PBWR: Parametric-Building-Wireframe Reconstruction from Aerial LiDAR Point Clouds

Shangfeng Huang[1], Ruisheng Wang[1,*], Bo Guo[2], Hongxin Yang[1]
[1]University of Calgary, [2]Guangdong University of Technology
{shangfeng.huang, ruiswang, hongxin.yang}@ucalgary.ca, guobo.lidar@gmail.com

Abstract

In this paper, we present an end-to-end 3D-building-wireframe reconstruction method to regress edges directly from aerial light-detection-and-ranging (LiDAR) point clouds. Our method, named parametric-building-wireframe reconstruction (PBWR), takes aerial LiDAR point clouds and initial edge entities as input and fully uses the self-attention mechanism of transformers to regress edge parameters without any intermediate steps such as corner prediction. We propose an edge non-maximum suppression (E-NMS) module based on edge similarity to remove redundant edges. Additionally, a dedicated edge loss function is utilized to guide the PBWR in regressing edges parameters when the simple use of the edge distance loss is not suitable. In our experiments, our proposed method demonstrated state-of-the-art results on the Building3D dataset, achieving an improvement of approximately 36% in Entry-level dataset edge accuracy and around a 42% improvement in the Tallinn dataset.

1. Introduction

3D-building-wireframe reconstruction is an important mid-level visual process [23]. 3D-wireframe models, lightweight in nature, offer a comprehensive representation of the shape and structural information of 3D objects. Further, these models can be easily and efficiently converted into 3D mesh models, including CAD models, that play a crucial role in the metaverse, smart cities, and virtual reality applications. Despite its practical and scientific importance, 3D-building-wireframe reconstruction remains an unsolved problem in computer vision.

Generally, point clouds are considered as one of the simplest and most flexible ways to represent 3D objects. It possesses innate ability to represent structure of objects with remarkable fidelity, capturing intricate details with precision. Wireframe models, in contrast, are also specifically designed to represent 3D objects. Unlike meshes, wireframes are a more simplified representation of 3D objects and can be readily converted into CAD. Therefore, the ABC [15] and Building3D [34] datasets recommend the use of wireframe models for point cloud reconstruction. Existing state-of-the-art deep learning methods [13, 18, 21, 43] have demonstrated impressive performance in wireframe reconstruction, but they are limited by the use of dense synthetic point clouds. Additionally, they involve heuristic-guided intermediate processing modules such as corner prediction and edge classification [18, 21, 34], which lead to error accumulation and limit performance improvement and further development.

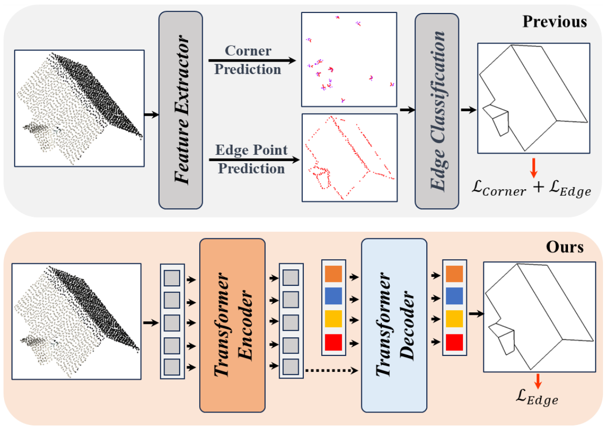

Figure 1. **Pipeline Comparison**. **Top:** The most popular wireframe-reconstruction methods based on point prediction and edge classification modules, are optimized by $\mathcal{L}_{Corner}$ and $\mathcal{L}_{Edge}$, respectively. **Bottom:** Our proposed PBWR avoids the intermediate point classification process.

In this paper, we propose a novel approach named parametric-building-wireframe reconstruction (PBWR) from aerial light-detection-and-ranging (LiDAR) point clouds. The method directly regresses edges from point clouds using transformers [32] and eliminates intermediate steps such as corner prediction and edge classification. Existing methods [18, 21, 34] predict N candidate corners that can generate C_N^2 edge proposals, where approximately 85% of the edges are considered useless. The PBWR is proposed to directly regress the desired edges to mitigate the generation of a large number of extraneous edges, as described in the ablation study.

On the other hand, these methods [18,21,34] simply rely on endpoint distances between edges for bipartite matching, often resulting in significant matching errors. We propose a novel edge-similarity method that incorporates the Hausdorff distance [7], edge length, and cosine similarity to enhance the bipartite matching, which significantly improves the performance of edge regression. Furthermore, an edge non-maximum suppression (E-NMS) module is designed to eliminate redundant positive edges, and a dedicated edge-loss function is proposed to guide the strategy of directly regressing edges.

Contributions This paper makes the following three key contributions to the field of wireframe reconstruction:

- We propose a novel method called PBWR, which directly regresses desired edges without any intermediate heuristic-guided process such as corner prediction and edge classification, and it yields a substantial number of positive candidate edges;
- This work uniquely proposes edge-similarity (Eq. (4)) and dedicated edge-loss functions in point clouds, effectively guiding the reconstruction of wireframe models. An E-NMS method based on edge similarity is proposed and applied to the wireframe-reconstruction task;
- Extensive experiments demonstrated that PBWR yielded a significant improvement in wireframe reconstruction compared with methods with intermediate heuristic steps. For example, the edge recall rate showed a notable improvement in the Entry-level dataset (from 46% to 82%) and Tallinn city dataset (from 23% to 65%).

