Learning to Construct 3D Building Wireframes from 3D Line Clouds

Yicheng Luo[1,2], Jing Ren[1,3]

Xuefei Zhe[1], Di Kang[1]

Yajing Xu[2], Peter Wonka[4]

Linchao Bao[1]

[1] Tencent AI Lab

[2] Beijing University of Posts and Telecommunications

[3] ETH Zurich

[4] KAUST

Abstract

Line clouds, though under-investigated in the previous work, potentially encode more compact structural information of buildings than point clouds extracted from multi-view images. In this work, we propose the *first* network to process line clouds for building wireframe abstraction. The network takes a line cloud as input , i.e., a nonstructural and unordered set of 3D line segments extracted from multi-view images, and outputs a 3D wireframe of the underlying building, which consists of a sparse set of 3D junctions connected by line segments. We observe that a *line patch*, i.e., a group of neighboring line segments, encodes sufficient contour information to predict the existence and even the 3D position of a potential junction, as well as the likelihood of connectivity between two query junctions. We therefore introduce a two-layer Line-Patch Transformer to extract junctions and connectivities from sampled line patches to form a 3D building wireframe model. We also introduce a synthetic dataset of multi-view images with ground-truth 3D wireframe. We extensively justify that our reconstructed 3D wireframe models significantly improve upon multiple baseline building reconstruction methods.The code and data can be found at https://github.com/Luo1Cheng/LC2WF.

1 Introduction

Recent advancement in photogrammetry makes it possible to obtain 3D data in city-scale from drone images. Traditional point-based methods for 3D surface reconstruction from image such as multi-view stereo [3, 9, 47] rely on accurate key point matching, which usually becomes challenging when facing texture-less surfaces (such as glass curtain) or large viewpoints changes. To tackle this challenge, line segment-based methods have been proposed as a promising solution to camera pose estimation [38, 43] and surface reconstruction [15, 30]. It is shown to be easier and more robust to extract reliable line segments than points from multi-view images, especially in the case of lacking texture [17]. Moreover, to alleviate computational costs of downstream geometry processing applications and to reduce storage cost

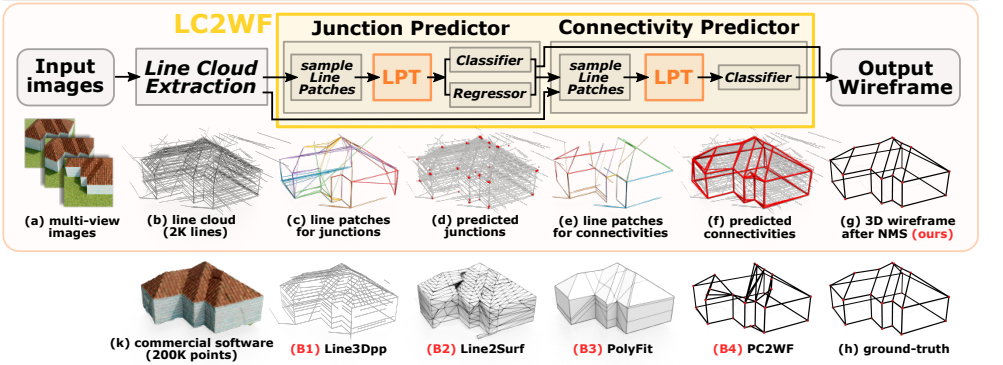


Figure 1: **Method Overview.** ***Top***: our method takes multi-view images (a) as input and outputs a high-quality 3D wireframe (g). Specifically, we first extract a line cloud (b) from the images, from which we sample line patches (c) and (e) to predict wireframe junctions (d) and connectivities (f) respectively. ***Bottom***: we compare to four baselines: Line3Dpp [17] (B1) produces abstracted line clouds from the input noisy line clouds (b). Line2Surf [30] (B2) takes (B1) as input and outputs a triangle mesh. From the point cloud (k), PolyFit [39] (B3) produces a polygonal mesh, while PC2WF [37] (B4) outputs a 3D wireframe.

of city-scale data, there is an increasing demand for urban reconstruction with *lightweight* models such as 3D wireframe models or low-resolution polygonal meshes. Besides, wireframe models are also widely-used in creating virtual cities or building information models.

To obtain lightweight building models, existing methods can be roughly categorized into two groups: (1) fit multiple simple primitives such as planes or boxes to the input point cloud to obtain a building abstraction as a polygonal mesh [8, 14, 18, 19, 28, 33, 39, 52]; (2) first construct a dense triangle mesh from the input point cloud using standard surface reconstruction techniques, e.g. [23, 27, 46]; then apply mesh simplification or decimation techniques to obtain an abstracted building model based on planar shape priors [2, 4, 29, 45]. However, both types of solutions rely on discrete operations (such as RANSAC-based fitting [39] or region-growing for mesh decimation [45]), which makes it hard to adapt existing solutions for learning-based frameworks. To close this gap, we present the *first* learning-based solution for 3D building wireframe reconstruction. We choose wireframe models as output since they are best suited for piece-wise planar objects such as urban buildings [37]. A wireframe is a graph representation of an object described by a set of junctions connected by line segments. Wireframe models have become popular for characterizing the contours of objects in both 2D [26, 57, 60, 61] and 3D [37]. However, learning a 3D building wireframe from a point or line cloud is a challenging and under-explored task, which still remains an open problem.

In this work, we propose a solution to extract the 3D building wireframe from a line cloud. As observed in [14, 19], line clouds potentially provide more structural information such as corner points and boundary edges of buildings, which are much harder to extract from point clouds. Moreover, a line cloud is more compact to characterize a building than a point cloud. For example, our method can output a reasonable building wireframe from a line cloud containing around 1K line segments. To achieve comparable result, a dense point cloud containing 50K-100K points is required for baseline methods such as PolyFit [39].

