3D deep learning enhanced void-growing approach in creating geometric digital twins of buildings

Yuandong Pan PhD Student, Chair of Computational Modeling and Simulation, Technical University of Munich, Germany (yuandong.pan@tum.de)
Alexander Braun Chair of Computational Modeling and Simulation, Technical University of Munich, Germany
André Borrmann Professor, Chair of Computational Modeling and Simulation Technical University of Munich, Germany
Ioannis Brilakis Laing O'Rourke Professor of Construction Engineering, Cambridge University, the United Kingdom

Abstract

The challenge this paper addresses is how to automatically generate geometric digital twins of the indoor environment of buildings. Unlike most previous research that starts with detecting planes in the point cloud and only considers the geometric information, the proposed "void-growing" approach is a full-automatic approach that starts with detecting void space inside rooms, considering geometric information, as well as semantic information predicted from deep learning. Then based on the detected room spaces, structural elements, as well as doors and windows, are extracted. The method can work in (1) rooms with complex structures like U-shape and L-shape, (2) rooms with different ceiling heights, and (3) rooms under a high occlusion level. Compared with previous studies that mainly only use geometric information, the approach also focuses on how to select useful information predicted by deep learning. This study used existing state-of-the-art deep learning architecture for the segmentation task in the proposed approach. By taking useful semantic information into consideration, the proposed approach performs better in creating geometric digital twins of buildings.

Keywords: Digital twin, deep learning, point cloud, 3D reconstruction

1. Introduction

In Architecture, Engineering, Construction, and Facilities Management (AEC&FM) sectors, digital twins have been entering the conversation as they can continuously offer substantial value to all associated stakeholders in various aspects. A digital twin as a regular-updated digital replica of a physical built asset in a built environment (Brilakis *et al.*, 2019). This paper focuses on creating a geometric digital twin of buildings. A geometric digital twin means a digital representation that contains the current geometric information of physical assets is referred.

However, despite a digital twin being valuable for building management and maintenance, only a few existing buildings have reliable a basic digital twin which only contains structural elements and reflects their current as-is state. There are mainly two reasons for this situation. The first one is that many buildings were constructed decades ago. There were no authoring tools to construct a 3D digital model or the concept of digital twin when they were built. The other reason is that even though some new buildings have a digital design model, this model was not updated when the asset was modified through its lifecycle. Therefore, most buildings do not have any valuable digital twin.

Existing capturing technologies such as laser scanning and photogrammetry make it possible to efficiently collect point clouds that contain geometric information about the as-is state in the built environment. Compared with acquisition time, it is much longer to extract and reconstruct a 3D model that contains components like columns, slabs, and walls from the input point cloud. Some leading 3D CAD vendors (like Autodesk, Bentley, and ClearEdge3D) have developed software that has a variety of 3D modeling features, which enable modeling from point cloud data. According to the authors' experience, it took around two hours to collect the point cloud in a working area of approximately 20 rooms by using a NAVVIS scanner (https://www.navvis.com/). But one junior modeller spent more than 80 hours modelling basic elements (ceilings, floors, walls, doors, windows, stairs) in this area using Autodesk Revit software. In the field of bridge modelling, Lu and Brilakis (2017) capture ten bridges by the laser scanner and counte the as-is modelling time by using Autodesk Revit as the modelling software. They reported that the average as-is modeling time is around 28 hours, while the corresponding capturing time is 2.82 hours. Agapaki *et al.* (2018) state that 64% of man-hour savings could be achieved by using the state-of-the-art, semi-automated commercial software EdgeWise (https://www.clearedge3d.com/edgewise/). But 2,382 manual labor hours are still needed to model an example of a small petrochemical plant with 240,687 objects and 53,834 pipes. In summary, despite the fact that some commercial software solutions can reduce the modeling time, it is still much higher than the time for capturing, which makes the cost and effort

to manually generate a digital model exceed its benefits. Therefore, researchers are trying to automate the process of digital twinning in built environments in order to reduce human effort.

Artificial intelligence (AI) has been penetrating the AEC domain in recent years and provides possibilities to reduce human effort in the process of creating digital twins from a different view. Especially deep neural networks (DNN) (Krizhevsky *et al.*, 2012) provide an efficient solution to achieve point cloud segmentation from a different perspective. However, the question of how exactly to use AI techniques and their output results to accelerate and improve the process is still an ongoing topic.

The contribution of this paper is summarised as follows: a) an automatic pipeline based on the void-growing method that extracts void room space inside rooms first is presented, which is first proposed in the previous conference paper (Pan *et al.*, 2021); b) state-of-the-art deep learning technologies are applied in the initial void-growing method to improve the performance of extracting walls, ceilings, and floors. c) Semantic information from deep learning is used to extract doors and windows. However, the window and door recognition by point cloud deep learning is not improved when adding more training data from different sources. The final geometric digital twin includes point clusters with annotated semantic information, which shows the detailed information of captured surfaces and a simplified mesh model of the structural elements. As the detailed geometric information provides information that could be missed during the process of creating a simplified mesh, the final output can be applied in use cases that require detailed information, like asset condition monitoring.

This paper is organised as follows: the research background including state of the art is reviewed in Section 2; the proposed pipeline is introduced in Section 3 in detail; experiments and implementation details are shown in Section 4; conclusions and future work are discussed in Section 5.

2. Research background

In this chapter, the state of the art in the process of creating digital twins of a building is discussed in Section 2.1. As 3D deep learning is fused into the proposed method, the recent research on deep learning in the point cloud is introduced in Section 2.2.

2.1. Process of creating a digital twin

In recent years, many approaches have been proposed to automate the process of reconstructing 3D models from the point cloud. However, this topic is still solved only partially, and most previous work is designed for specific types of buildings and is restricted to reconstructing specific kinds of objects. The discussion of the state of the art is organised in three different room categories as follows.

