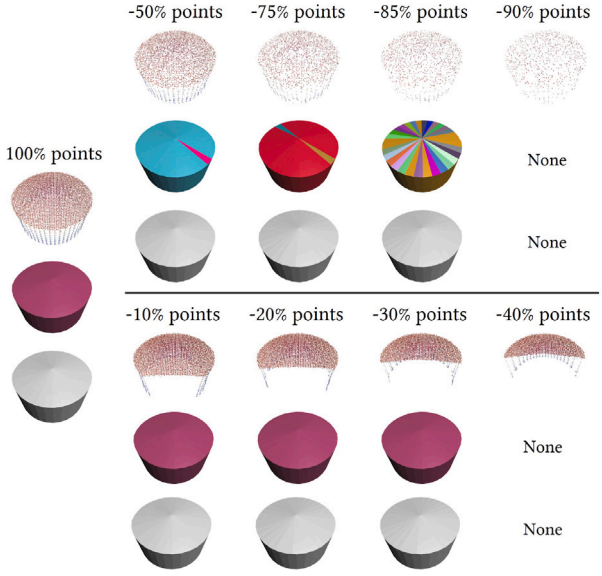

Fig. 20. Robustness to point cloud distribution with various point densities (top) and levels of missing points (bottom). Each triplet from top to bottom: input point cloud, polyhedra classified as building components, and reconstructed model. Polyhedra are randomly color-coded.

neural network, and demonstrated its superior performance compared to other variants. Furthermore, PolyGNN is end-to-end optimizable and is designed to accommodate variable-size input points, polyhedra, and queries with an index-driven batching technique.

We developed PolyGNN on a large-scale synthetic building dataset furnished with polyhedral labels and analyzed its transferability on cross-city synthetic data and real-world data. Both qualitative and quantitative results demonstrate the effectiveness of PolyGNN, particularly in terms of efficiency. Moreover, our framework is designed to be generic. It can potentially be extended to handle other types of point clouds, such as photogrammetric ones, and can be utilized for the reconstruction of generic piecewise planar 3D objects beyond buildings.

Finally, we remark on the gap between synthetic and real-world point clouds. In future work, we aim to bridge this gap further, enabling learning-based reconstruction methods to better abstract and leverage a vast volume of training data. Additionally, we intend to explore techniques for integrating semantic attributes to enrich the polyhedral graph and integrate plane extraction into the neural architecture.

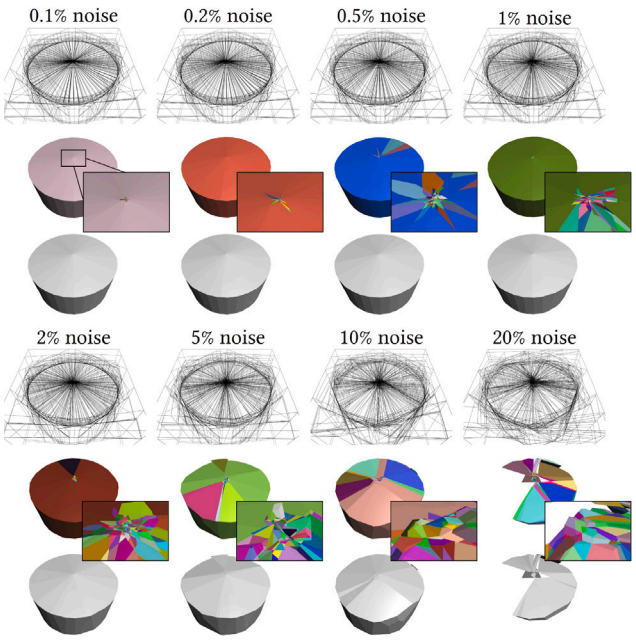

Fig. 21. Robustness to quality of planar primitives. Increasing levels of Gaussian noise are added to perturb the initial planar primitives. From top to bottom: cell complex, polyhedra classified as building components, and reconstructed model. Polyhedra are randomly color-coded.