To summarize, our main *contributions* are: **(1)** a novel *learning-based* solution to reconstruct 3D building wireframe from multi-view images; **(2)** LC2FW: a transformer-based and the *first* network to process line clouds based on line patches; **(3)** an adapted synthetic dataset with annotated multi-view images and ground-truth 3D wireframe models.

2 Related Work

There is relatively limited work that focuses on building wireframe reconstruction from either multi-view images or point clouds. We mainly review related work of building reconstruction, wireframe reconstruction, as well as existing datasets for building reconstruction.

3D Point/Line Reconstruction Structure-from-Motion [1, 5, 46, 48, 49, 51] is an effective method to acquire 3D point clouds or line clouds for surface reconstruction from multi-view images. Corresponding feature points extracted from multi-view images are used to estimate camera parameters and generate 3D point clouds. Similarly, 3D line clouds can be generated from corresponding 2D line segments detected from multi-view images [17]. In our work, we focus on line clouds since the building shapes can be easily characterized by line structures.

Building Reconstruction Multiple optimization-based algorithms have been proposed for building reconstruction from point clouds. Some works [14, 33, 34, 62] use the Manhattan-world assumption to further regularize the building reconstruction. The reconstructed building meshes are usually dense and noisy, and thus different methods have been proposed for simplification or abstraction [24, 32, 54]. Primitive-based building reconstruction is another popular direction to get abstracted polygonal mesh by exploiting high-level primitives such as cubes [14, 33, 52], planes [8, 18, 19, 28, 39], or general 3D templates [35, 40] to fit input point clouds of buildings. However, building reconstruction from a 3D *line cloud* has been rarely investigated. Existing works [14, 19, 30, 50] take 3D lines into consideration to fit planes first, instead of directly extracting the building structure from the lines. Sugiura et al. [50] extend the tetrahedra-carving method to the 3D point-and-line cloud setting, while Holzmann et al. [19] use additional semantic labels from image segmentation to cluster lines for plane fitting. Langlois et al. [30] propose a RANSAC-based method to extract planes from the input line cloud, which are fused to form a watertight mesh. He et al. [14] estimate planes and corners from a line cloud for box fitting. Some other works [16, 17] provide heuristics for line cloud abstraction. In our work, we propose the first *learning-based* solution to process line clouds for 3D building wireframe reconstruction.

Wireframe Extraction As a special case of 2D edge detection [10, 11, 12, 21, 36, 41, 55, 56, 59], 2D wireframe detection from a single image [26, 57, 60, 61] is much more explored compared to the 3D wireframe reconstruction setting. A recent work PC2WF [37] proposes a CNN-based method to extract 3D wireframe models from point clouds. Zhou et al. [62] provide a method to reconstruct *partial* 3D wireframe models from a single image, from which depth maps, junction heatmaps, edge maps, and vanishing points are estimated independently for wireframe prediction. In our work, a *complete* 3D wireframe model is reconstructed from a *noisy* line cloud extracted from multi-view images.

Dataset There are multiple datasets that contain ground-truth 2D lines in images with semantically meaningful annotations [12, 31]. Here we mainly review datasets that can be potentially used for either wireframe or building reconstruction. [20] and [6] provide ground-truth 2D wireframe annotations for single images of indoor or outdoor scenes. [62] proposes a synthetic city dataset that contains 2D synthetic images with ground-truth depth and partial 3D wireframe annotations that are visible from a single view. There are also some datasets consisting of CAD models [25] or polygonal meshes [42] that can be potentially adapted to wireframes. The ABC dataset [25] is a recent dataset consisting of one million CAD models, most of which are mechanical parts. In this work, we build on [42] to create a synthetic dataset with *complete* ground-truth 3D wireframe models paired with multi-view images.

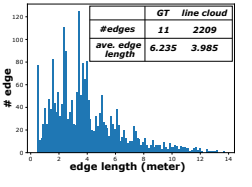
3 Background & Training Dataset

Notation Our method takes a set of multi-view images $\mathcal{I} = \{I_i\}_{i=1}^{m}$ as input, from which we extract a *line cloud* [14, 17] that consists of a group of *line segments* $\mathcal{L} = \{l_i\}_{i=1}^{N}$, where each line segment l_i is denoted by its two 3D endpoints, i.e., $l_i = (p_i, q_i), p_i, q_i \in \mathbb{R}^3$. We denote $\mathcal{G}$ as a group of line segments belonging to $\mathcal{L}$, i.e., $\mathcal{G} \subset \mathcal{L}$. The underlying 3D *wireframe* model of the line cloud $\mathcal{L}$ is denoted as $\mathcal{W} = (\mathcal{V}, \mathcal{E})$, which is defined by a set of 3D vertices (junctions) $\mathcal{V}$ and a set of edges (connectivities) $\mathcal{E}$ that connect those vertices. Specifically, we have $\mathcal{V} = \{v_i\}_{i=1}^{n_v}, v_i \in \mathbb{R}^3$, and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$.

Overall Pipeline The goal of our method is to output an accurate and clean wireframe model $\mathcal{W}$ from a set of input images $\mathcal{I}$ of a building. Our method contains the following major building blocks (see Fig. 1): (1) a *line cloud extraction* step where a dense line cloud is extracted from the input images (Sec. 3.1); (2) a *junction predictor* which classifies if there exists a junction in a group of lines and regresses the junction position accordingly (Sec. 4.3); (3) a *connectivity predictor* that instantiates edges between the predicted junctions (Sec. 4.4).