2.1.1. Single room reconstruction

Some approaches are proposed to reconstruct individual rooms. Budroni and Boehm (2010) use the plane-sweeping method to segment horizontal and vertical structures. Then the positions of elements like floor, ceilings, and walls are detected. Adan and Huber (2011) use a histogram to determine surfaces of ceilings and floors and then use Hough transform to detect wall surfaces. Xiong *et al.* (2013) apply a region growing method to form different point patches and implement a stacked learning approach to classify the detected patches. As a digital twin is more valuable in larger facilities than in a single room, and apparently, the process is much more complex for larger buildings as well, many researchers focus on the reconstruction process for buildings with multiple rooms.

2.1.2. Multiple rooms reconstruction

Some research focuses on reconstructing digital models from the outdoor environment (Chen *et al.*, 2020), which is not within the scope of this paper. There are many approaches that aim to reconstruct multiple rooms in indoor environments of buildings.

Sanchez and Zakhor (2012) propose an approach that employs principle component analysis and Random sample consensus (RANSAC) to detect large-scale architectural structures, such as ceilings and floors, as well as relatively small-scale structures like staircases. In this

approach, all points are classified into door, ceiling, wall, and remaining points categories. Monszpart *et al.* (2015) extract planar structures in a point cloud that follows regularity constraints and then optimise the plane arrangement. Authors use this approach to extract planes in many scenes, such as urban scenes, building exteriors, and building interiors. Oesau *et al.* (2013) use the horizontally-slicing method and volumetric-cell labelling approach to reconstruct watertight surface meshes. The binary labelling of the volumetric cells is formulated as energy minimisation and solved by the graph-cut method.

Xiao and Furukawa (2014) propose a method named "inverse constructive solid geometry" that detects planar surfaces and then fits the cuboid primitives to the point cloud. Mura *et al.* (2014) use an approach based on the diffusion process of space partitioning. They extract patches using a region growing process based on the normal deviation and plane offset. Wang *et al.* (2017) use principal component analysis to estimate the normal for each point, RANSAC to fit linear primitives, and graph-cut to identify walls. The proposed hierarchical clustering method can identify rooms without knowing the number of rooms in advance. Murali *et al.* (2017) use the RANSAC-based method to detect vertical and horizontal planes. Then they create a wall graph and fit cuboids to rooms. Ochmann *et al.* (2016) propose a method that explicitly represents buildings as interconnected volumetric wall elements. They determine an optimal room and wall layout by graph-cut based multi-label energy minimisation.

Some approaches use prior knowledge explicitly to reconstruct walls and rooms. Stambler and Huber (2014) propose the concept of enclosure reasoning that premises rooms are cycles of walls enclosing free interior space. They use the region growing method to segment the point clouds and simulated annealing to optimise rooms and walls. Tran *et al.* (2019) use the shape grammar approach to model indoor environments. They generate 3D parametric models by placing cuboids into point clouds, and classifying them into elements and spaces. The wall candidates are obtained from pairs of adjacent peaks in the histogram of point coordinates. Truong-Hong and Lindenbergh (2022) propose a method to extract elements like floors, ceiling slabs, columns, and beams by rough extracting the candidate points of the component and then filtering the surface points of the components from the construction site.

2.1.3. Multi-storey reconstruction

Some studies target multi-storey buildings. Macher *et al.* (2017) propose a semi-automatic reconstruction approach for multi-storey buildings. The automatic part of their work is to segment the input data into sub-spaces (rooms) and planes. After segmentation, the 3D geometry of the elements is translated to the BIM file manually. Ochmann *et al.* (2019) reconstruct volumetric models of walls and slabs in multi-storey buildings. Planes are detected first by employing RANSAC first, and then the detected planes are classified as horizontal slab surfaces and vertical wall surfaces. A 3D plane arrangement is constructed by intersecting all planes, yielding over segmented cells. Subsequently, the integer linear programming approach is used to find an optimal label for all cells.

2.1.4. Opening detection

When collecting data with laser scanner in indoor environment, doors are usually opened. The transparent glasses of windows cannot be captured by a laser scanner. So, there are usually door and window openings in existence the in laser-scanned point clouds of indoor environments. Some methods have been proposed to detect the openings in point clouds. Mayer and Reznik (2005) interpret the building facade from images and detect windows by predefined shape. Pu and Vosselman (2007) extract windows from terrestrial point clouds of building facade by grouping points in planar segments and then fitting rectangles to the boundary of hole points. In their following research (Pu and Vosselman, 2009), prior knowledge such as the wall, opening and roof features' sizes, positions, orientations, and topology is used to recognise these objects. Ripperda (2008) reconstruct the structure of facades by a facade grammar and a reversible jump Markov Chain Monte Carlo process. Truong-Hong *et al.* (2013) propose a sample method to categorise points as boundary or interior points based on an angle criterion. Holes caused by occlusions are distinguished if they do not fit the predefined opening dimensions. Haghighatgou *et al.* (2022) propose an approach that investigates the house facade where openings have different shapes, sizes, and non-symmetrical positions. But the occlusion holes having characteristics similar to window opening still cause problems in the detection.

In summary, most approaches are not full-automatic, which means they still require human effort in the process of reconstruction, And most of them are only relying on geometric information, which makes them requires predefined shape as prior knowledge to recognise objects. In addition, the performance, when applied in the point cloud with a high occlusion level, would decrease because of geometric information deletion caused by the occlusion of furniture.

2.2. Deep learning in point cloud

Deep learning approaches have been widely applied to predict unknown labels based on a set of features present in the input data. While predicting unknown labels is the classification problem in deep learning (Goodfellow *et al.*, 2016), point cloud segmentation can also be seen as a classification problem for each point in the point cloud. Therefore, many networks are designed to solve the classification problem and segmentation problem in one architecture framework when processing point clouds.