2. Related Work

2.1. Building Reconstruction

Many optimization-based algorithms [2, 5, 8, 10, 20, 25] have been proposed for building reconstruction from point clouds. Primitive-based building reconstruction [2, 5, 8, 25, 35, 42] is a popular method for generating polygonal meshes. Other widely adopted building reconstruction methods [8, 16, 19, 33] are based on the strong Manhattan World assumption [3], which requires planes to follow only three orthogonal directions to generate dense meshes. City3D [12] combines the initial meshes generated by PolyFit [25] with vertical walls from height maps to reconstruct buildings from aerial LiDAR point clouds. Chen et al. [2] identified rooftop boundaries from aerial LiDAR point clouds and clustered them in terms of topological consistency , and they directly generated vertical walls connecting roofs. Similarly, 2.5D dual contouring [42], which extends classic dual contouring to a 2.5D method [14], employs an adaptive grid as a supporting data structure to infer the roofs in each grid node. These traditional methods have demonstrated effectiveness in practice, but they accumulate noticeable errors during the inference process. In addition, their sensitivity to parameters can lead to unstable results.

2.2. Wireframe Reconstruction

As deep learning in point cloud processing has flourished [30,39,41], the exploration of its applications in wireframe reconstruction has also grown. Initially, some studies [9, 38, 40] have attempted to treat wireframe reconstruction as an edge-point classification task, where these edge points, also known as contour points, can also represent the geometric structure of objects. Specifically, EC-Net [38] identifies object contours by reconstructing the distance distribution from each point to the edges. However, this representation is impractical. To construct solid wireframe models, recently, researcher [13, 18, 21] first predicted the positions of corners and then regressed edges from many edge proposal, as shown in Fig. 1. Some methods [1, 31] additionally predict edge points to enhance the accuracy of edge proposals. Specifically, PC2WF [21] initially refines corner positions from generated-corner candidate patches. NerVE [43] classifies contour voxels from voxelized point clouds to infer edge parameters. However, obtaining accurate corners or edge coordinates from complex scenes can frequently be error-prone, limiting the performance of subsequent edge regression. Therefore, the proposed PBWR uses a powerful transformer to directly regress edges without the need to predict corners or edge proposals. The recently innovative method [22] reconstructs wireframe models from line clouds containing thousands of edge proposals from multi-view images.

3. The Method

3.1. Motivation

When considering direct regression of an edge in 3D space, a straightforward method is to predict two endpoints of an edge. However, existing methods [24, 28] have demonstrated that directly regressing the precise positions of points in 3D space is extremely challenging. An alternative method is to determine an edge by regressing its parameters. In this work, parameterized edges are formulated as follows:

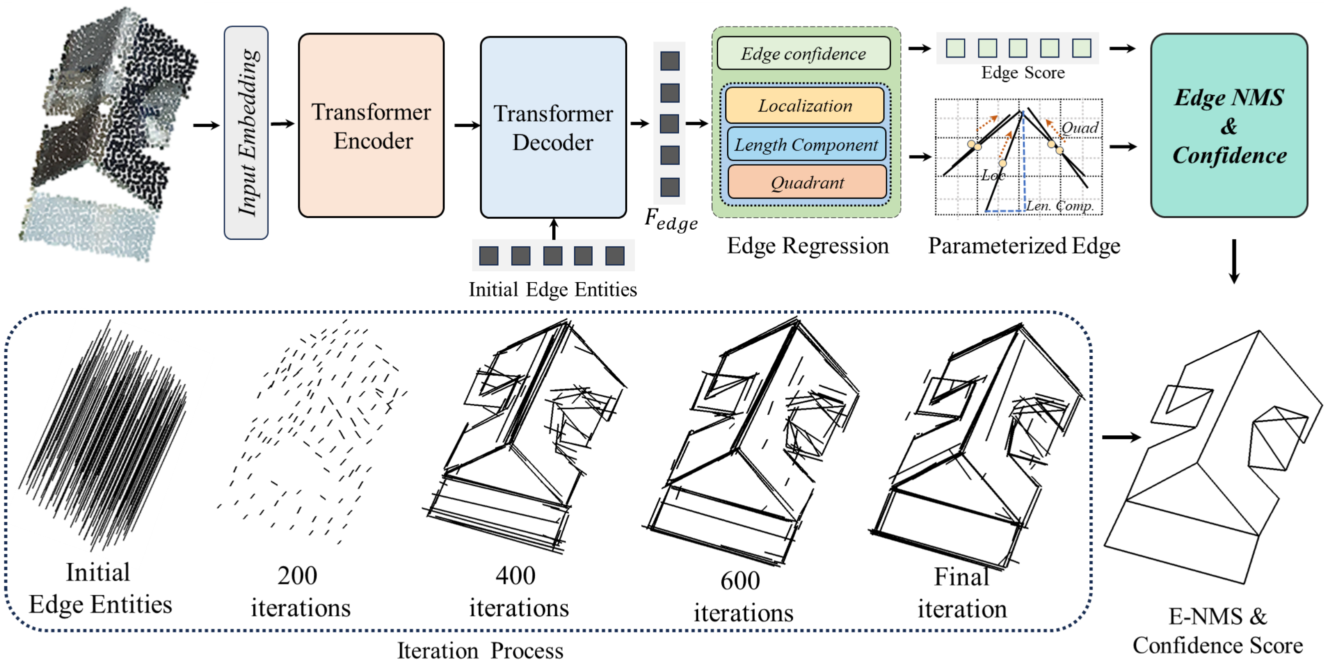

Figure 2. **PBWR Overview**. **Top**: the illustration of PBWR pipeline. It takes a roof point cloud as input and uses Input Embedding and Transformer Encoder modules to generate high-dimensional features for each point. Subsequently, point features and initial edge entities, as query embeddings, are fed into the Transformer Decoder and Edge Regression modules to obtain edge regression results, then optimized by the E-NMS to generate the wireframe model. **Bottom**: the reconstruction results are presented at different iterations during training.