3.1 Line Cloud Extraction

There are roughly two groups of methods to extract a line cloud from multi-view images: (1) reconstruct 3D lines and estimate camera parameters simultaneously [13, 44, 58]; (2) reconstruct 3D lines with fixed camera parameters estimated from standard structure-from-motion (SfM) methods [17, 22]. In the work, we follow the latter one to reconstruct a line cloud, which is also adopted in Line3Dpp [17], the current state-of-the-art line cloud abstraction method. Specifically, the camera parameters are estimated from the multi-view images using SfM. Correspondences between the 2D line segments detected from each image (using any existing line detector) are established based on epipolar constraints, which are then used to solve 3D line segments based on the camera parameters. We use the line cloud extractor as provided in [17]. Note that, the extracted line cloud is potentially dense, noisy, and incomplete. The inset figure shows the histogram of the length of the line segments in the line cloud shown in Fig. 1 (b). Around 85% of the extracted line segments has a shorter length than the average edge length of the underlying building (Fig. 1 (h)). This suggests that the extracted line clouds contain large portion of short (and potentially noisy in orientations) line segments, which makes it challenging to extract a clean wireframe.



3.2 BuildingWF Dataset: Training Dataset

Challenges To design a data-driven solution for building wireframe reconstruction, we need large-scale datasets with ground-truth 3D wireframe annotations paired with either multi-view images or point clouds. However, it is quite challenging to obtain such datasets. Existing building datasets can be roughly categorized as follows: (1) single image with ground-truth 2D line segments [12]; (2) single image with ground-truth 2D wireframe [6, 20]; (3) single depth image with ground-truth *partial* 3D wireframe that is visible in the image [62]. On the other hand, the dataset used in PC2WF [37] are indeed in large-scale but only contain ground-truth 3D wireframe for *man-made objects* such as mechanical objects [25] and furniture.

BuildingWF Dataset In this work, we introduce a synthetic dataset with ground-truth 3D building wireframe models based on the Roof-Image dataset proposed in [42], which con-

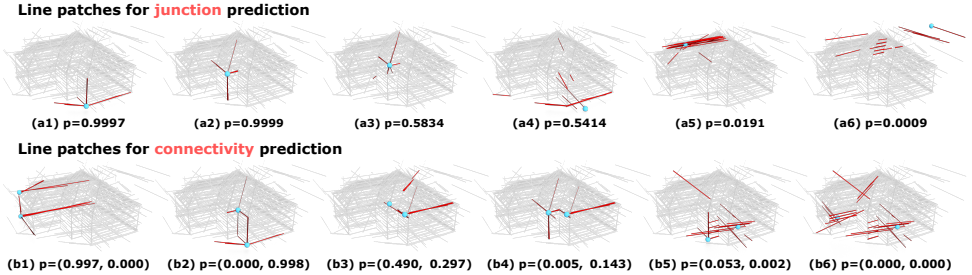


Figure 2: Example line patches (red lines) w.r.t. the sampling points (blue). ***Top***: we report the probability for each line patch to have a junction. ***Bottom***: we report two probabilities for a pair of sampling points, i.e., (1) two points are connected, and (2) two points that potentially have graph distance of 2. Note all the line patches have the same number of lines.

tains around 3.6K polygon meshes of residential buildings (denoted as $\mathcal{M}_i$). See supplementary materials for some examples. We first extract the ground-truth wireframe $\mathcal{W}^{\text{gt}} = (\mathcal{V}^{\text{gt}}, \mathcal{E}^{\text{gt}})$ from the provided building mesh $\mathcal{M}$. We then synthesize multi-view images $\mathcal{I}$ of the building $\mathcal{M}$ in Blender with synthetic textures based on the provided face labels. A 3D line cloud $\mathcal{L}$ is extracted from synthetic images $\mathcal{I}$ as mentioned in Sec. 3.1. We then use the ground-truth wireframe $\mathcal{W}^{\text{gt}}$ and camera parameters to label each 3D line segment in the line cloud $\mathcal{L}$.

Specifically, we first project the 3D ground-truth wireframe $\mathcal{W}^{\text{gt}}$ to image planes using the corresponding camera parameters to get the ground-truth 2D wireframe for *each* image $I_i \in \mathcal{I}$, which allows us to check if a 3D line $l_i \in \mathcal{L}$ is part of the wireframe $\mathcal{W}^{\text{gt}}$ or not. If the 2D line segments, that are used to reconstruct the 3D line l_i, are close enough to the ground-truth 2D wireframes, l_i will be classified as part of $\mathcal{W}^{\text{gt}}$ and be labeled as 1. For a line l_i with label 1, we further associate it with two ground-truth junction vertices that are the endpoints of the corresponding edge in $\mathcal{W}^{\text{gt}}$ that l_i belongs to. In summary, each line l_i has a 5-dimensional label: (f, i_1, d_1, i_2, d_2) where f is binary indicating if this line is part of the wireframe, i_1, i_2 are the junction index in $\mathcal{V}^{\text{gt}}$ and d_1, d_2 are the distances from l_i to the two ground-truth junctions respectively if $f = 1$. Note that the label f is used to supervise our junction classifier, and the remaining labels are used to supervise our junction regressor.

4 LC2WF: Line Cloud to Wireframe

In this section, we present the key component of our framework, LC2WF network that reconstructs a 3D wireframe from a line cloud. Before we dive into the architecture details, we would like to first motivate our design choices. The core problem is how to correctly predict the junction positions and the connectivities between junctions from a line cloud. Similar to a point cloud, a line cloud is *nonstructural*, *dense*, *noisy*, and potentially *incomplete*. However, on the other hand, the *orientation* and *length* is properly defined for line segments, which makes the neighborhood in a line cloud potentially more informative than the neighborhood in a point cloud, where only the distance between points is defined.

In the following, we first introduce *line patches* to define the neighborhood in a line cloud. We then propose our line-patch transformer[53], LPT, that is designed to process line patches to effectively extract information for junction and connectivity prediction, which are combined to produce the final 3D wireframe.

