A Two-Stage Method for 3D Architecture Wireframe Reconstruction from Airborne LiDAR Point Cloud

Jiahao Zhang, Qi Liu, Le Hui, Yuchao Dai
Northwestern Polytechnical University, Xi'an, China
E-mail: jiahao2023@mail.nwpu.edu.cn,
{liuqi, huile, daiyuchao}@nwpu.edu.cn

***Abstract*—The 3D roof reconstruction from airborne LiDAR point clouds is an important mid-level vision process in computer vision. Most of the existing three-dimensional roof reconstruction models are trained and tested on simulated roof datasets. Compared with the complex real roof point cloud environment, it is difficult to regress the precise roof wireframe. Based on the Building3D Tallinn dataset, we propose a deep learning-based two-stage approach to reconstruct the roof wireframe from airborne LiDAR point cloud. Initially, the method aims to identify the locations of vertices, followed by predicting the edges connecting these vertices. In the first stage, the primary concept involves offset regression from candidate corner points and utilizing clustering to label the actual corner of the point cloud. The corner points are fully connected between two pairs to execute the second classification of the projected edge, while the Graph Neural Network is employed to extract the feature for label regression, ultimately leading to the final result. In the experiments, we provide several mainstream backbone network feature extraction methods for comparison, and the top-performing method outperformed the best approach reported by the Building3D Tallinn dataset.**

I. Introduction

The utilization of 3D wireframe models has become increasingly prevalent in various fields such as computer vision, architecture, and geospatial analysis. One of the foremost advantages of 3D wireframe models is their efficiency in representing complex geometric shapes with minimal data. Unlike surface or solid models, wireframe models utilize lines and curves to delineate the edges of an object, which significantly reduces computational load and memory usage. Moreover, in the context of airborne LiDAR point cloud data, 3D wireframe models play a pivotal role in efficiently reconstructing architectural features. Their ability to accurately capture the geometric framework of buildings from sparse point cloud data makes them indispensable for applications in remote sensing and geospatial intelligence. This efficiency and accuracy are particularly valuable in scenarios where rapid and large-scale data processing is required. Although the 3D wireframe reconstruction of buildings has high application value and scientific significance, it still remains a challenging problem in computer vision tasks.

Point clouds are a versatile representation of three-dimensional (3D) data, widely utilized in various fields such as computer vision, robotics, and architectural analysis. They provide a rich source of spatial information by capturing the 3D coordinates of numerous points on object surfaces. Despite their advantages, the raw point cloud data often lack structural connectivity, making it challenging to directly infer the underlying geometric structures. Meanwhile, Wireframes offer a simplified yet informative representation by highlighting the essential edges and vertices of objects. They facilitate efficient geometric analysis and visualization, bridging the gap between raw point clouds and structured 3D models. The task of regressing wireframes from point clouds is thus a critical step in converting raw spatial data into meaningful geometric representations. The existing methods for wireframe regression from airborne LiDAR point cloud predominantly rely on synthetic datasets. These datasets, while useful for controlled experiments and method development, often fail to capture the complexity and variability of real-world scenarios. In particular, for applications involving rooftop point cloud datasets, the regularity and simplicity of synthetic data do not adequately reflect the irregular and intricate nature of actual rooftops. This mismatch significantly impairs the applicability and performance of these methods on real-world data.

The regression of wireframe models from rooftop point clouds is a significant task in the fields of computer vision and 3D modeling, with applications ranging from urban planning to augmented reality.In recent years, many traditional data-driven algorithms have been proposed. However, there are two fundamental disadvantages of data-driven algorithms. First, their accuracy is based on the precision of the proposed geometric primitives. Second, their operation sequence is progressive, which easily leads to accumulated errors.

To address the above issues, inspired by Point2Roof[1], we propose a deep learning-based two-stage approach to reconstruct the roof wireframe from airborne LiDAR point cloud. In terms of model training, we have tried several trunk feature extraction networks and adopted a new loss function. Moreover, the labeling strategy for points has been improved. Finally, we have achieved the best results on the largest real airborne point cloud dataset–Building3D[2]. The main contributions of our paper are as follows:

- We propose a new two-stage method for 3D Architecture Wireframe reconstruction from airborne LiDAR point cloud, which first regresses the corners and then regresses the wireframe models based on the corners. This method can be applied to the existing large-scale real datasets.
- In our experiments, we compared the effectiveness of various current mainstream backbone feature extraction

networks for corner feature extraction from point clouds, providing a reference to facilitate future research.