$$\begin{aligned}\begin{bmatrix} V_i \\ V_j \end{bmatrix} &= \begin{bmatrix} 1 & 1/2 \\ 1 & -1/2 \end{bmatrix} \begin{bmatrix} P_m \\ \vec{v} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} P_m \\ \vec{u} \odot v' \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1/2 \\ 1 & -1/2 \end{bmatrix} \left(\begin{bmatrix} I_{3*1} \\ \vec{u}_{3*1} \end{bmatrix} \odot \begin{bmatrix} x_m & y_m & z_m \\ |v_i| & |v_j| & |v_z| \end{bmatrix} \right)\end{aligned} \tag{1}$$

where V_i and V_j represent two endpoints of an edge, P_m denotes the midpoint of the edge, and $\vec{v}$ denotes the directional vector equal to $V_i - V_j$. Additionally, $\vec{u}$ represents the symbolic matrix, I denotes an identity matrix, and $v' = [|v_i|, |v_j|, |v_z|]$ denotes the projected length of the edge along the x-, y-, and z- axes. An essential parameter of the parameterized edge is a known point on the edge. Compared with other points on an edge, regressing the midpoint (P_m) is a preferable choice to locate the position of the edge. Other two crucial parameters are the directional vector and length of the edge. The directional vector ($\vec{v}$) can be deconstructed into directional information ($\vec{u}$) and projected lengths (v') of the edge. Directional information ($\vec{u}$), also known as the symbolic matrix, essentially indicates the quadrant toward which the edge is pointing. As edges are directionless, $\vec{u}$ can be represented by four, rather than eight, quadrants. PBWR regresses edges from roof point clouds by predicting these seven parameters. We also propose PBWR-Corner, which directly regresses endpoints to determine the edge, as a comparative approach.

3.2. Overall Network Architecture

In this work, PBWR takes roof point clouds as input and predicts parameterized edges, which are then fed into the E-NMS & Confidence module to reconstruct a wireframe model, as shown in Fig. 2. For a given point cloud with N points $P \in \mathbb{R}^{N\times 7}$ encompassing coordinates, RGB information, and reflection, it is initially directed to the input embedding module, where a multi-layer perceptron (MLP) is employed to aggregate the local structural information of points. The resulting features ($F_{embed} \in \mathbb{R}^{N\times C_{embed}}$) are input to the Transformer Encoder module to obtain $F_{en} \in \mathbb{R}^{N\times C_{en}}$ features. Subsequently, the initial edge entities for edge queries and F_{en} are passed to the Transformer Decoder module for learning edge representation. Finally, the resulting edge features (F_{edge}) are parsed by the Edge Regression module, and redundant edges are removed by E-NMS. The results of edges at different iteration stages are also presented in Fig. 2 (bottom).

3.3. Transformer Encoder and Decoder

In this work, the standard transformer with the self-attention mechanism [32] is employed. The transformer encoder and decoder modules, as shown in Fig. 2, do not have any specific modifications to adapt to 3D data. Even the sampling strategies, widely employed in 3D transformer mechanisms [6, 17, 41] to reduce computational overhead and enhance performance, are not used in our approach. In PBWR, the transformer decoder is designed to learn distinguishable edge features, along with input, which comprises encoder features (F_{en}) and initial edges entities, where initialized entities are considered as object queries. Generally, object queries are often kept consistent with the input encoder module, such as point-coordinate queries [17, 41] and point-patch queries [26, 39]. Similarly, we choose M query points, using the farthest-point sampling strategy as object queries. The initial edge entities entail object queries with an initial direction and length. The M query points frequently don't align with the midpoints of the edges. Inspired by recent 3D object detection methods that predict residual offsets from original candidate points to interesting points [24, 28], PBWR predicts the distance from query points to the nearest midpoint of the edges. Specifically, M query points are input to positional embedding [32] to obtain query embeddings. The resulting query embeddings ($F_{query} \in \mathbb{R}^{M\times C_{query}}$) and point features ($F_{en}$) are fed into the transformer decoder to generate distinguishable edge features ($F_{edge} \in \mathbb{R}^{M\times C_{edge}}$). In the edge regression module, F_{edge} is parsed to obtain seven parameters for edges, including residual offsets.

3.4. Edge Regression Module

The edge regression module is designed to parse the features (F_{edge}) obtained from the transformer encoder and decoder structures to generate parameterized edges. The module employs three dedicated MLPs to regress distinct parameters of the parameterized edges, specifically P_m, v' and $\vec{u}$, as described in Eq. (1). The localization MLP predicts residual offsets from candidate points to midpoints of edges, instead of predicting midpoints directly. The orientation of the edge is collectively determined by the absolute values (v') of the components of the edge along the x-, y-, and z- axes and $\vec{u}$ serves as the symbolic matrix. The symbolic matrix can be regarded as a classification problem, specifically determining which quadrant the vector points toward, as described in Sec. 3.1. Therefore, a component MLP is employed to predict three component values, while a quadrant MLP is utilized to predict the probability of the predicted edge vector pointing from various quadrants, simply called quadrant classes. Additionally, an extra MLP is employed to obtain confidence scores for the edges.

3.5. Bipartite Edge Matching and E-NMS

Effective bipartite matching between prediction and the ground truth is crucial in deep learning. Current methods [18, 21, 34] match prediction with the ground truth by minimizing the distance between predicted corners and ground-truth corners, which naturally extends to matching edges. However, this approach is not applicable to the proposed

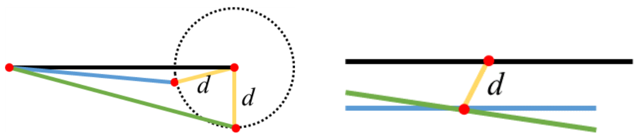

Figure 3. (a) Edge distance d quantified by corner distance, and (b) edge distance d quantified by midpoint distance.