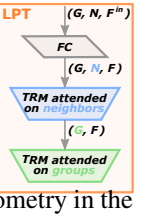
4.1 Line Patches

In our setting, a line patch is defined as a group of 3D line segments collected w.r.t. sampling points. Specifically, given an arbitrary point $x \in \mathbb{R}^3$, the corresponding line patch $\mathcal{G}(x)$ is defined as: $\mathcal{G}(x) = \{l \in \mathcal{L} \,|\, \text{dist}(x,l) \leq \varepsilon\}$, where $\text{dist}(x,l)$ measures the *point-to-line* distance between a point x and the 3D line where the line segment l lies. We can similarly define the line patch w.r.t. a pair of sampling points as $\mathcal{G}(x,y) = \mathcal{G}(x) \cup \mathcal{G}(y)$. We observe that the line patches encode sufficient information to predict junction positions and connectivity between junctions. Specifically, the line patch $\mathcal{G}(x)$ of point x can be used to estimate the probability of having a junction located around point x, while the line patch $\mathcal{G}(x,y)$ can be used to estimate the probability of having an edge connecting the point x and point y.

See Fig. 2 for an illustration: In example (a1) and (a2), the lines in the red patch have multiple dominant orientations, suggesting that the blue sampling point is indeed close to a wireframe junction. This aligns with the fact that a 3D junction is formed by the intersection of at least three planes, and the corresponding intersecting lines shape the contour of the underlying building, which would lead to dominant line clusters in images. The blue sampling point in example (a5) is located on a roof plane, where the roof texture (see Fig. 1) contains structural lines. In this case there exists only one dominant direction, which is not enough to support a junction. Example (a6) shows the line patch of an outlier point, where the lines in the patch are extremely unstructured. Similar for the examples in (b1-b6), we can see that if two sampling points are likely to be connected to each other, the corresponding line patch will reveal strong pattern (e.g., having duplicated lines) to support it. All these observations of line patches perfectly align with the properties of the wireframe models of planar objects: the junctions are formed by the intersection of planes with at least three dominant directions determined by the intersecting lines. As a comparison, other points such as corners in textures or noisy points do not have comparably strong signals.

4.2 Line-Patch Transformer (LPT)

Given a line patch $\mathcal{G}(x)$ or $\mathcal{G}(x,y)$, how can we tell if there exists a junction or an edge? We propose a line-patch transformer, LPT, to extract features from line patches, which can then be used to predict the junctions/edges. Specifically, a line patch $\mathcal{G}(x)$ can be represented as a 2D tensor (N, F^{in}), that stores N neighboring lines in $\mathcal{G}(x)$, and each line has F^{in} features including the coordinates of the two endpoints and the distance between the line and the sampling point x. We then collect G groups of line patches in a 3D tensor (G, N, F^{in}). LPT contains two transformers (see inset figure): (1) the first transformer attends to the N neighbors for each line patch, to potentially find the most prominent lines for junction predictions; (2) the second transformer attends on the G groups, to potentially attend to the junctions that are co-planar. We believe the attention mechanism inside the transformers is more effective in capturing *local* geometry information in line patches as well as the *global* geometry in the line cloud. We similarly use LPT to process line patches $\mathcal{G}(x,y)$ for connectivity prediction, where the initial features can be obtained by concatenating the features of $\mathcal{G}(x)$ and $\mathcal{G}(y)$.



4.3 Junction Predictor

We sample G points $\{x_k\}_{k=1}^{G}$ from all the line endpoints of the line cloud $\mathcal{L}$ to construct line patches for junction prediction. Specifically, we first sample a smaller set of points (around 25%) according to the endpoint density and then sample the remaining points via Farthest Point Sampling (FPS) [7]. We then obtain the corresponding line patch $\mathcal{G}(x_k)$ for

each sample x_k as discussed in Sec. 4.1. The line patches $\{\mathcal{G}(x_k)\}$ are fed into LPT to extract patch features, which are used to *classify* if there exist a junction close to x_k, and *regress* the potential junction position p_k. We then collect the predicted junctions p_k in $\mathcal{V}^{\text{pred}}$. The classifier can help to filter out junctions with a low confidence. During the training phase, we first draw samples that are close to the ground-truth junctions, then sample via density and FPS to get G sampling points for constructing the line patches. This guarantees that we draw both positive line patches (containing a junction) and negative line patches. Specifically, a line patch is considered as a negative sample if there are more than half of the line segments in the patch are labeled as noise (introduced in Sec. 3.2). The loss function for the classifier is a binary cross-entropy $E_{\text{v-clf}}$. The loss function for the regression is L_2 distance between the predicted position and the ground-truth position.

4.4 Connectivity Predictor

We first sample G pairs of predicted junctions $(p_k, q_k) \in \mathcal{V}^{\text{pred}} \times \mathcal{V}^{\text{pred}}$ w.r.t. the predicted probability. We can then construct the line patches $\{\mathcal{G}(p_k, q_k)\}$ and feed them into LPT to extract patch features, which is used to classify the junction pair (p_k, q_k) into five groups: (A) labeled as -1 if at least one of (p_k, q_k) is a false positive junction (i.e., does not belong to the underlying wireframe); (B) two vertices are true positive junctions and the pair is labeled by the *graph distance* in the underlying wireframe, i.e., (B.0) with graph distance 0 (p_k is identical to q_k), (B.1) with graph distance 1 (p_k is connected to q_k), (B.2) with graph distance 2 (p_k, q_k are adjacent to the same vertex), or (B.3) having graph distance larger than 2. The proposed fine-grained classification with 5 categories can provide more *informative* labels and more *balanced* distribution (which leads to better results as justified in Tab.6 in the supplementary) than a naive binary classification, which implicitly assumes all predicted junctions are true positive and are not redundant. Note that the edge labels can help to further prune the false positive junctions besides the probability produced by the junction classifier. During training, we sample from $\mathcal{V}^{\text{gt}} \times \mathcal{V}^{\text{gt}}$ and $\mathcal{V}^{\text{pred}} \times \mathcal{V}^{\text{pred}}$ to learn junction connectivity. A vertex from $\mathcal{V}^{\text{pred}}$ is regarded as a false positive junction if its distance to the nearest ground-truth junction is larger than a threshold ε. Any vertex pair that contains a false positive junction is labeled as -1. The rest vertex pairs is labeled according to the graph distance in the ground-truth wireframe, where for a pair of predicted junctions, we use the graph distance between their nearest ground-truth junctions. The loss function for the classifier is standard cross-entropy $E_{e\text{-clf}}$.