Some networks work on voxel structure which requires point cloud voxelisation first. Prokhorov (2010) uses 3D Convolutional Neural Networks (CNNs) to solve a binary classification problem. A more general network with 3D CNN architecture, Maturana and Scherer (2015) propose VoxNet to detect classes of objects for 3D point cloud data. A volumetric grid that can represent the spatial occupancy is calculated first and then applied to 3D CNNs.

Unlike methods that require voxelised input, some networks process points directly. The neural network architecture, PointNet, Qi *et al.* (2016), is the first network that is proposed for points in the point cloud. It takes point clouds as input directly and outputs labels for the entire input or labels for each point. PointNet processes the feature of individual points independently and extracts global features of the entire point set. An improved network named PointNet++ based on PointNet considering spatial information of point sets is proposed Qi *et al.* (2017). In PointNet++, the set of points is grouped into overlapping local regions by the distance. Local features are extracted by capturing geometric structures from small neighbourhoods, and then grouped into larger units to compute higher-level features.

Other approaches inspired by PointNet, which process point cloud directly, are proposed. Dynamic graph CNN (Wang *et al.*, 2018), differing from networks working on individual points like PointNet, constructs local geometric structures by a local neighbourhood graph and applies convolution-like operations to the graph. Compared with graph CNNs, the graph in this model is not fixed but is dynamically updated after each layer of the network so that convolutions are implemented not only in the geometric neighbourhood but also in the semantic one. Li *et al.* (2018) propose a novel convolution operator called χ- operator to extract features from the point cloud. Thomas *et al.* (2019) present another novel way to apply convolution operation in the point cloud, which is called kernel point convolution. In this method, the convolution operation is performed in kernel points and points close to them. Hu *et al.* (2020) introduce a novel local feature aggregation module, which works faster in large-scale point clouds. Fan *et al.* (2021) design a module that learns spatial contextual features from the point cloud and embeds it in an encoder-decoder architecture. Qiu *et al.* (2021) augment the local context of points and interpret the distinctness of the points from multiple resolutions to achieve semantic segmentation. Some networks adapt the transformer architecture (Vaswani *et al.*, 2017) from natural language processing and apply the idea in point cloud segmentation (Guo *et al.*, 2021; Zhao *et al.*, 2021; Engel *et al.*, 2021). These networks show the transformer architecture is also powerful in point cloud processing. Perez-Perez *et al.* (2021) design a network for the Scan-to-BIM process to segment the structural, architectural, and mechanical components.

For point cloud segmentation of buildings, the S3DIS dataset (Armeni *et al.*, 2016) that contains point clouds of six areas of over $6,000m^2$, is widely used to evaluate the performance of different architectures. The performance evaluation of some above-mentioned architectures on the S3DIS dataset is compared by mean Intersection over Union (mIoU) in Table 1. It is obvious that most networks perform well for classes like ceiling, floor, and wall, while the performance for other categories is not at the same level. For example, the predicted ceiling class (mIoU around 90%) is more reliable than that of a window class (mIoU around 60%). Therefore, how to define and extract the useful information computed from neural networks and apply it in the reconstruction process is still an ongoing topic.

model	mIoU	ceil.	wall	floor	wind.	door	colu.	beam	chair	table	book.	sofa	board	clut.
PointNet (Qi *et al.*, 2016)	47.6	88.0	69.3	88.7	47.5	51.6	23.1	42.4	42.0	54.1	38.2	9.6	29.4	35.2
SPG (Landrieu and Simonovsky, 2018)	62.1	89.9	76.4	95.1	55.3	68.4	47.1	62.8	73.5	69.2	63.2	45.9	8.7	52.9
RSNet (Huang *et al.*, 2018)	56.5	92.8	92.5	78.6	51.6	68.1	34.4	32.8	60.1	59.7	50.2	16.4	44.9	52.0
PointCNN (Li *et al.*, 2018)	65.4	94.8	75.8	97.3	58.4	57.2	51.7	63.3	71.6	69.1	39.1	61.2	52.2	58.6
KPConv (Thomas *et al.*, 2019)	70.6	93.6	83.1	92.4	66.1	76.6	54.3	63.9	57.8	64.0	69.3	74.9	61.3	60.3
PointTransformer (Zhao *et al.*, 2020)	70.4	94.3	84.7	97.5	66.1	78.2	58.1	55.6	74.1	77.6	71.2	67.3	65.7	64.8

Table 1. Segmentation mIoUs (%) on S3DIS dataset (evaluated with 6-fold cross-validation)

2.3. Research gaps

In indoor environments, occlusion caused by furniture occurs quite often, which leads to geometry missing in point clouds. The majority of methods are sensitive to the occlusion because they only use geometric information in point clouds. They first detect surfaces of elements in point clouds utilizing Hough transform (Adan and Huber, 2011), RANSAC (Wang *et al.*, 2017; Ochmann *et al.*, 2019; Murali *et al.*, 2017), or region growing (Xiong *et al.*, 2013; Mura *et al.*, 2014; Stambler and Huber, 2014). When occlusions occur in the environment, the performance of these methods declines, especially when there are complex rooms (like L-shape rooms and U-shape rooms) in the point cloud. The main reason is that starting with detecting surfaces solely based on geometric information makes it hard to distinguish a large surface of furniture like a cupboard from a relatively small wall surface. In the void-growing approach, the aim is to detect the furthest surface in all directions by checking geometric information as well as semantic information in the detection process. Furthermore, it can also reconstruct rooms with different ceiling heights, which is usually caused by suspended ceilings and is quite common in the building. Most previous approaches do not try to reconstruct these rooms from the point cloud.