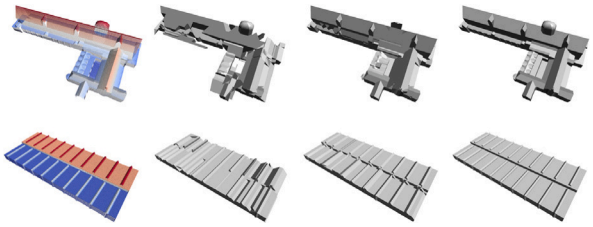

Fig. 22. Suboptimal reconstruction of buildings with complex structures. From left to right: input point cloud color-coded by height field, reconstructed model with the plain encoder, reconstructed model with the convolutional encoder, and ground truth.

CRediT authorship contribution statement

Zhaiyu Chen: Writing – original draft, Writing – review & editing, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Yilei Shi:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Liangliang Nan:** Writing – review & editing, Validation, Methodology. **Zhitong Xiong:** Writing – review & editing, Methodology, Formal analysis. **Xiao Xiang Zhu:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Xiao Xiang Zhu reports financial support was provided by German Federal Ministry of Education and Research. Xiao Xiang Zhu reports financial support was provided by German Research Foundation.

Acknowledgments

This work is jointly funded by the TUM Georg Nemetschek Institute for Artificial Intelligence for the Built World as part of the AI4TWINNING project, by the German Federal Ministry of Education and Research (BMBF) in the framework of the international future AI lab "AI4EO – Artificial Intelligence for Earth Observation: Reasoning, Uncertainties, Ethics and Beyond" (grant number: 01DD20001), by the German Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection (BMUV) based on a resolution of the German Bundestag (grant number: 67KI32002B; Acronym: *EKAPEx*) and by Munich Center for Machine Learning.

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. (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), 2842–2889.

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

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., Ledoux, H., Khademi, S., Nan, L., 2022. Reconstructing compact building models from point clouds using deep implicit fields. ISPRS J. Photogramm. Remote Sens. (ISSN: 0924-2716) 194, 58–73.

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.

Chen, D., Wang, R., Peethambaran, J., 2017. Topologically aware building rooftop reconstruction from airborne laser scanning point clouds. IEEE Trans. Geosci. Remote Sens. 55 (12), 7032–7052.

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

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.

Du, J., Zhang, S., Wu, G., Moura, J.M., Kar, S., 2017. Topology adaptive graph convolutional networks. arXiv preprint arXiv:1710.10370.

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.

Gao, X., Wu, K., Pan, Z., 2022. Low-poly mesh generation for building models. In: ACM SIGGRAPH 2022 Conference Proceedings. pp. 1–9.

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.

Gurobi Optimization, LLC, 2023. Gurobi optimizer reference manual. URL https://www.gurobi.com.

Henn, A., Gröger, G., Stroh, V., Plümer, L., 2013. Model driven reconstruction of roofs from sparse LIDAR point clouds. ISPRS J. Photogramm. Remote Sens. 76, 17–29.

Holzmann, T., Maurer, M., Fraundorfer, F., Bischof, H., 2018. Semantically aware urban 3D reconstruction with plane-based regularization. In: Proceedings of the European Conference on Computer Vision. ECCV, pp. 468–483.

Huang, J., Stoter, J., Peters, R., Nan, L., 2022. City3D: Large-scale building reconstruction from airborne LiDAR point clouds. Remote Sens. 14 (9), 2254.

Kazhdan, M., Hoppe, H., 2013. Screened Poisson surface reconstruction. ACM Trans. Graph. 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).

Lafarge, F., Alliez, P., 2013. Surface reconstruction through point set structuring. In: Computer Graphics Forum, Vol. 32. (2pt2), Wiley Online Library, pp. 225–234.

Li, Y., Liu, S., Yang, X., Guo, J., Guo, J., Guo, Y., 2023. Surface and edge detection for primitive fitting of point clouds. In: ACM SIGGRAPH 2023 Conference Proceedings. pp. 1–10.

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, L., Sung, M., Dubrovina, A., Yi, L., Guibas, L.J., 2019. Supervised fitting of geometric primitives to 3D point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2652–2660.

Li, M., Wonka, P., Nan, L., 2016b. 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.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2980–2988.