4.5 Implementation Details

Network Details Our *LPT architecture* includes fully-connected (FC) layers (with batch normalization and ReLU activation) and two transformer encoder layers (with layer normalization, ReLU activation, and pre-normalization). The output sizes of the FC layers are set to 64/64/128/128/256 resp. In the transformer encoder layer, the input size and the latent layer size are set to 256. The *classifier/regressor* for junction prediction, and the *classifier* for connectivity prediction similarly include FC layers, ReLU activation, and batch normalization. The output size of the FC layers are 256, 128, 64, 32, and 2/3/5 (for junction-classifier/junction-regressor/connectivity-classifier resp.). See the supplementary for more details.

Training Loss The total training loss for our complete networks is: $E_{\text{total}} = E_{v\text{-clf}} + \lambda_v E_{v\text{-reg}} + \lambda_e E_{e\text{-clf}}$, where λ_v, λ_e are balancing weights.

Post-processing For post-processing, we first use non-maximum suppression (NMS) to remove duplicated vertices and redundant edges that are close to each other. We then use the connectivity predictor to further prune the predicted junctions that tend to be false positives. Specifically, if a vertex pair is categorized to be identical (i.e., with label 0), then the junction with a lower confidence will be removed. For two vertices with similar Hamming distance in adjacency and small Euclidean distance, the vertex with a lower confidence will be removed. We also remove the isolated edges from the final wireframe.

5 Experiments

We compare different methods for building mesh/wireframe reconstruction on our BuildingWF dataset with ground-truth annotations (introduced in Sec. 3.2). We briefly introduce the baselines and the metrics for evaluation in Sec. 5.1. In Sec. 5.2 we show quantitative and qualitative results on building wireframe reconstruction. See supplementary materials for ablation study, more results and discussions. Code and data will be released.

5.1 Baselines & Evaluation Metrics

To the best of our knowledge, there is no existing baseline for reconstructing building wireframes from multi-view images directly. We therefore mainly compare to the state-of-the-art 3D line cloud abstraction method, line3Dpp [17], building reconstruction methods, Line2Surface [30] and PolyFit [39], and 3D wireframe reconstruction method PC2WF [37]. Specifically, line3Dpp [17] outputs an abstracted line cloud from a dense line cloud based on heuristics for line clustering. Line2Surface [30] is an optimization-based method that extracts planes from a line cloud via RANSAC to form a building mesh. PolyFit [39] is the state-of-the-art optimization-based method for building mesh reconstruction from a potentially noisy point cloud. PC2WF [37] is a novel learning-based method to reconstruct a 3D wireframe from a point cloud, which achieves plausible results on man-made objects such as mechanical objects and furniture. For evaluation, we follow PC2WF[37] to measure the precision and recall on both predicted junctions and wireframes, and the Wireframe Edit Distance(WED): (1) $v\text{AP}_\eta$ and $v\text{Recall}_\eta$ show the precision/recall on the predicted *junctions*. (2) $s\text{AP}_\eta$ and $s\text{Recall}_\eta$ report the *structural* quality of the predicted *wireframes*. Specifically, it checks if a predicted edge is a true positive or if a ground-truth edge is retrieved according to the distances between the edge endpoints. (3) WED reports the number of operations and the editing distances of adding/removing predicted junctions/edges that are needed to transform the graph structure of the predicted wireframe into the ground-truth wireframe.

Table 1: Baselines

Method	Input type	Input size	Output type	Runtime (sec)
line3Dpp	lines	1,388	lines	33.1
line2Surf.	lines	120	mesh	220.8
PolyFit	points	86,396	mesh	45.6
PC2WF	points	86,396	wireframe	31.7
Ours	lines	1,388	wireframe	0.9

5.2 Results and Comparisons

We compare to baseline methods on 757 test buildings. The line clouds (for line3Dpp, line2Surf, and our method) and the point clouds (for PolyFit and PC2WF) are extracted using the *same* camera parameters. Note that we use a *commercial software* to extract high-quality point clouds (See Fig. 1 (k) and Fig. 3 (a) for some examples). Moreover, to make a fair comparison to line2Surface [30] and PolyFit [39], we post-process the output meshes into wireframes by merging co-planar faces and parallel adjacent edges, removing interior edges and isolated vertices, etc. For PC2WF we use the provided NMS for post-processing. We report the evaluations on the results after post-processing (Tab. 2).

(a) Precision/Recall of the predicted **junctions** (vAP/vRecall) and the predicted **wireframe** models (sAP/sRecall) on results **after post-processing**. We highlight the **best** and the **second best** results.

Method	$v\text{AP}_\eta/v\text{Recall}_\eta$ (%)				$s\text{AP}_\eta/s\text{Recall}_\eta$ (%)			
	$\eta = 0.15$	$\eta = 0.25$	$\eta = 0.35$	avg.	$\eta = 0.25$	$\eta = 0.35$	$\eta = 0.50$	avg.
line2Surf.	26.7/83.9	27.4/85.8	27.6/86.6	27.2/85.4	24.2/58.8	25.1/61.0	25.8/62.6	25.0/60.8
PolyFit	52.1/70.8	62.0/84.3	64.3/87.4	59.5/80.8	45.5/53.8	58.7/69.5	65.5/77.5	56.6/66.9
PC2WF	11.9/26.8	43.2/54.3	58.5/65.2	37.9/48.8	0.84/7.61	7.68/23.3	23.0/40.4	10.5/23.8
Ours	**91.3/92.2**	**93.4/93.9**	**94.4/94.8**	**93.0/93.6**	**76.8/84.7**	**80.6/87.1**	**83.9/89.5**	**80.4/87.1**

(b) **Wireframe Edit Distance** (WED) of the reconstructed **wireframes**. We report the number of operations (Num) and the editing distances in meters (Dist).