II. Related Work

To our knowledge, there are mainly three methods to reconstruct roof models from point clouds: model-driven methods, data-driven methods, and deep learning-based methods.

A. Model-driven methods

The model-driven method, as a top-down approach, is based on the principle of presupposing buildings as a combination of several simple parameterized shapes. The key issue of the model-driven methods is how to define a suitable roof shape and estimate the optimal parameters of the roof model. Li and Shan [3] proposed a multi-primitive reconstruction method for generating 3D building models from point clouds. This method initially employs the RANSAC plane segmentation algorithm to extract planar patches of each building. Subsequently, a metric method is defined to select potential architectural primitive types from predefined primitives. Finally, parameter estimation is conducted on multiple primitives to reconstruct a topologically consistent model. Although model-driven approaches can maintain the integrity of prediction models as much as possible when dealing with sparse and low-density point clouds, it may be challenging to propose a set of geometric shapes that can cover all buildings when confronted with complex real-world roof models.

B. Data-driven methods

The data-driven method, also known as the bottom-up approach, is generally divided into two stages. In the first stage, a point segmentation algorithm is used to extract major roof primitives from the input LiDAR point clouds. In the second stage, they reconstruct the 3D building model by analyzing the original topological structure. Nan and Wonka [4] proposed a framework named PolyFit for reconstructing manifold and watertight polygon surface models from extracted planar patches. It initially utilizes a RANSAC-based plane segmentation algorithm [5] to generate candidate planes. Subsequently, an optimal subset of planes is selected from these candidates through energy optimization to generate the final polygon surface. The data-driven approach can achieve remarkable results when dealing with complete and high-quality datasets, yet it still encounters two fundamental issues, which were discussed in Section 1.

C. Deep learning-based methods

With the continuous development of deep learning in point cloud processing in recent years, many methods for extracting deep features of point clouds have been proposed successively. PointNet++[6] uses multi-scale feature extraction for point clouds, PointTransformer[7] uses attention mechanism to extract deep features, and SPVCNN[8] uses sparse convolution to voxelize point clouds and then performs 3D convolution to obtain features. These backbone feature extraction networks provide diverse options for wireframe reconstruction. Among the existing methods, Point2Roof uses PointNet++ to extract deep features of point clouds, but it has poor performance in real sparse airborne LiDAR point cloud scenes of roofs. Thus, we propose a two-stage wireframe regression model based on PointTransformer.

III. Method

As we know, a wireframe is consisted of a group of corners and their connection relationship.So we use a two-stage end-to-end deep learning method to regress wireframe from input cloud points. In the first step, our method use PointTransformer V2 to extract point-wise feature from input airborne LiDAR cloud points, which used to classify the input cloud points based on our label strategy, and generate the offset of each input point cloud for the corresponding corner at the same time. And then use DBSCAN algorithm to merge corners from predicted classifications and offsets. Then, the location and feature of these predicted corners are refined by point-wise features.Then in the second stage, we score the connection relationship between each point based on corner features, determine whether there is a connection relationship between two points based on the score, and then fuse the connection relationship with the corner points to reconstruct a complete wireframe. Next, a detailed introduction to the method will be provided based on three parts.The whole pipeline of Point2Roof is shown in Fig. 1.

A. Point-wise feature extracting and cluster prediction

Our method employs PointTransformer V2 as the feature extraction backbone network. Compared with PointNet++, this network can combine global features with local features, making it more capable of acquiring contextual information, and it is highly robust and adaptive to sparse point cloud data.

Label generation.Because the wireframe ground truth cannot be directly used in model training. We supply a new label strategy, which will generate two types of labels. Firstly, the classification labels will determine whether there are true corner points within its spherical radius range, And the offset labels indicates the offset between it and the corresponding corner point.

For the classification labels, if a cloud point has a ground truth corner within its radius will be assigned label 1; otherwise the label 0. And for the offset labels, if the classification label corresponding to this point is 0, the offset label is [0,0,0]. If it is 1, the offset label Δp is represented as follows:

$$\Delta p = \left[\frac{\mathbf{p}_x - \mathbf{c}_x}{r}, \frac{\mathbf{p}_y - \mathbf{c}_y}{r}, \frac{\mathbf{p}_z - \mathbf{c}_z}{r} \right] \tag{1}$$

Where $(\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z)$ and $(\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z)$ are the coordinates of point and its corresponding nearest corner.