PBWR, which circumvents the intermediate step of predicting corners. Hence, a specialized bipartite edge matching strategy is proposed to address the wireframe reconstruction problem. It is evident that distance, direction, and length can be employed to quantify similarity between edges. **Distance:** The Fig. 3a illustrates that the use of corner distance results in the same distances from blue and green to black edges. The midpoint distance also yields the same distance in the example shown in Fig. 3b. However, the blue edge is a more suitable match when considering the black edge. Therefore, the Hausdorff distance, defining the distance between two point sets, is employed to quantify edge distance for optimal bipartite edge matching:

$$\begin{aligned} H_d(e_i, e_j) &= \max\left(h_d(e_i, e_j), h_d(e_j, e_i)\right) \\ h_d(e_i, e_j) &= \max_{p_{e_i}\in e_i}\left\{\min_{p_{e_j}\in e_j}\left\|p_{e_i} - p_{e_j}\right\|\right\} \\ h_d(e_j, e_i) &= \max_{p_{e_j}\in e_j}\left\{\min_{p_{e_i}\in e_i}\left\|p_{e_j} - p_{e_i}\right\|\right\} \end{aligned} \tag{2}$$

where p_{e_i} and p_{e_j} represent points sampled uniformly from edges e_i and e_j, respectively. $H_d(e_i, e_j)$ denotes the quantified distance between edges e_i and e_j. **Direction:** Cosine similarity is utilized to evaluate the similarity in edge. **Length:** Length similarity is quantified based on the ratio of the lengths between edges. They can be formulated as follows:

$$\begin{aligned} Dir_{sim}(e_i, e_j) &= 1 - \frac{|e_i \cdot e_j|}{\|e_i\| \times \|e_j\|} \\ Len_{sim}(e_i, e_j) &= 1 - \frac{\min(\|e_i\|, \|e_j\|)}{\max(\|e_i\|, \|e_j\|)} \end{aligned} \tag{3}$$

Hence, the **edge similarity** can be represented as follows:

$$\begin{aligned} Edge_{sim}(e_i, e_j) = \alpha H_d(e_i, e_j) + \beta Dir_{sim}(e_i, e_j) \\ + \gamma Len_{sim}(e_i, e_j) \end{aligned} \tag{4}$$

where α, β, and γ denote the balancing coefficients. This indicates that when the $Edge_{sim}(e_i, e_j)$ value is close to 0, edges e_i and e_j are similar. The Hausdorff distance also implicitly considers the edge length and directional information, as evidenced by Eq. (2) and the selection of the better-matching blue edge in Fig. 3 based on the Hausdorff distance. However, a reasonable combination of all three factors contributes to improved performance, as demonstrated in the ablation study. The E-NMS algorithm takes both edge similarity and predicted-confidence scores as inputs to remove redundant edges. Details of the E-NMS pseudocode can be found in the *Supplementary materials*.

3.6. Loss Function

$Edge_{sim}\left(P_{edge}, G_{edge}\right)$ can be utilized to calculate the edge similarity between the predicted edge set (P_{edge}) and the ground truth edge set (G_{edge}). The resulting edge similarity is fed into the Hungarian algorithm to perform bipartite edge matching between N_{pos} positive predictions and the ground truth edge set.

Midpoint and Component length loss: The ℓ_1 distance loss is employed for both the midpoint positions and predicted lengths of components v' in the x-, y-, and z- axis. The midpoint loss ($\mathcal{L}_{mid}$) is formulated as:

$$\mathcal{L}_{mid} = \frac{1}{N_{pos}} \sum_{i,j \in N_{pos}} \left(p^i_{mid} - g^j_{mid}\right) \tag{5}$$

where p and g represent elements within the sets of edges, P_{edge} and G_{edge}, respectively. The component length loss ($\mathcal{L}_{comp}$) is similar to the midpoint loss.

Confidence and Quadrant classification loss: The ℓ_{CE} cross-entropy loss is used to optimize predicted confidence scores and quadrant classes as follows:

$$\begin{aligned} \mathcal{L}_{con} &= \ell_{CE}\left(p_{con}, g_{con}\right) \\ g_{con} &= \begin{cases} 1 - Edge_{sim} & \text{if } Edge_{sim} < 1 \\ 0 & \text{otherwise} \end{cases} \\ \mathcal{L}_{quad} &= \frac{1}{N_{pos}} \underset{i,j \in N_{pos}}{\ell_{CE}} \left(p^i_{quad}, g^j_{quad}\right) \end{aligned} \tag{6}$$

When simply setting the confidence scores of positive predictions to 1 and negative predictions to 0, the extreme class imbalance leads to the network failing to converge, regardless of whether cross-entropy or focal loss is used. Therefore, the confidence scores of the ground truth are redefined based on edge similarity ($Edge_{sim}$).

Edge similarity loss $\mathcal{L}_{sim}$ is directly computed as the average edge-similarity scores for N_{pos} positive predictions. Thus, the final loss ($\mathcal{L}$) is formulated as:

$$\begin{aligned} \mathcal{L} = \lambda_{mid}\mathcal{L}_{mid} + \lambda_{comp}\mathcal{L}_{comp} + \lambda_{con}\mathcal{L}_{con} \\ + \lambda_{quad}\mathcal{L}_{quad} + \lambda_{sim}\mathcal{L}_{sim} \end{aligned} \tag{7}$$

where λ_* is employed as a coefficient to balance various loss terms.

4. Experiments

4.1. Dataset and Evaluation Metric

The Building3D [34] dataset was employed to evaluate our model. Specifically, the Entry-level dataset of Building3D consists of 5,698 training point clouds and 583 testing point clouds, while the Tallinn city dataset includes 32,618 training point clouds and 3,472 testing point clouds. All the benchmark samples were extracted from sparse aerial point clouds. During training stages, data augmentation was implemented to enhance the network with desirable robustness and invariance. Specifically, we added an augment of horizontal flips along the YZ or XZ plane, along with random rotations along the z-axis (-5°∼ 5°), to increase the input shape diversity.

The same evaluation metric [34] was employed to evaluate the proposed PBWR as shown in Tab. 1. The Average Corner Offset (ACO) metric was used to evaluate the average offsets between predicted and ground truth corners. Corner Precision (CP) and Edge Precision (EP) represent the precision of corner prediction and edge prediction, respectively. The Corner Recall (CR) and Edge Recall (ER) denote the recall of corners and edges, respectively. Corner F_1 score (CF_1) and Edge F_1 score (EF_1) represent corresponding F_1 scores.