Method	(WED) **+vertex**		(WED) **+edge**		(WED) **-edge**		(WED) **Total**	
	Num.	Dist	Num	Dist	Num	Dist	Num	Dist
line2Surf.	1.012	13.78	6.223	35.76	9.427	48.77	16.66	98.31
PolyFit	1.681	3.170	4.811	21.41	0.969	5.285	7.463	29.86
PC2WF	5.216	3.445	17.01	87.94	4.622	33.38	26.84	124.8
Ours	0.766	1.810	2.880	11.49	1.655	14.03	**5.301**	**27.33**

Table 2: Precision/Recall and Wireframe Edit Distance results after post-processing

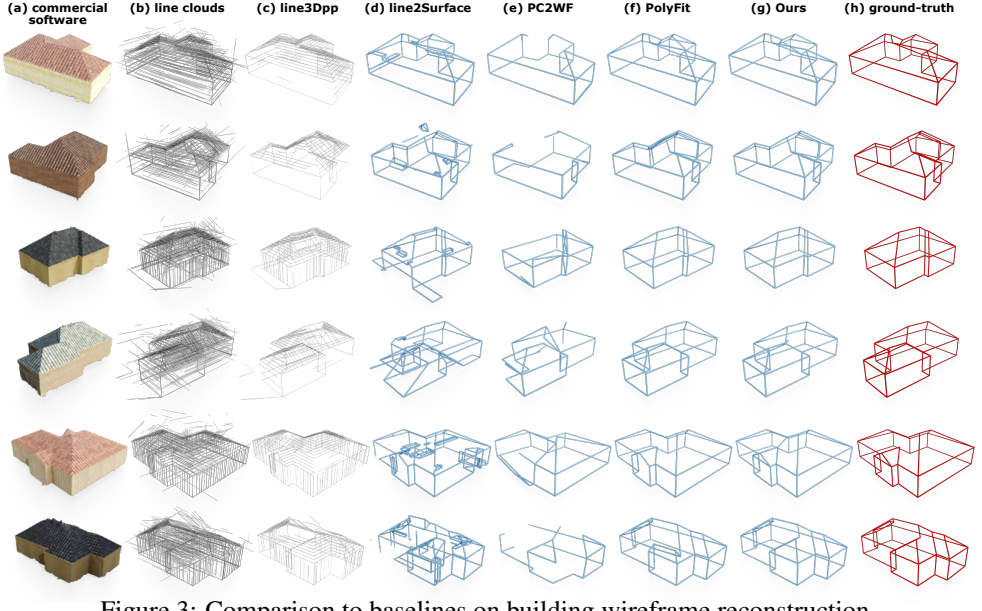


Figure 3: Comparison to baselines on building wireframe reconstruction.

Tab. 2a shows a fair comparison to line2Surf and PolyFit, where all the output meshes are post-processed into cleaner wireframes (see Fig. 3 for some examples of the post-processed results; see Fig. 1 (B2,B3) and supplementary materials for examples of direct outputs from different methods). we report the number of vertex/edges, and the precision/recall of the predicted junctions/wireframes on the results after post-processing. We do not compare to Line3Dpp in this case since it outputs a nonstructural line cloud. The results show that our method outperforms all the baselines on building wireframe reconstruction.

Note that, PolyFit shows visually comparable results to ours in Fig. 3, but its accuracy is much lower as shown in Tab. 2a. The reason is that in PolyFit, the junction positions are determined by the intersections among the *estimated* planes, and their L_2 distance to the ground-truth junctions can be large even though the overall shape looks alike. At the same time, redundant faces will be generated to satisfy the watertight hard constraint in PolyFit.

Tab. 2b shows the wireframe edit distance for different methods, where our method achieves the least number of editing operations and the smallest editing distances. Fig. 3 shows a qualitative comparison of different baselines. We can see that line3Dpp can indeed provide more abstracted line clouds, but they are still far from clean wireframe models. Line2Surf can robustly recover planes from the line cloud (from Line3Dpp), but it is not

robust to the noise. PC2WF is trained on point clouds from mechanical objects and furniture, which are likely to have a large domain gap to the rooftop structures. Therefore, the wireframes constructed by PC2WF can only recover the walls in building point clouds. Moreover, we also observe that the point clouds stemming from our scenes are more noisy (though they are accurate enough) than the point clouds that PC2WF is trained on. This can also lead to the less satisfactory results that PC2WF obtains. PolyFit is a powerful method for building reconstruction that is robust to noisy point clouds. However, PolyFit can be computationally costly when the input point cloud is too dense since the algorithm involves integer linear programming. As a comparison, our method can achieve visually comparable and quantitatively better results in a much more efficient way. For example, on average it takes our method 0.9s to infer a building wireframe while it takes PolyFit 45.6s to optimize a building mesh (see Tab. 1). We show more results of our reconstructed wireframes in the supplementary materials.

Fig. 4 shows some preliminary but reasonable results on two real-world noisy scans without finetuning. One of the main challenges of our task is the lack of large-scale real-world buildings paired with clean and complete wireframes (e.g., manually created by artists). Adapting existing datasets is almost as hard as designing a new one as discussed in Sec.3.2. Inspired by PC2WF where synthetic point clouds are generated for training and testing, we therefore justify our LC2WF on synthetic dataset. We believe our LC2WF can be fine-tuned on future real-world datasets to get better performance.