With regard to applying 3D deep learning in the process of creating digital twins, it is not easy and clear how to extract useful and precise information from imprecise and redundant information predicted by neural networks. In the proposed approach, the void-growing approach is enhanced by 3D deep learning and considers semantic information as a potential stopping condition.

3. Methodology

3.1. Overview

Instead of detecting surfaces in the point cloud first, the proposed void-growing approach aims to find the largest void space volume inside each room. Based on the found void volumes, structural elements are reconstructed. Moreover, the detection process is improved, especially when adding semantic information from 3D deep learning. The overall workflow of the proposed approach is illustrated in Figure 1. In this paper, the authors propose a novel pipeline to fully automate the process of creating the geometric digital twin of a building, applying point cloud segmentation by deep learning and extending a void-growing method proposed in the previous conference paper (Pan *et al.*, 2021) to extract 3D models from the point cloud.

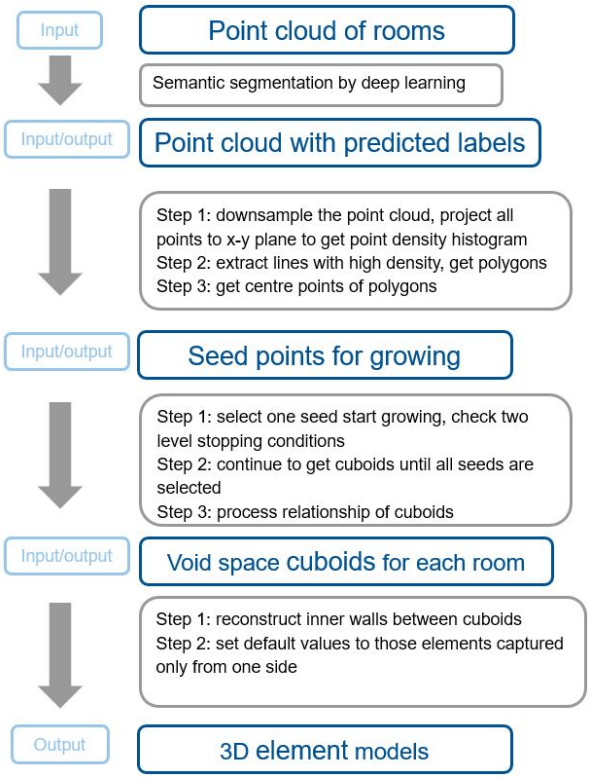


Figure 1. Workflow of proposed approach

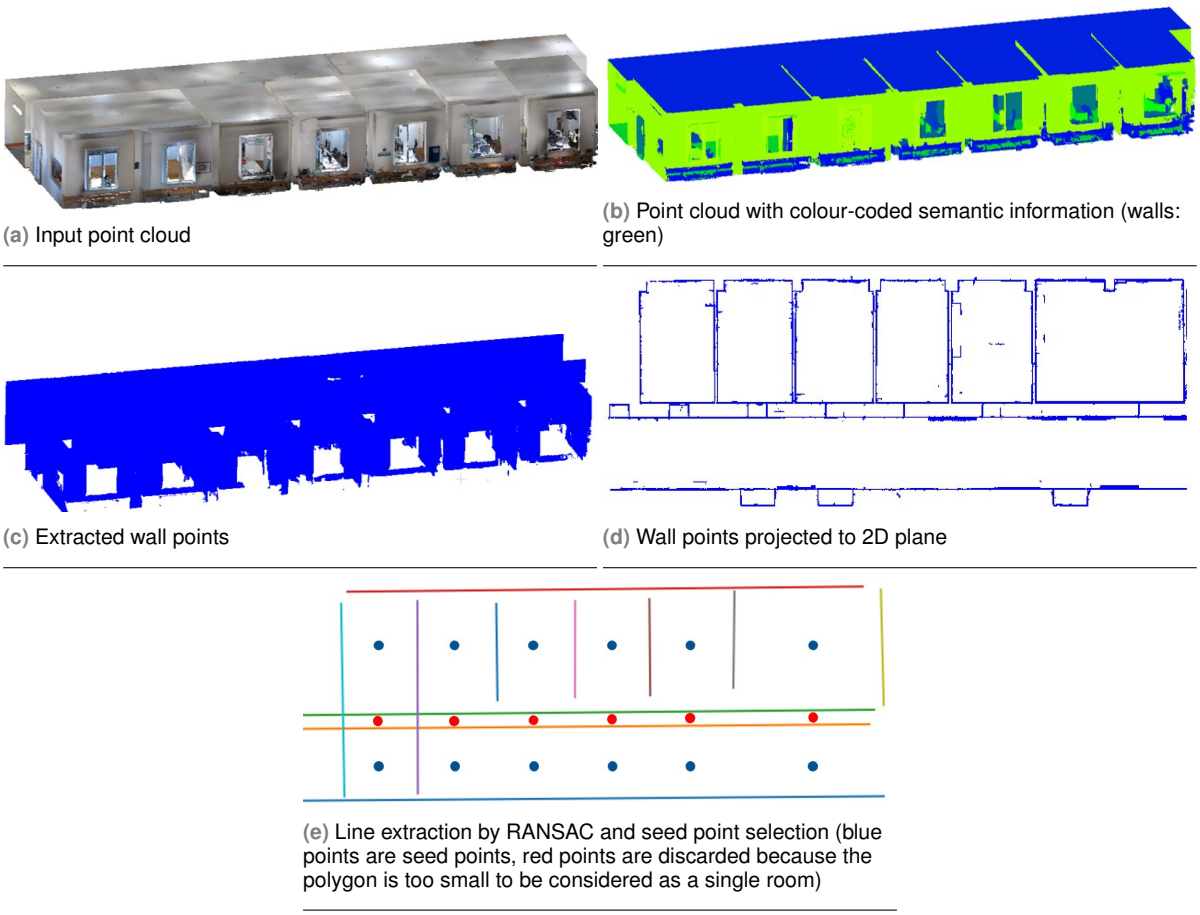
3.2. Point cloud segmentation by deep learning