4.2. Implementation

The number of points (N) within a batch was set to 2560, which is approximately equal to the average number of points per sample in the dataset. The point features $P \in \mathbb{R}^{N\times 7}$ were fed into the input embedding module to obtain embedding features with $C_{embed} = 256$ channels. The output of the encoder was features with $C_{en} = 256$ channels; then, these features, along with $M = 128$ query points and $C_{query} = 64$ query embeddings, were both fed into the decoder to obtain F_{edge} with $C_{edge} = 256$ channels. Note that 128 query points was a reasonable setting, as demonstrated in the ablation study. The detailed parameter settings for the network, E-NMS, and other modules are provided in the *Supplementary materials*.

4.3. Results and Comparisons

Through our best efforts, we present a summary of the quantitative comparison between PBWR and existing wireframe reconstruction methods is presented in Tab. 1 and Tab. 2. Specifically, Tab. 1 shows the results on the Entry-level dataset, while Tab. 2 represents the results on the Tallinn City dataset from Building3D dataset. Experimental results demonstrate that proposed PBWR exhibited a significant margin of improvement in performance compared with the baseline provided by Building3D, With the **CR and ER increasing by 19% and 36%**, respectively, with corresponding F_1 scores showing an improvement of 15% and 25%, respectively, on the Entry-level dataset. On the Tallinn City dataset, **CR and ER increased by 15% and 42%**, respectively, with corresponding F_1 scores showing an improvement of 14% and 40%, respectively. Furthermore, the ACO distance significantly decreased. These re-

Method	Distance (m)	Accuracy					
	ACO	CP	CR	CF$_1$	EP	ER	EF$_1$
PointNet* [29]	0.36	0.71	0.50	0.59	0.81	0.26	0.39
PointNet++* [30]	0.34	0.79	0.52	0.63	0.84	0.33	0.47
RandLA-Net* [11]	0.35	0.70	0.60	0.65	0.67	0.16	0.25
DGCNN* [27]	0.32	0.73	0.58	0.65	0.81	0.30	0.44
PAConv* [37]	0.33	0.75	0.57	0.65	0.85	0.31	0.45
Stratified Transformer* [17]	0.38	0.72	0.51	0.62	0.75	0.22	0.34
Point2Roof [18]	0.30	0.66	0.48	0.56	0.71	0.26	0.38
Building3D-supervised [34]	0.26	0.89	0.66	0.76	0.91	0.46	0.61
PC2WF [21]	0.45	0.24	0.13	0.16	0.02	0.14	0.04
PBWR	0.22	0.97	0.85	0.91	0.91	0.82	0.86
PBWR-Tallinn	**0.18**	**0.99**	**0.87**	**0.93**	**0.96**	**0.84**	**0.90**

Table 1. Performance comparisons were conducted on the Entry-level data from the Building3D dataset. * indicates that this method serves as the feature extractor in the wireframe-reconstruction network. PBWR-Tallinn represents training on Tallinn city data, while evaluating on Entry-level dataset.

Method	Distance (m)	Accuracy (%)					
	ACO	CP	CR	CF$_1$	EP	ER	EF$_1$
PointMAE* [29]	0.33	0.75	0.47	0.58	0.52	0.12	0.20
PointM2AE* [30]	0.32	0.79	0.58	0.67	0.50	0.07	0.12
Point2Roof [11]	0.39	0.65	0.30	0.41	0.66	0.08	0.14
Building3D-Linear self-supervised [27]	0.35	0.70	0.60	0.65	0.67	0.16	0.25
Building3D-supervised [34]	0.29	0.90	0.53	0.66	0.88	0.23	0.36
PC2WF [21]	0.52	0.18	0.67	0.28	0.02	0.15	0.01
PBWR-Tallinn	**0.22**	**0.96**	**0.68**	**0.80**	**0.91**	**0.65**	**0.76**

Table 2. Performance comparisons were conducted on the Tallinn city data from the Building3D dataset. * indicates that this method serves as the feature extractor in the wireframe-reconstruction network.

markable improvements are a result of PBWR's avoidance of the intermediate corner prediction step, leading to reduced error accumulation and fewer constraints on the edge regression performance.

Method	Dis.	Accuracy			
	ACO	CP	CR	EP	ER
Corner Matching	0.24	0.97	0.39	0.93	0.30
Midpoint Matching	0.25	0.96	0.85	0.87	0.81
PBWR-Corner	0.44	0.90	0.10	0.06	0.12
PBWR	**0.22**	**0.97**	**0.85**	0.91	**0.82**

Table 3. Effect of different bipartite edge matching strategies and regression methods

Some methods have been proposed recently for extracting edge points, which can be further processed for generating wireframe models. We present a visual comparison with respect to these methods [21,36,38,43] and some popular traditional methods, as shown in Fig. 4. We display extracted edges in black, some extracted corners in red, and the ground truth wireframe model in gray to improve the visualization, especially as when some compared methods could only extract incomplete edges. The primary reason that these methods [21, 36, 38, 43] failed is that their designed models are based on complete 3D point clouds, making them unsuitable for the sparsity of aerial LiDAR point clouds, as well as the presence of missing data and noise in the point clouds. The quantitative comparison, including RMSE, 3D IOU, and face number between PBWR and traditional methods is detailed in the *Supplementary materials*. Fig. 5 visualizes the results of a large-scale scene reconstruction.