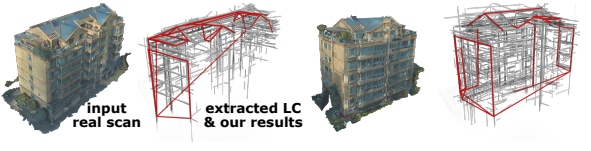


Figure 4: **Two real-data examples (without finetuning)**: we overlay our reconstructed wireframe (*red*) on top of the extracted line cloud (*gray*).

6 Conclusion, Limitation & Future Work

In this work, we present the first learning-based solution for building wireframe reconstruction from line clouds, which can be efficiently extracted from multi-view images. We construct a synthetic dataset, BuildingWF, containing multi-view images of 3.6K buildings and the corresponding ground-truth wireframe models. The key component of our method is a Line-Patch Transformer which can be used for junction and connectivity prediction from line patches, a group of neighboring line segments that potentially encode the contour information of the underlying building. Our method outperforms multiple state-of-the-art building reconstruction methods on both accuracy and efficiency.

Our method still has some limitations. For example, we assume the input multi-view images cover the overall region of the underlying buildings, and we expect to extract a building wireframe from a noisy but relatively complete line clouds. Therefore, no prior knowledge or extra regularizers are investigated to complete a wireframe from a partial line cloud with large missing regions. We would like to leave it as future work to investigate wireframe reconstruction from partial line clouds. Moreover, in this work we do not investigate how to convert a wireframe into a watertight mesh. We believe it would be interesting to try to learn face information from line patches as well, which we leave as future work.

Acknowledgments The authors thank the anonymous reviewers for their valuable comments and suggestions. We would like to acknowledge support from the SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence. We thank *Zhenbang He* and *Yue Qian* for their helpful suggestions.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[2] Jean-Philippe Bauchet and Florent Lafarge. Kinetic shape reconstruction. *ACM Transactions on Graphics (TOG)*, 39(5):1–14, 2020.

[3] Neill DF Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. pages 766–779, 2008.

[4] Anne-Laure Chauve, Patrick Labatut, and Jean-Philippe Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. pages 1261–1268, 2010.

[5] David Crandall, Andrew Owens, Noah Snavely, and Dan Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. pages 3001–3008, 2011.

[6] Patrick Denis, James H. Elder, and Francisco J. Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *European Conference on Computer Vision (ECCV)*, pages 197–210, 2008.

[7] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997.

[8] Hao Fang and Florent Lafarge. Connect-and-Slice: an hybrid approach for reconstructing 3D objects. In *Computer Vision and Pattern Recognition (CVPR)*, Seattle, US, 2020.

[9] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(8): 1362–1376, 2009.

[10] Rafael Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32:722–32, 04 2010.

[11] Geonmo Gu, Byungsoo Ko, SeoungHyun Go, Sung-Hyun Lee, Jingeun Lee, and Minchul Shin. Towards real-time and light-weight line segment detection, 2021.

[12] Qi Han, Kai Zhao, Jun Xu, and Ming-Ming Cheng. Deep hough transform for semantic line detection. In *European Conference on Computer Vision (ECCV)*, pages 750–766, 2020.

[13] Yijia He, Ji Zhao, Yue Guo, Wenhao He, and Kui Yuan. Pl-vio: Tightly-coupled monocular visual–inertial odometry using point and line features. *Sensors*, 18(4):1159, 2018.

[14] Zhenbang He, Yunhai Wang, and Zhanglin Cheng. Manhattan-world urban building reconstruction by fitting cubes. *Computer Graphics Forum*, 40(7):289–300, 2021.

[15] Manuel Hofer, Andreas Wendel, and Horst Bischof. Incremental line-based 3d reconstruction using geometric constraints. 2013.

[16] Manuel Hofer, Michael Maurer, and Horst Bischof. Line3d: Efficient 3d scene abstraction for the built environment. In *German Conference on Pattern Recognition*, pages 237–248. Springer, 2015.

[17] Manuel Hofer, Michael Maurer, and Horst Bischof. Efficient 3d scene abstraction using line segments. *Computer Vision and Image Understanding*, 157:167–178, 2017. ISSN 1077-3142.

[18] Thomas Holzmann, Martin R Oswald, Marc Pollefeys, Friedrich Fraundorfer, and Horst Bischof. Plane-based surface regularization for urban 3d construction. In *British Machine Vision Conference (BMVC)*, pages 1–9, 2017.

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

[20] Kun Huang, Yifan Wang, Zihan Zhou, Tianjiao Ding, Shenghua Gao, and Yi Ma. Learning to parse wireframes in images of man-made environments. In *Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[21] Siyu Huang, Fangbo Qin, Pengfei Xiong, Ning Ding, Yijia He, and Xiao Liu. TP-LSD: tri-points based line segment detector. In *European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science. Springer, 2020.

[22] Arjun Jain, Christian Kurz, Thorsten Thormählen, and Hans-Peter Seidel. Exploiting global connectivity constraints for reconstruction of 3d line segments from images. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1586–1593. IEEE, 2010.

[23] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Symposium on Geometry processing*, volume 7, 2006.

[24] Tom Kelly and Niloy J. Mitra. Simplifying urban data fusion with bigsur, 2018.

[25] 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 *Computer Vision and Pattern Recognition (CVPR)*, pages 9601–9611, 2019.

[26] Naejin Kong, Kiwoong Park, and Harshith Goka. Hole-robust wireframe detection, 2021.

[27] Patrick Labatut, J-P Pons, and Renaud Keriven. Robust and efficient surface reconstruction from range data. 28(8):2275–2290, 2009.

[28] Florent Lafarge and Pierre Alliez. Surface reconstruction through point set structuring. In *Computer Graphics Forum*, volume 32, pages 225–234. Wiley Online Library, 2013.

[29] Florent Lafarge and Clément Mallet. Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. *International journal of computer vision*, 99(1):69–85, 2012.

[30] Pierre-Alain Langlois, Alexandre Boulch, and Renaud Marlet. Surface Reconstruction from 3D Line Segments. In *International Conference on 3D Vision (3DV)*, pages 553–563, Québec City, Canada, September 2019. IEEE. doi: 10.1109/3DV.2019.00067.