Because geometric information can be missing if some surfaces are occluded in a point cloud, the proposed approach uses semantic information from deep learning to extract semantic information by deep learning. As the results on the S3DIS dataset (Armeni *et al.*, 2016) are shown in Table 1, KPConv (Thomas *et al.*, 2019) is one of the well-performing neural networks for point cloud segmentation in indoor environment. As the point clouds used in the experiments are also captured in the similar indoor environment, it is reasonable to argue that KPconv can also perform well in the dataset. In this step, the point cloud is tested with a trained KPconv model and predicted semantic information is extracted by the network. More information on the dataset preparation and the results of point cloud segmentation are evaluated in Section 4.

3.3. Generating seeds for growing

In this step, the growing seeds for further steps are generated. The proposed method here is under the Manhattan-world assumption. Firstly, the method performs the voxelisation downsampling method to the point cloud with semantic information. All voxels in the voxelisation process are stored, including void voxels and non-void voxels. While void voxels represent empty space in the point cloud, a non-void voxel means there are points within the voxel. In order to find seed points for growing, all points that are predicted as walls in Section 3.2 by deep learning technology are projected to the XY plane after the voxelisation downsampling. Subsequently, RANSAC is used to extract lines in the XY plane. According to the Manhattan world assumption (Coughlan and Yuille, 2000), lines that are not parallel or perpendicular to X or Y coordinates are not extracted.

The next step is to extend the lines and calculate the intersecting points of these lines to get polygons. Some points (like those on large vertical surfaces of furniture) could also be predicted as wall points and projected on XY plane. Therefore, these polygons are potential



(a) Input point cloud

(b) Point cloud with colour-coded semantic information (walls: green)

(c) Extracted wall points

(d) Wall points projected to 2D plane

(e) Line extraction by RANSAC and seed point selection (blue points are seed points, red points are discarded because the polygon is too small to be considered as a single room)

Figure 2. The process of generating seeds

floor plan representations, not the real floor plan. Then the centroids of these polygons are computed subsequently and selected as the potential seed points in the 2D plane. In 3D space, a default value needs to be set for each potential seed point. In the experiment, $1.5m$ is used as the Z coordinate for seed points. Because the aim of the proposed approach is to grow a void space inside a room, the voxels of the seed points should be void voxels. If not, the neighbouring voxels can be selected as seeds, as in this step, seed points only need to be selected roughly. It is not necessary to distinguish polygons from different rooms. As described later in the algorithm of the growing method, if a seed point from a polygon is included in the grown space of another seed, it will not grow from this point anymore. This process is illustrated in Figure 2.

3.4. Growing from each seed

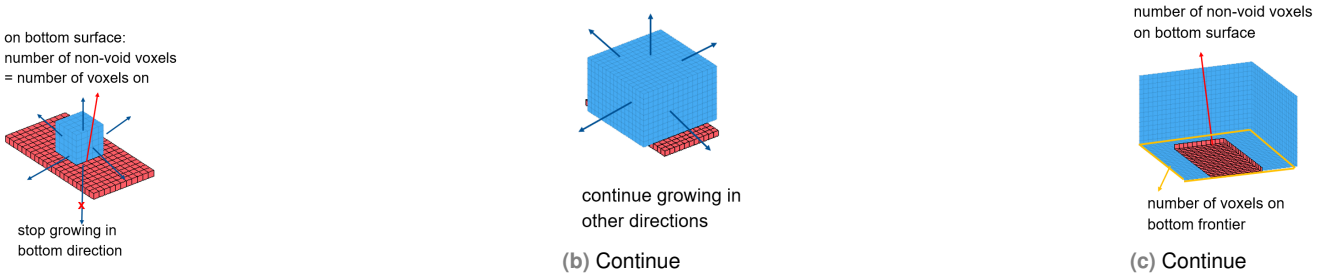
In this step, the method tries to get the largest void space volume inside a room, which means the final space should enclose the points of furniture inside the room. Meanwhile, it should not expand to the outside through window and door openings. The method grows the void space in a ”cuboid” way. That means the void space is grown from one seed (void voxel) in six directions (top, bottom, left, right, front, back). Similar to the region growing approach, neighbours of seed are checked whether they belong to the volume space at each step. The reason why the growing process is done in a ”cuboid” way is that it is clear to check the frontiers in six directions during the cuboid growing process and use the information on frontiers to determine whether it should stop growing in each direction. It is always vital to determine when to stop the growing process. Two different stopping conditions that determine when to stop growing are used: semantic stopping condition and geometric stopping condition.

In the previous step, all voxels from the input point cloud are categorised into two classes: void and non-void voxels. S is used to denote the set that contains all found seeds, and the algorithm picks a seed from S until there are no non-used seeds. This process is introduced as follows, and the pseudocode for the algorithm is shown in Algorithm 1:

- the algorithm checks whether the selected seed has been used before. If so, it would delete this seed and pick another seed;
- the picked seed voxel is added to another set denoted by N_t. It represents the seed set at step t. N_0 denotes the initial set that contains only one seed from S;
- for every seed voxel in N_t at step t, the algorithm finds its 26 neighbours. The set that contains all neighbor voxels is denoted by P. It represents the potential seeds for the next step. And N is used to represent the union of all previous seed sets, from time step 0 to time step t;
- voxels that are already shown in N are removed from P;
- the algorithm checks whether it fulfils any of the stopping conditions on frontiers (stopping conditions will be introduced later in this chapter). The set of these voxels on the frontier is denoted by E. Only if it stops in all six directions the growing process would be stopped;
- voxels in E that belong to the frontier of stopping directions are removed from potential seed set P as it would not grow in the corresponding directions. Moreover, seed points from the previous step in this direction need to be added to the new seed set of the next step. Otherwise, it cannot grow in this direction anymore without the corresponding seeds. The newly generated seed set is denoted by N_{t+1}. Then go back to step 2);

Algorithm 1 The void growing algorithm.