5. Ablation Study

Edge Generation It is not advisable to use the strategy of directly regressing two endpoints of an edge (PBWR-Corner), as it results in poor performance, as described in Tab. 3. Compared with these methods that are based on heuristic corner strategy, PBWR predicted more positive edges, as shown in Fig. 6. As the number of these edges in

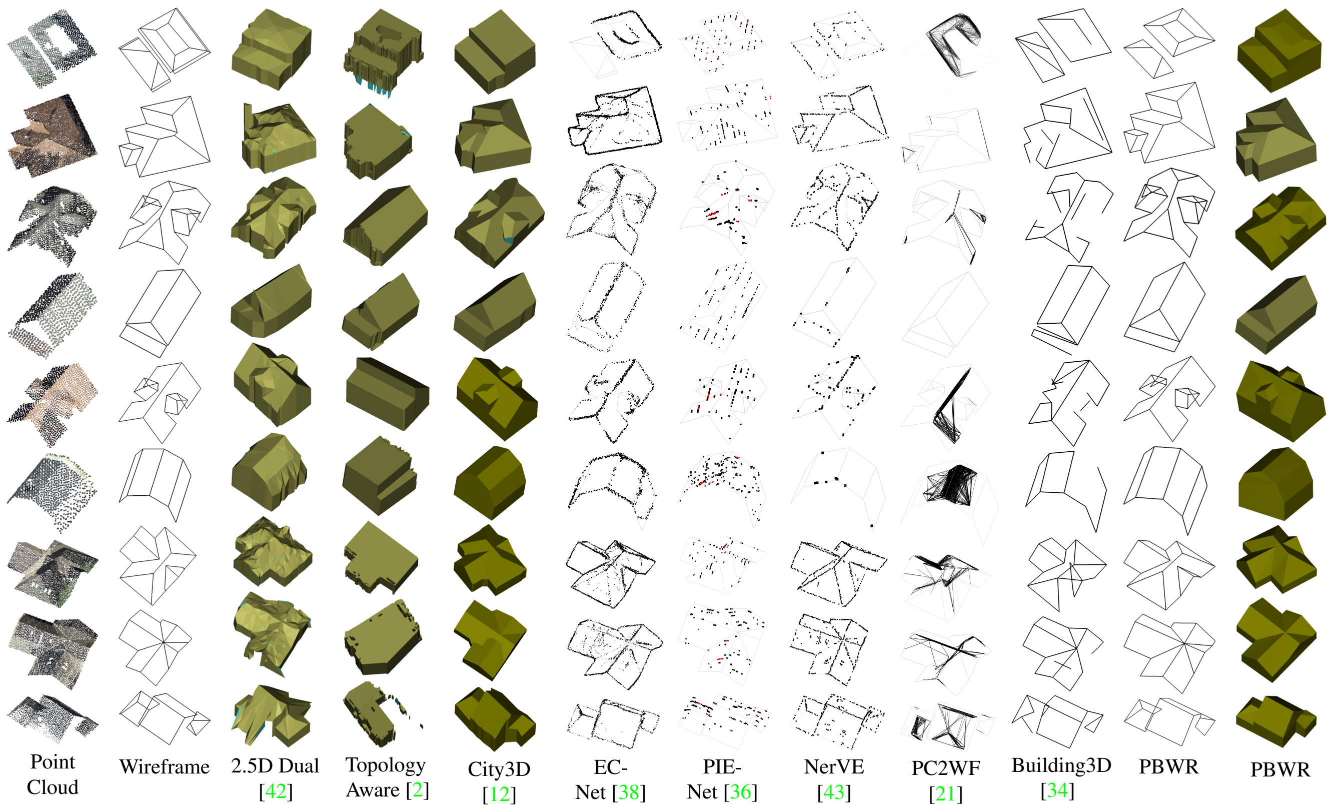

Figure 4. Qualitative evaluation of traditional and deep learning methods. Traditional methods provide mesh results, and deep learning methods provide wireframe results. To enhance visualization clarity, we incorporated the nonbolded ground-truth data as a background for better observation of the visualization results from EC-Net to PC2WF.

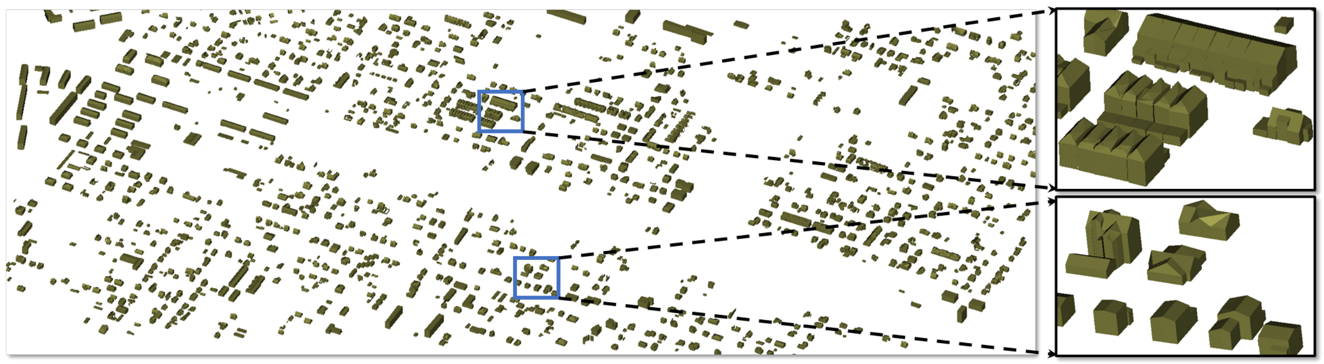

Figure 5. **Visualization of Tallinn City Data Reconstruction Results.** The wireframe-reconstruction results of buildings in a 1000 m x 2000 m area in central Tallinn city transformed into roof mesh and corresponding facade mesh

a wireframe model increased, PBWR showed a rising ratio of predicted positive edges. However, the methods based on the heuristic corner strategy, such as PC2WF [21] and PointRoof [18], represent a continual declining trend. This significantly contributed to the substantial improvement of the performance demonstrated by our method.

Bipartite Edge Matching The experimental results in Tab. 3 demonstrate that the matching strategy that minimizes the Hausdorff distances significantly outperformed strategies that match corresponding corners and midpoints. Furthermore, the subtle performance differences using the combination of the Hausdorff distance, length, and angle similarities are also shown in Fig. 7.

E-NMS The performance of the E-NMS module is demonstrated in Tab. 4. The experimental results indicate that an excessive number of redundant positive sample edges, if not removed, resulted in a rapid decline in precision (**-36% on CP, and -22% on EP**) and also affect the recall.