Lorensen, W.E., Cline, H.E., 1987. Marching cubes: A high resolution 3D surface construction algorithm. ACM SIGGRAPH Comput. Graph. 21 (4), 163–169.

Mura, C., Mattausch, O., Pajarola, R., 2016. Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. In: Computer Graphics Forum, Vol. 35. (7), Wiley Online Library, pp. 179–188.

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.

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

Peters, R., Dukai, B., Vitalis, S., van Liempt, J., Stoter, J., 2022. Automated 3D reconstruction of LoD2 and LoD1 models for all 10 million buildings of the Netherlands. Photogramm. Eng. Remote Sens. 88 (3), 165–170.

Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017. PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660.

Rabbani, T., Van Den Heuvel, F., Vosselmann, G., 2006. Segmentation of point clouds using smoothness constraint. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 36 (5), 248–253.

Rella, E.M., Chhatkuli, A., Konukoglu, E., Van Gool, L., 2022. Neural vector fields for implicit surface representation and inference. arXiv preprint arXiv:2204.06552.

Salinas, D., Lafarge, F., Alliez, P., 2015. Structure-aware mesh decimation. In: Computer Graphics Forum, Vol. 34. (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. (2), Wiley Online Library, pp. 214–226.

State of Bavaria, 2022. Bavarian open geodata. URL https://geodaten.bayern.de/opengeodata/.

Stucker, C., Ke, B., Yue, Y., Huang, S., Armeni, I., Schindler, K., 2022. ImpliCity: City modeling from satellite images with deep implicit occupancy fields. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. 2, 193–201.

Sulzer, R., Marlet, R., Vallet, B., Landrieu, L., 2023. A survey and benchmark of automatic surface reconstruction from point clouds. arXiv preprint arXiv:2301.13656.

Sun, Y., Mou, L., Wang, Y., Montazeri, S., Zhu, X.X., 2022. Large-scale building height retrieval from single SAR imagery based on bounding box regression networks. ISPRS J. Photogramm. Remote Sens. 184, 79–95.

Suveg, I., Vosselman, G., 2004. Reconstruction of 3D building models from aerial images and maps. ISPRS J. Photogramm. Remote Sens. 58 (3–4), 202–224.

The Sage Developers, 2021. SageMath, the sage mathematics software system (version 9.1). URL https://www.sagemath.org.

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. (5), Wiley Online Library, pp. 1553–1562.

Vanegas, C.A., Aliaga, D.G., Benes, B., 2012. Automatic extraction of Manhattan-world building masses from 3D laser range scans. IEEE Trans. Vis. Comput. Graphics 18 (10), 1627–1637.

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

Wang, R., Huang, S., Yang, H., 2023. Building3D: An urban-scale dataset and benchmarks for learning roof structures from point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 20076–20086.

Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M., 2019. Dynamic graph CNN for learning on point clouds. ACM Trans. Graph. 38 (5), 1–12.

Wichmann, A., Agoub, A., Kada, M., 2018. RoofN3D: Deep learning training data for 3D building reconstruction. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 42, 1191–1198.

Winiwarter, L., Pena, A.M.E., Weiser, H., Anders, K., Sánchez, J.M., Searle, M., Höfle, B., 2022. Virtual laser scanning with HELIOS++: A novel take on ray tracing-based simulation of topographic full-waveform 3D laser scanning. Remote Sens. Environ. 269, 112772.

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.

Yang, X., Lin, G., Chen, Z., Zhou, L., 2023. Neural vector fields: Implicit representation by explicit learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16727–16738.

Yao, S., Yang, F., Cheng, Y., Mozerov, M.G., 2021. 3D shapes local geometry codes learning with SDF. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2110–2117.

Yu, M., Lafarge, F., 2022. Finding good configurations of planar primitives in unorganized point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6367–6376.

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, X.X., Wang, Y., Shi, Y., Lachaise, M., Montazeri, S., Jancauskas, V., Kuzu, R., 2022. Global LoD-1 building model from TanDEM-X data. In: EUSAR 2022; 14th European Conference on Synthetic Aperture Radar. pp. 1–4.