Feature extraction.We use PointTransformer v2 as the backbone network for feature extraction. Point Transformer V2 is an improved point cloud processing model that enhances the ability to capture local and global geometric information through a more efficient self attention mechanism and

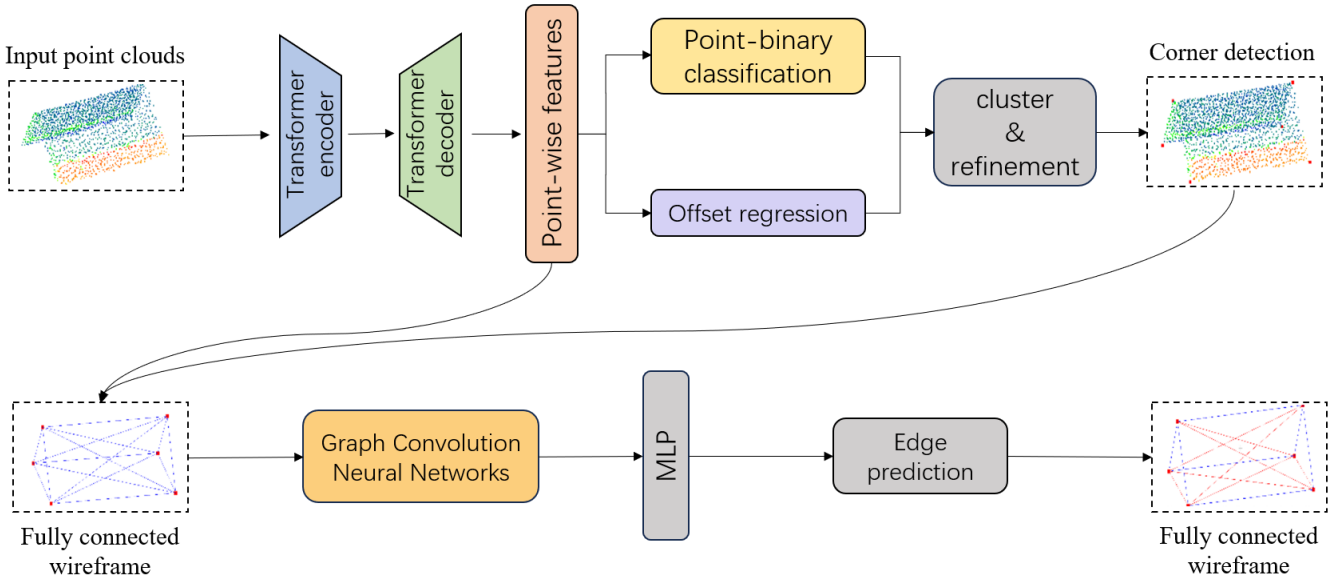

Fig. 1. The framework diagram of our usage method. The upper part is used to detect corners, while the lower part detects edges

optimized feature representation methods, enhancing computational efficiency and robustness. It performs well in point cloud classification tasks and can more accurately recognize and classify complex 3D shapes, making it suitable for multiple fields such as autonomous driving and robot navigation.

In our method, we input the normalized coordinates and RGB information of the point cloud as features into the PointTransformer V2, and obtain deep features with 64 channels, which facilitate point-wise binary classification and offset prediction. The obtained positive points, combined with the predicted offset , will yield the predicted point clusters for subsequent clustering.

B. Candidate point clustering

After passing through the backbone network, we obtained the predicted candidate corners near the corresponding truth inflection points. For these points, we used DBSCAN clustering method to cluster them into one point, which is the predicted corner. However, there is often an offset gap between the predicted corners and the truth points. Therefore, for each clustered corner, an offset relative to the truth corner is labeled. Then, the feature of each point cloud point is weighted, and the prediction score is extracted at multiple scales to extract the final predicted corner feature, thereby obtaining the predicted offset of the predicted point.The visualization of corner point prediction results is shown in Fig. 2.