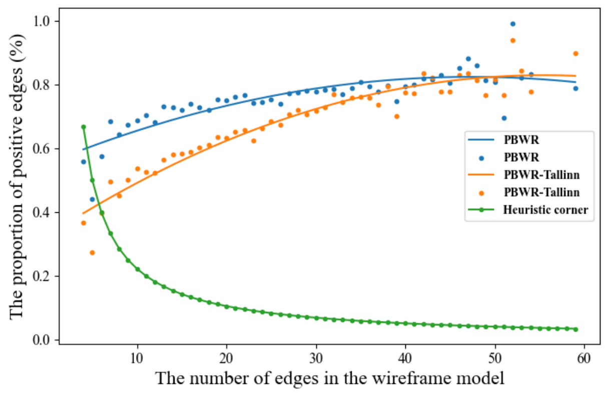

Figure 6. **Ratio of positive sample edges** Points represent the original data. The curve is fitted given the point set. Heuristic corner represents the popular method based on intermediate corner prediction.

Hard Labels	Soft Labels	$\mathcal{L}_{sim}$	E-NMS	Accuracy CP	CR	EP	ER
✓		✓		0.62	0.42	0.56	0.35
✓		✓	✓	0.98	0.50	0.78	0.41
	✓		✓	0.97	0.84	0.89	0.80
	✓	✓	✓	**0.97**	**0.85**	**0.91**	**0.82**

Table 4. **Effectiveness of modules**. Ablation study of different loss label strategies in the $\mathcal{L}_{con}$ loss and E-NMS module.

Loss Module Previous bipartite edge-matching results between prediction and ground truth can be utilized to assign ground-truth labels to the prediction. Employing hard labels, commonly depicted as 0 and 1, is a prevalent practice. However, in doing so, the positive-to-negative sample ratio would be approximately 1:10. In the confidence-score loss, the extreme sample imbalance limited the performance of the network, even when the focal loss was applied to hard labels, as shown in Tab. 4. Hence, edge similarity was utilized as soft labels, as described in Eq. (6), to alleviate the sample imbalance. In contrast to edge-confidence scores, which necessitated clearly discriminative scores for positive and negative samples, other outputs only required the optimization of the obtained positive samples. Consequently, hard labels were employed to supervise these outputs. Tab. 4 also demonstrates the significance of edge similarity loss ($\mathcal{L}_{sim}$).

Number of Queries Fig. 7 depicts the influence of varying quantities of query points on performance. An insufficient number of points made it impractical to generate enough predicted edges corresponding to the ground truth. Conversely, an excess of points resulted in an abundance of redundant edges, thereby increasing the network burden.

Wireframe Model Generation During evaluation, the confidence-score threshold was set to 0.7, effectively eliminating unexpected edges from the original network output,

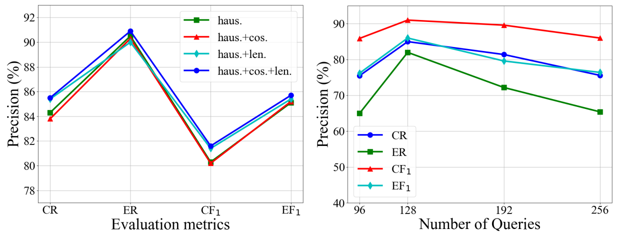

Figure 7. **Left**: Comparison of different combinations of matching strategies. **Right**: comparison of impact of four different query points on performance.

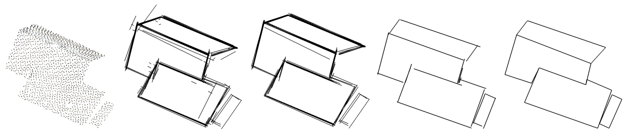

Figure 8. **Visualization of model generation.**: Left to right: point clouds, network output, confidence processing, E-NMS, and wireframe generation

as shown in Fig. 8. The E-NMS algorithm was subsequently employed to eliminate remaining redundant edges. To obtain a seamlessly connected wireframe model, we use the DBSCAN algorithm [4] is utilized to merge corners, using centroids of the clusters as new corners. All accuracy evaluation were based on the resultant wireframe models. Note that the DBSCAN algorithm, with a distance threshold of 0.05 and a minimum point count of 2, was not used to generate any new edges.

6. Conclusion

In this paper, we proposed PBWR, an end-to-end wireframe reconstruction model that directly regresses edges, bypassing intermediate heuristic modules for corner or edge prediction. The resultant parameterized edges undergo bipartite edge matching using the proposed Hausdorff distance-based similarity algorithm. E-NMS leverages edge similarity as a crucial parameter to eliminate redundant positive sample edges. Additionally, a loss function specifically designed for edge optimization is proposed to guide the network optimization and model generation. In experiments, PBWR achieved performance far beyond that of existing baselines.

Limitations and Future Work One limitation of this work is that the edges generated by PBWR are not a continuously connected edge set, as shown in Fig. 8. Fortunately, corners of the obtained edge sets are closely located. Therefore, the DBSCAN algorithm [4] with a small distance threshold of 0.05 m is employed to aggregate endpoints. In our opinion, this strategy might not be prudent, as it will result in unforeseen errors. In our upcoming research, our emphasis will be on eliminating the need for DBSCAN processing, instead of directly generating a continuously connected set of edges.