[31] Jun-Tae Lee, Han-Ul Kim, Chul Lee, and Chang-Su Kim. Semantic line detection and its applications. In *International Conference on Computer Vision (ICCV)*, 2017.

[32] Minglei Li and Liangliang Nan. Feature-preserving 3d mesh simplification for urban buildings. *ISPRS Journal of Photogrammetry and Remote Sensing*, 173:135–150, 2021.

[33] Minglei Li, Liangliang Nan, and Shaochuang Liu. Fitting boxes to manhattan scenes using linear integer programming. *International Journal of Digital Earth*, 9(8):806–817, 2016.

[34] Minglei Li, Peter Wonka, and Liangliang Nan. Manhattan-world urban reconstruction from point clouds. In *European Conference on Computer Vision (ECCV)*, pages 54–69. Springer, 2016.

[35] 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.

[36] Yancong Lin, Silvia L. Pintea, and Jan C. van Gemert. Deep hough-transform line priors. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *European Conference on Computer Vision (ECCV)*, 2020.

[37] Yujia Liu, Stefano D'Aronco, Konrad Schindler, and Jan Dirk Wegner. Pc2wf: 3d wireframe reconstruction from raw point clouds. In *International Conference on Learning Representations (ICLR)*, 2020.

[38] Pedro Miraldo, Tiago Dias, and Srikumar Ramalingam. A minimal closed-form solution for multi-perspective pose estimation using points and lines. pages 474–490, 2018.

[39] Liangliang Nan and Peter Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *International Conference on Computer Vision (ICCV)*, pages 2353–2361, 2017.

[40] Liangliang Nan, Caigui Jiang, Bernard Ghanem, and Peter Wonka. Template assembly for detailed urban reconstruction. In *Computer Graphics Forum*, volume 34, pages 217–228. Wiley Online Library, 2015.

[41] Rémi Pautrat*, Juan-Ting Lin*, Viktor Larsson, Martin R. Oswald, and Marc Pollefeys. Sold2: Self-supervised occlusion-aware line description and detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[42] Jing Ren, Biao Zhang, Bojian Wu, Jianqiang Huang, Lubin Fan, Maks Ovsjanikov, and Peter Wonka. Intuitive and efficient roof modeling for reconstruction and synthesis. *ACM Transactions on Graphics (TOG)*, 40(6):1–17, 2021.

[43] Yohann Salaün, Renaud Marlet, and Pascal Monasse. Robust and accurate line-and/or point-based pose estimation without manhattan assumptions. pages 801–818, 2016.

[44] Yohann Salaün, Renaud Marlet, and Pascal Monasse. Line-based robust sfm with little image overlap. In *2017 International Conference on 3D Vision (3DV)*, pages 195–204. IEEE, 2017.

[45] David Salinas, Florent Lafarge, and Pierre Alliez. Structure-aware mesh decimation. In *Computer Graphics Forum*, volume 34, pages 211–227. Wiley Online Library, 2015.

[46] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. pages 4104–4113, 2016.

[47] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. pages 501–518, 2016.

[48] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM Transactions on Graphics (TOG)*, pages 835–846. 2006.

[49] Noah Snavely, Steven M Seitz, and Richard Szeliski. Skeletal graphs for efficient structure from motion. pages 1–8, 2008.

[50] Takayuki Sugiura, Akihiko Torii, and Masatoshi Okutomi. 3d surface reconstruction from point-and-line cloud. In *International Conference on 3D Vision (3DV)*, pages 264–272. IEEE, 2015.

[51] Chris Sweeney, Torsten Sattler, Tobias Hollerer, Matthew Turk, and Marc Pollefeys. Optimizing the viewing graph for structure-from-motion. pages 801–809, 2015.

[52] Carlos A Vanegas, Daniel G Aliaga, and Bedrich Benes. Automatic extraction of manhattan-world building masses from 3d laser range scans. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1627–1637, 2012.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[54] Yannick Verdie, Florent Lafarge, and Pierre Alliez. Lod generation for urban scenes. *ACM Transactions on Graphics (TOG)*, 34(ARTICLE):30, 2015.

[55] Yifan Xu, Weijian Xu, David Cheung, and Zhuowen Tu. Line segment detection using transformers without edges. In *Computer Vision and Pattern Recognition (CVPR)*, pages 4257–4266, 2021.

[56] Nan Xue, Song Bai, Fudong Wang, Gui-Song Xia, Tianfu Wu, and Liangpei Zhang. Learning attraction field representation for robust line segment detection. In *Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[57] Nan Xue, Tianfu Wu, Song Bai, Fudong Wang, Gui-Song Xia, Liangpei Zhang, and Philip H.S. Torr. Holistically-attracted wireframe parsing. In *Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[58] Guoxuan Zhang, Jin Han Lee, Jongwoo Lim, and Il Hong Suh. Building a 3-d line-based map using stereo slam. *IEEE Transactions on Robotics*, 31(6):1364–1377, 2015. doi: 10.1109/TRO.2015.2489498.

[59] Haotian Zhang, Yicheng Luo, Fangbo Qin, Yijia He, and Xiao Liu. Elsd: Efficient line segment detector and descriptor, 2021.

[60] Ziheng Zhang, Zhengxin Li, Ning Bi, Jia Zheng, Jinlei Wang, Kun Huang, Weixin Luo, Yanyu Xu, and Shenghua Gao. Ppgnet: Learning point-pair graph for line segment detection. In *Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[61] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *International Conference on Computer Vision (ICCV)*, October 2019.

[62] Yichao Zhou, Haozhi Qi, Yuexiang Zhai, Qi Sun, Zhili Chen, Li-Yi Wei, and Yi Ma. Learning to reconstruct 3d manhattan wireframes from a single image. In *International Conference on Computer Vision (ICCV)*, October 2019.