C. Edge feature extraction

By using the previous stage, the predicted corners and feature information for each point have been obtained. After pairwise connection of each scene's points and comparison with the truth values, the truth labels can be obtained. After passing the obtained features through a Graph convolutional neural network (GCN) module, the features for each edge can be obtained. The GCN model focuses on extracting features from the feature relationships between corners themselves and between corners. It can carefully select salient features for each edge. For each corner features $\mathbf{C} = \{c_1, c_2, \ldots, c_d\}$, we first directly form a feature matrix $\mathbf{H}^{(0)} \in R^{n \times d}$. Then, we feed the added features into server GCN layers to obtain deeper corner features. Then we further use a feature concatenation operator to extract the final edge features. The detailed process of the GCN module can be expressed as:

$$\begin{aligned} \mathbf{H}^{(l+1)} &= \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \\ \mathbf{F}_{ij} &= \mathbf{H}_i^{(L)} \| \mathbf{H}_j^{(L)} \end{aligned} \tag{2}$$

Among them, σ represent ReLU function. In addition, the $\mathbf{W}^{(l)}$ is the weights of the l-th layer of GCN, whereas $\hat{\mathbf{A}}$ is the normalized adjacency matrix. Subsequently, we use an MLP layer to further extract edge features and perform binary classification.

D. Loss function

Our method is end-to-end model training, so the overall loss function is also composed of three parts: L_{pw} for the point-wise classification and offset regression, $L_{cluster}$ for clustering refinement, and Edge Regression loss function L_{edge}

$$L = L_{pw} + L_{cluster} + L_{edge} \tag{3}$$

The backbone network loss function L_{pw} consists of two parts: a point-wise binary classification loss function L_{cls} and an offset regression loss function L_{reg}. According to our investigation, the number of negative points in the label

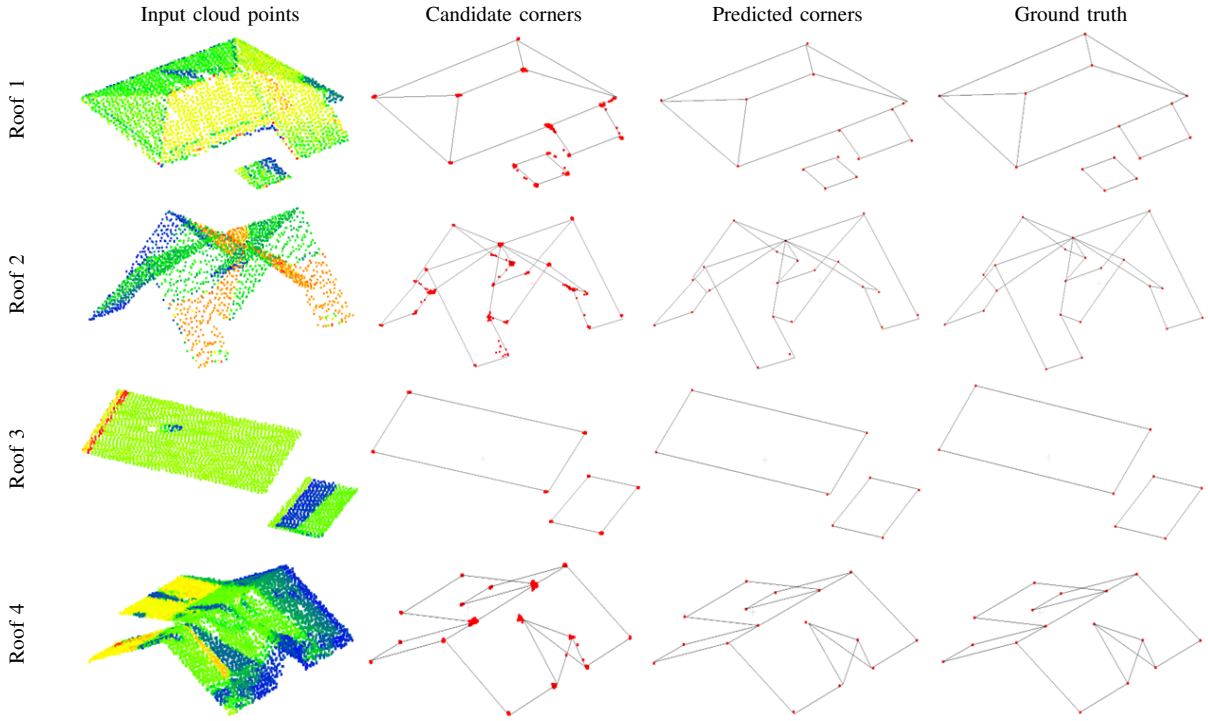

Fig. 2. Demonstrating the corner prediction process in stages.