Input:
void voxels and non-void voxels of the point cloud;
initial seed, S;
the stopping conditions, $F()$;
functions to find all neighbors, $\alpha()$;
functions to get voxles on the frontier of any direction, $\beta()$;
Initialize:
voxel list of void volume space in point cloud, $O \leftarrow \varnothing$
Algorithm:
while S is not empty **do**
  select one initial seed N_0 from S
  if $N_0 \in O$ **then**
    while $F(N_t)$ is not fulfilled in six directions **do**
      find seeds' neighbours $P \leftarrow \alpha(N_t)$
      $P \leftarrow P \setminus (P \cap O)$
      if $F(P)$ is fulfilled in any direction **then**
        find voxels in that direction $E \leftarrow \beta(N_t)$
        get seeds for next round $N_{t+1} \leftarrow P \setminus E$
        $N_{t+1} \leftarrow N_{t+1} \cup \beta(Nt-1)$
        $O \leftarrow O \cup N_t$
      end if
    end while
  end if
end while



(a) Stop growing in bottom direction

(b) Continue growing in other directions

(c) Continue growing in bottom direction

Figure 3. Growing process and stopping condition (Pan *et al.*, 2021)

How the growing process works when it grows to a plane is illustrated in Figure 3 for an example. The red voxels represent the non-void voxels that form a plane, while the blue voxels are the void voxels. In Figure 3a, when it grows to a position in the bottom direction that has many non-void voxels (like the top surface of a desk), at this step, the algorithm is not expected to know whether it grows to a desk surface or a floor surface. So, it stops growing in the bottom direction and continues growing in the other five directions until it does not meet the stopping condition in this direction anymore (for example, in Figure 3c).

3.5. The stopping conditions

In the proposed approach, the stopping conditions are defined to determine when to stop the growing process in different directions separately. Despite the fact that the semantic information predicted from deep learning is very valuable, it is not a perfect result. Therefore, semantic information, as well as geometric information, is used as the stopping conditions. In the growing process, the information on the frontier in each direction is checked to determine whether it fulfils the stopping conditions at every step.

The basic idea is that the algorithm compares the ratio between the number of target points in all directions at every step and a predefined threshold value T (in the experiment, the default value is set $T = 0.1$ to make it does not neglect smaller non-void surface). If the ratio is larger than the threshold T, as there is a relatively large number of non-void voxels or there are some voxels with essential labels (like walls, windows, etc.), the algorithm will stop growing in this direction in this step.

3.5.1. Semantic stopping conditions

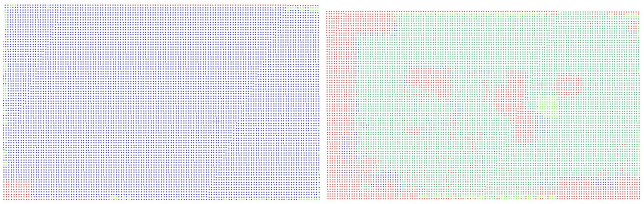
In semantic stopping conditions, labels predicted by deep learning are considered. For the top direction, a ratio value that is used to check whether it stops growing in this direction is defined as follows:

$$r_{top} = P_{ceiling}/Q_{top}, \tag{1}$$

where $P_{ceiling}$ denotes the number of voxels predicted as the ceiling in the top direction, and Q_{top} is the number of nonvoid voxels in the top direction. If $r_{top} > T$, it means there are a number of points predicted as ceiling and the algorithm would stop growing in this direction at this step, and continues checking the stopping conditions in further steps.

Similarly, for the bottom direction, a ratio value is defined as the stopping condition as follows:

$$r_{bottom} = P_{floor}/Q_{bottom}, \tag{2}$$



(a) Top stopping surface (red: void, blue: ceiling voxel, others: voxels predicted as other class)

(b) Bottom stopping surface (red: void, green: floor voxel, others: voxels predicted as other class)

Figure 4. Stopping surfaces in top and bottom directions

where P_{floor} denotes the number of voxels predicted as the floor in the bottom direction, and Q_{bottom} is the number of nonvoid voxels in the bottom direction.

For other four directions (left, right, front, back) where windows and doors might exist in these directions, the ratio value is defined slightly different:

$$r_{other} = (P_{wall} + P_{window} + P_{door})/Q_{other}, \tag{3}$$

where P_{wall}, P_{window}, and P_{door} denotes the number of voxels predicted as wall, window and door in one of the other four directions, and Q_{other} is the number of nonvoid voxels in the same direction.

3.5.2. Geometric stopping conditions

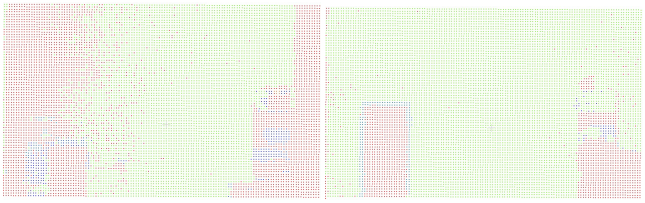
As labels from deep learning are not always correct, it is helpful to consider other information rather than relying solely on semantic information. In geometric stopping conditions, the number of void and nonvoid voxels are considered. It aims to stop the growing process where a plane exists, but the points on the plane are wrongly predicted (for example, wall points are predicted as furniture points).