References

[1] Li Cao, Yike Xu, Jianwei Guo, and Xiaoping Liu. Wireframenet: A novel method for wireframe generation from point cloud. *Computers & Graphics*, 115:226–235, 2023. 2

[2] Dong Chen, Ruisheng Wang, and Jiju Peethambaran. Topologically aware building rooftop reconstruction from airborne laser scanning point clouds. *IEEE Transactions on Geoscience and Remote Sensing*, 55(12):7032–7052, 2017. 2, 7

[3] James Coughlan and Alan L Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. *Advances in Neural Information Processing Systems*, 13, 2000. 2

[4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, pages 226–231, 1996. 8

[5] Hao Fang and Florent Lafarge. Connect-and-slice: an hybrid approach for reconstructing 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13490–13498, 2020. 2

[6] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021. 4

[7] Felix Hausdorff. Dimension und äußeres maß. *Mathematische Annalen*, 79(1-2):157–179, 1918. 2

[8] Zhenbang He, Yunhai Wang, and Zhanglin Cheng. Manhattan-world urban building reconstruction by fitting cubes. In *Computer Graphics Forum*, pages 289–300. Wiley Online Library, 2021. 2

[9] Chems-Eddine Himeur, Thibault Lejemble, Thomas Pellegrini, Mathias Paulin, Loic Barthe, and Nicolas Mellado. Pcednet: A lightweight neural network for fast and interactive edge detection in 3d point clouds. *ACM Transactions on Graphics (TOG)*, 41(1):1–21, 2021. 2

[10] Thomas Holzmann, Michael Maurer, Friedrich Fraundorfer, and Horst Bischof. Semantically aware urban 3d reconstruction with plane-based regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 468–483, 2018. 2

[11] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11108–11117, 2020. 6

[12] Jin Huang, Jantien Stoter, Ravi Peters, and Liangliang Nan. City3d: Large-scale building reconstruction from airborne lidar point clouds. *Remote Sensing*, 14(9):2254, 2022. 2, 7

[13] Tengping Jiang, Yongjun Wang, Zequn Zhang, Shan Liu, Lei Dai, Yongchao Yang, Xin Jin, and Wenjun Zeng. Extracting 3-d structural lines of building from als point clouds using graph neural network embedded with corner information. *IEEE Transactions on Geoscience and Remote Sensing*, 2023. 1, 2

[14] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, 2002. 2

[15] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[16] Florent Lafarge and Clément Mallet. Building large urban environments from unstructured point data. In *2011 International Conference on Computer Vision*, pages 1068–1075. IEEE, 2011. 2

[17] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8500–8509, 2022. 4, 6

[18] Li Li, Nan Song, Fei Sun, Xinyi Liu, Ruisheng Wang, Jian Yao, and Shaosheng Cao. Point2roof: End-to-end 3d building roof modeling from airborne lidar point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 193:17–28, 2022. 1, 2, 4, 6, 7

[19] Minglei Li, Peter Wonka, and Liangliang Nan. Manhattan-world urban reconstruction from point clouds. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 54–69. Springer, 2016. 2

[20] Hui Lin, Jizhou Gao, Yu Zhou, Guiliang Lu, Mao Ye, Chenxi Zhang, Ligang Liu, and Ruigang Yang. Semantic decomposition and reconstruction of residential scenes from lidar data. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013. 2

[21] Yujia Liu, Stefano D'Aronco, Konrad Schindler, and Jan Dirk Wegner. Pc2wf: 3d wireframe reconstruction from raw point clouds. *arXiv preprint arXiv:2103.02766*, 2021. 1, 2, 4, 6, 7

[22] Yicheng Luo, Jing Ren, Xuefei Zhe, Di Kang, Yajing Xu, Peter Wonka, and Linchao Bao. Learning to construct 3d building wireframes from 3d line clouds. *arXiv preprint arXiv:2208.11948*, 2022. 2

[23] D Man and A Vision. A computational investigation into the human representation and processing of visual information. *WH San Francisco: Freeman and Company, San Francisco*, 1, 1982. 1

[24] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2906–2917, 2021. 2, 4

[25] Liangliang Nan and Peter Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2353–2361, 2017. 2

[26] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European conference on computer vision*, pages 604–621. Springer, 2022. 4

[27] Anh Viet Phan, Minh Le Nguyen, Yen Lam Hoang Nguyen, and Lam Thu Bui. Dgcnn: A convolutional neural network over large-scale labeled graphs. *Neural Networks*, 108:533–543, 2018. 6

[28] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019. 2, 4

[29] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 6

[30] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2, 6

[31] Xuefeng Tan, Dejun Zhang, Long Tian, Yiqi Wu, and Yilin Chen. Coarse-to-fine pipeline for 3d wireframe reconstruction from point cloud. *Computers & Graphics*, 106:288–298, 2022. 2

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 3, 4

[33] Vivek Verma, Rakesh Kumar, and Stephen Hsu. 3d building detection and modeling from aerial lidar data. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2213–2220. IEEE, 2006. 2

[34] Ruisheng Wang, Shangfeng Huang, and Hongxin Yang. Building3d: A urban-scale dataset and benchmarks for learning roof structures from point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20076–20086, October 2023. 1, 2, 4, 5, 6, 7

[35] Senyuan Wang, Guorong Cai, Ming Cheng, José Marcato Junior, Shangfeng Huang, Zongyue Wang, Songzhi Su, and Jonathan Li. Robust 3d reconstruction of building surfaces from point clouds based on structural and closed constraints. *ISPRS Journal of Photogrammetry and Remote Sensing*, 170:29–44, 2020. 2

[36] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. Pie-net: Parametric inference of point cloud edges. *Advances in neural information processing systems*, 33:20167–20178, 2020. 6, 7

[37] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3173–3182, 2021. 6

[38] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *Proceedings of the European conference on computer vision (ECCV)*, pages 386–402, 2018. 2, 6, 7

[39] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19313–19322, 2022. 2, 4

[40] Weini Zhang, Linwei Chen, Zhangyue Xiong, Yu Zang, Jonathan Li, and Lei Zhao. Large-scale point cloud contour extraction via 3d guided multi-conditional generative adversarial network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 164:97–105, 2020. 2

[41] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021. 2, 4

[42] Qian-Yi Zhou and Ulrich Neumann. 2.5 d dual contouring: A robust approach to creating building models from aerial lidar point clouds. In *European conference on computer vision*, pages 115–128. Springer, 2010. 2, 7

[43] Xiangyu Zhu, Dong Du, Weikai Chen, Zhiyou Zhao, Yinyu Nie, and Xiaoguang Han. Nerve: Neural volumetric edges for parametric curve extraction from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13601–13610, 2023. 1, 2, 6, 7