of the model is much greater than the number of positive points. We use the commonly used focus loss function for the classification loss function and use smooth L_1 loss for the offset regression loss function.These two loss functions are defined as:

$$\begin{aligned} L_{pw} &= L_{\text{cls}} + L_{\text{offset}} \\ L_{cls} = L_{\text{cls}} &= \sum_{p} L_{\text{bce}}(s_p, l_{\text{cls}}(\mathbf{p})) \cdot \frac{1}{\max(N_{pn}, 1)} \\ L_{\text{offset}} &= \frac{1}{N_{pn}} \sum_{\mathbf{n}} L_{\text{smooth-Ll}}(\Delta\hat{\mathbf{n}}, \Delta_{\text{reg}}(\mathbf{n})) \cdot \eta(s_{\mathbf{n}}) \end{aligned} \tag{4}$$

Among them, s_n and $l_{cls}(n)$ are the predicted classification labels and truth labels for each point n, N_{pn} is the number of labeled positive points, $\Delta\hat{\mathbf{n}}$ and Δ_{reg} are the predicted offset values and true offset values, respectively.$\eta(\cdot) = 1$ if $s_n = 1$; otherwise $\eta(\cdot) = 0$.

For the offset regression loss function of clustering refinement $L_{cluster}$ and the loss function of edge binary classification L_{edge}, they use smooth-L1 loss and focal loss[9], respectively.

$$\begin{aligned} L_{cluster} &= L_{\text{offset}} \\ L_{\text{edge}} &= \sum_{\mathbf{e}} L_{\text{bce}}(s_{\mathbf{e}}, l_{\text{edge}}(\mathbf{e})) \cdot \frac{1}{\max(N_{en}, 1)} \end{aligned} \tag{5}$$

where $s_{\mathbf{e}}$ and l_{edge} refer to the predicted confidence and the label of each edge e.N_{en} is the number of labeled positive edges.

IV. Experiment

A. Implementation details

For the PointTransformer module, the hyperparameters we use are consistent with the Point Transformer V2 Mode 2 provided by PointTransformer v2. After generating features, we added sequential models for classification and offset prediction, and finally output three channel offset prediction and one channel classification prediction. For the corner feature extraction module, we apply PointNet++ MSA module to extract the initial corner features. The multi-scale receiving radii are 0.1 and 0.2. The final number of channel corner features is 128. For the corner refinement module, the network structure is the same as the offset regression head. In the GCN module, we use two GCN layers for each corner feature. Then, apply the FC layer to generate one-dimensional labels. All experiments were conducted on 8 RTX 3090 machines. For self-supervised training, the model is optimized by AdamW with a basic learning rate of 0.0001 and a batch size of 64. The number of training points is 2048. Data denoising includes Gaussian perturbation and random rotation, with probabilities of 0.8 and 0.5, respectively. The learning rate scheduler uses MultiStepLR.

B. Evaluation metric

According to the evaluation indicators requested by the competition on the official website, we can submit our running results for testing to obtain the performance of our method. The introduction of the evaluation indicators are as follows: **Wireframe edit distance** (WED) is a variant of the graph edit distance (GED). It measures how many elementary graph edit