The ratio that determines whether it stops growing in one direction is defined as follows:

$$r = P/Q, \tag{4}$$

where P denotes the number of nonvoid voxels in one direction, and Q is the number of void voxels in the same direction. If $r > T$, it means there are a number of nonvoid voxels; despite that no semantic information is available, the algorithm should stop growing in this direction at this step and continues checking the stopping conditions in further steps.

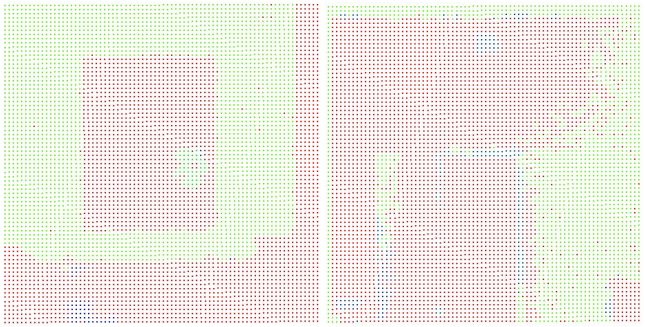
If any stopping condition (semantic or geometric) is fulfilled in one direction, it stops growing in this direction and continues growing in other directions. Whether it stops growing in one direction does not influence the growth of other directions, and it can still enlarge the frontier of the stopped direction when continuing growing in other directions. That means in further steps, if the stopping condition is not fulfilled anymore, it can continue growing in this direction. When it stops growing in all six directions, both stopping conditions are checked. If still only one stopping condition is met in this direction, it continues growing in this direction until both conditions are met. The result is supposed to be the void space, whose top surface is the ceiling, the bottom surface is the floor, and the other four surfaces are the wall surfaces.



(a) Left stopping surface (red: void, green: wall voxel, others: voxels predicted as other class)

(b) Right stopping surface (red: void, green: wall voxel, others: voxels predicted as other class)

Figure 5. Stopping surfaces in left and right directions



(a) Front stopping surface (red: void, green: wall voxel, others: voxels predicted as other class)

(b) Back stopping surface (red: void, green: wall voxel, others: voxels predicted as other class)

Figure 6. Stopping surfaces in front and back directions

The surfaces where the algorithm stops growing in the bottom and top directions are shown in Figure 4. Similarly, the surfaces where it stops growing in other four directions are shown in Figure 5 and Figure 6. Red points denote the centre points of void voxels; green points denote the centre points of voxels predicted as walls; blue points represent the centre points of voxels predicted as other classes (like doors, windows). As the number of nonvoid voxels (green and blue) is similar to that of nonvoid voxels (others), the geometric stopping condition is met. Meanwhile, among all the nonvoid voxels, most points are predicted as walls, which means the geometric stopping condition is met. Therefore, the geometric and semantic stopping conditions meet at the same time in the final growing result.

3.6. Detect window and door openings

The patterns on frontiers in each direction when it stops growing are checked to identify whether there are openings on the wall in the corresponding direction. When capturing buildings by laser scanners, doors are usually open. This and the fact that the window glass surfaces do not reflect the laser beam lead to the fact that windows and doors are void areas without any points (like in Figure 7). As the proposed approach considers void space inside the point cloud, it is convenient to check the void area on the wall surfaces where the algorithm stops growing.

A wall surface with a door opening where it stops growing is shown in Figure 7. In this figure, red points are the centre points of void voxels; green points represent the centre points of non-void voxels predicted as the wall; blue points are the predicted door points. It is obvious to see that in the marked area, there is a red rectangle (opening) surrounded by blue points (door). Therefore, the door opening rectangle can be extracted by fitting a minimum rectangle that encloses all the points of void voxels.

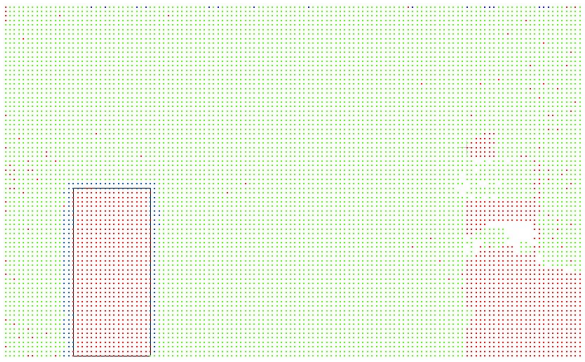


Figure 7. A wall surface with a door opening

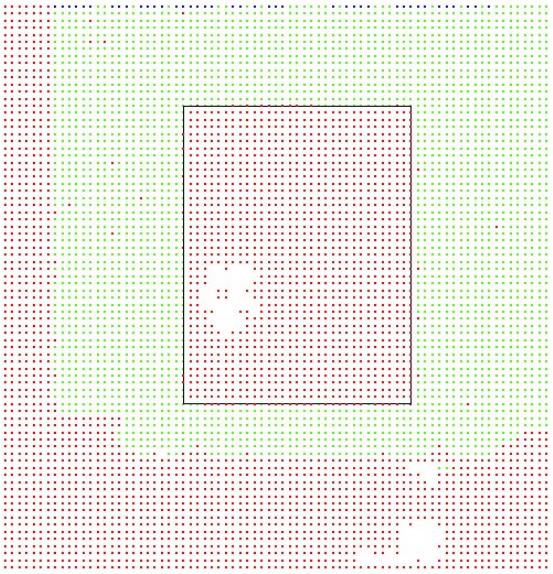


Figure 8. A wall surface with a window opening

A wall surface with a window opening where it stops growing is shown in Figure 8. In this figure, again, red points are the centre points of void voxels; green points represent the centre points of voxels predicted as the wall; blue points are the predicted window points. In the marked area, it is obvious that there is a red rectangle (opening) and some window points at the boundary between wall points (green) and centre points of void voxels (red).