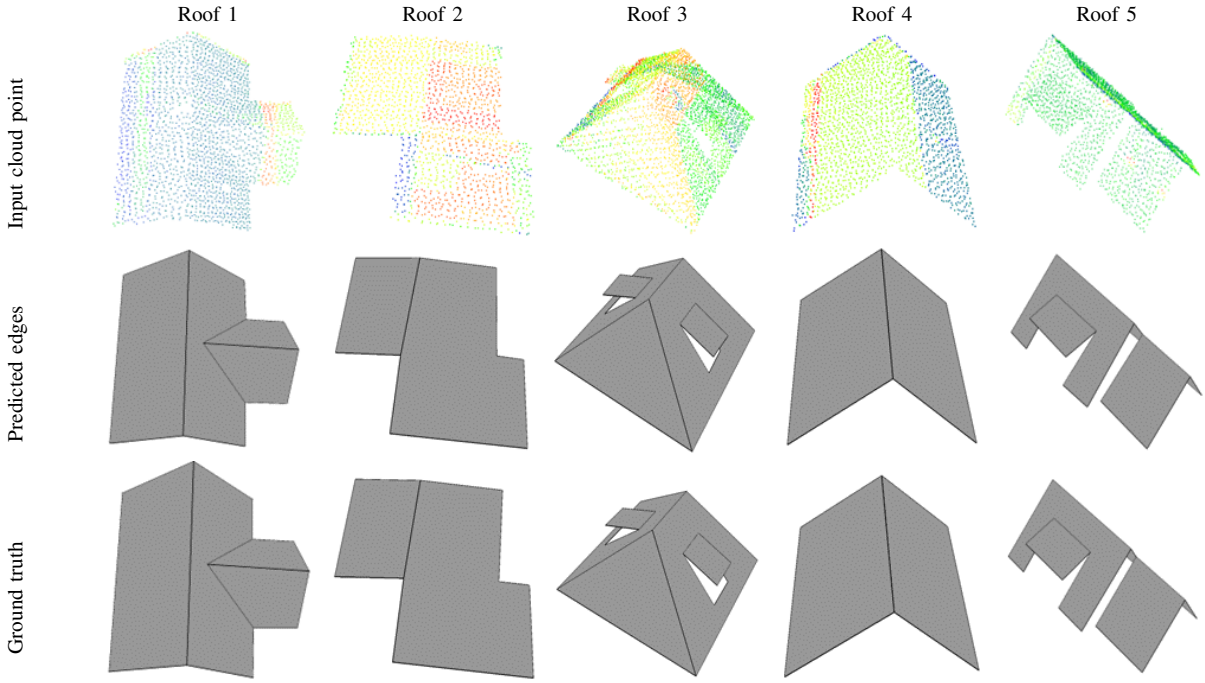

Fig. 3. Edge prediction results

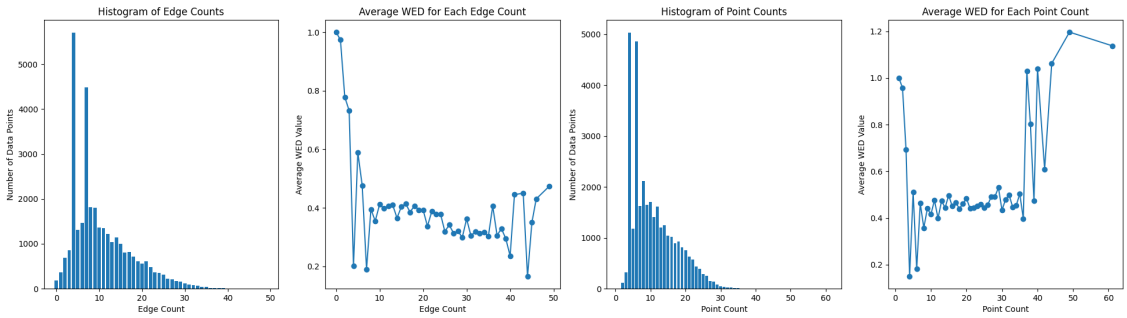

Fig. 4. The horizontal axis of the two graphs on the left represents the predicted number of edges or corners, the vertical axis represents the total number of data under that number, the horizontal axis of the two graphs on the right also represents the predicted number of edges or corners, and the vertical axis represents the average WED of the data under each number.

operators (corner insertion/deletion, edge insertion/deletion, etc.) are needed to transform predicted wireframe to ground truth wireframe, which firstly introduced in the PC2WF[10]. **Average corner offset** (ACO) is the average offsets between predicted corners and ground-truth corners. The smaller offsets indicate better quality of generated models. **Corner precision** (CP), **edge precision** (EP), **corner recall** (CR), and **edge recall** (ER) are calculated through confusion matrix to evaluate the accuracy of corner and edge classification. larger CP and EP values indicate more precise classification of corners and edges, while larger CR and ER values indicate lower rates of missing edge detection.

$$\begin{aligned} \mathrm{CP} &= \frac{C_{TP}}{C_{TP}+C_{FP}}, \quad \mathrm{CR} = \frac{C_{TP}}{C_{TP}+C_{FN}} \\ \mathrm{EP} &= \frac{E_{TP}}{E_{TP}+E_{FP}}, \quad \mathrm{ER} = \frac{E_{TP}}{E_{TP}+E_{FP}} \end{aligned} \tag{6}$$

CF1 and **EF1** represent the F1 score of corners and edges, respectively.

$$CF1 = \frac{2CP \times CR}{CP + CR}, \quad EF1 = \frac{2EP \times ER}{EP + ER} \tag{7}$$

C. Comparative experiments

During the competition, we conducted partial control experiments, and the research results showed that our method has adopted a better strategy. We used the backbone networks of PointNet++ and PointTransformer respectively, and used KNN query and ball query for candidate point annotation methods.

We tested Tallinn's train dataset on the optimal checkpoint model and classified the dataset based on the predicted number of edges or corners. Firstly, we separately counted the number of detected edges or corners, and then calculated the average WED value for each edge or corner. The statistical graph is

TABLE I
COMPARATIVE EXPERIMENTS BETWEEN DIFFERENT METHODS.

Methods	WED	ACO	CP	CR	CF1	EP	ER	EF1
PC2WF	-	0.45	0.24	0.13	0.16	0.02	0.14	0.04
Building3D	-	0.29	0.90	0.53	0.66	0.88	0.23	0.36
Ours(SPVCNN)	0.783	0.240	0.897	0.771	0.829	0.764	0.287	0.417
Point2Roof	0.447	0.282	0.938	0.634	0.756	0.865	0.423	0.568
Ours(PointTransformer)	**0.345**	**0.235**	**0.967**	**0.69**	**0.80**5	**0.875**	**0.556**	**0.68**

TABLE II
ABLATION STUDY OF THE GCN MODULE.

Edge extra model	WED	ACO	CP	CR	EP	ER
Concatenation	0.465	0.265	0.953	0.653	0.735	0.425
GCN	**0.345**	**0.235**	**0.967**	**0.69**	**0.875**	**0.68**

shown in Fig .3. According to the experimental results, it can be found that we will get better results when detecting an intermediate number of corners or edges. Errors only occur when facing poor quality point clouds of some roofs. The results of comparative experiments between different methods are shown in TABLE I, and the edge prediction results are partly shown in Fig. 3.

D. Ablation study

To validate the effectiveness of our graph convolutional neural network, we compared it with the concatenation layer. As observed, when using the graph convolutional neural network and the concatenation layer respectively, their performance on the indicators CP and CR were basically the same. However, the graph convolutional neural network demonstrated better results in terms of EP, ER and WED. The experimental results indicate that the proposed edge prediction network can effectively enhance the effectiveness of edge prediction.The results of the ablation study are shown in TABLE II.

V. CONCLUSIONS

To directly reconstruct the wireframe structure from airborne LiDAR point clouds, we devised a two-stage roof reconstruction approach based on deep learning. We divided the task into two parts: corner prediction and edge prediction. Furthermore, our method can directly generate models applicable to large-scale real-world roof point cloud datasets. Experimental results demonstrate that our method can accurately predict the positions of corners and edges on the Tallinn dataset of build3D, and our current approach significantly outperforms traditional methods trained on simulated datasets.

REFERENCES

[1] "Point2roof: End-to-end 3d building roof modeling from airborne lidar point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 193, pp. 17–28, 2022.

[2] R. Wang, S. Huang, and H. Yang, "Building3d: An urban-scale dataset and benchmarks for learning roof structures from point clouds," *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20 019–20 029, 2023.

[3] "Ransac-based multi primitive building reconstruction from 3d point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 185, pp. 247–260, 2022.

[4] L. Nan and P. Wonka, "Polyfit: Polygonal surface reconstruction from point clouds," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[5] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, 2007.

[6] C. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Neural Information Processing Systems*, 2017.

[7] X. Wu, Y. Lao, L. Jiang, X. Liu, and H. Zhao, "Point transformer v2: Grouped vector attention and partition-based pooling," *ArXiv*, vol. abs/2210.05666, 2022.

[8] H. Tang, Z. Liu, S. Zhao, *et al.*, "Searching efficient 3d architectures with sparse point-voxel convolution," 2020.

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[10] Y. Liu, S. D'aronco, K. Schindler, and J. D. Wegner, "Pc2wf: 3d wireframe reconstruction from raw point clouds," *ArXiv*, vol. abs/2103.02766, 2021.