However, it would cause a problem if those predicted door/window points were used to extract a rectangle. As shown in Figure 7 and Figure 8, the predicted window and door points are not perfect rectangles because many points are wrongly predicted as other classes, and this would make the fitting result imprecise. Another reason why it is not a proper way to extract doors and windows from the predicted points can be seen from Table 2. The result for doors and windows (mIoU around 44%) is worse than that of the other three elements (ceiling, floor, and wall's mIoU larger than 80%).

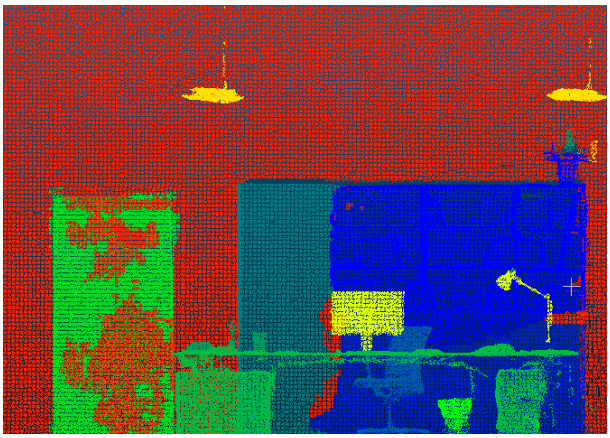
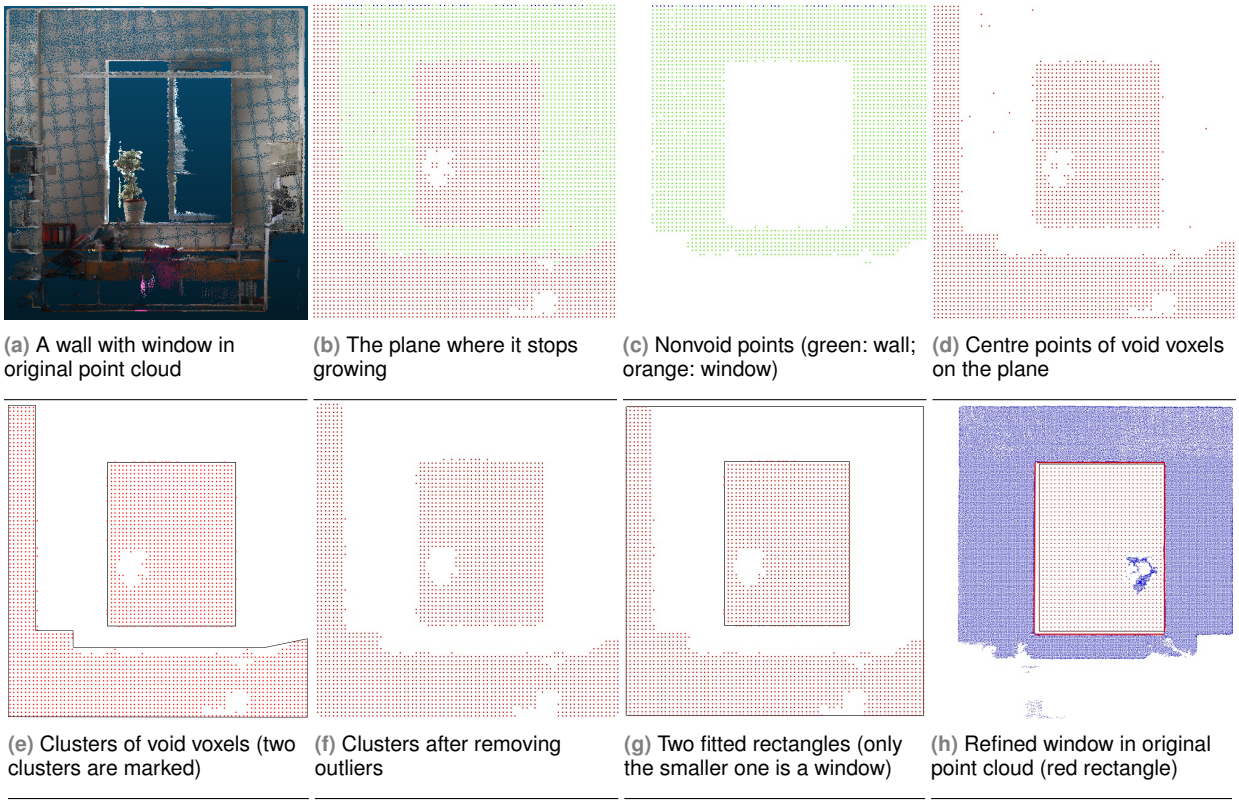


Figure 9. The prediction of a bookcase in front of a wall

Therefore, in the proposed approach, semantic information about doors and windows is only used to check the existence of doors and windows, not used in the opening reconstruction. Rectangles are extracted based on geometric information of the void space inside walls under the assumption that all door and window openings are rectangles.

Semantic information from deep learning is quite helpful to distinguish the wall surfaces and furniture surfaces, despite the fact that some furniture surfaces are sometimes very large. These large furniture surfaces are quite hard to identify if using geometric information only. A large bookcase in front of a wall is shown in Figure 9. In this figure, red points denote the wall behind the bookcase; blue points are predicted as a bookshelf, while grey points next to the blue points are predicted as closet points. Although the neural network model predicts the points of a bookcase into two classes, it is understandable because closets and bookshelves are sometimes quite similar in real life. But it does not mislabel points of furniture to points of the wall, which is important to determine when to stop growing.

The process of fitting rectangles to find a window opening on the wall is illustrated in Figure 10. In the plane where a window exists, the window opening should be close to a rectangle, assuming all windows in the building are rectangles. Apart from the window opening, if the wall is occluded by some furniture which is quite common in the indoor environment, there would also be some furniture void voxels, as shown in Figure 10b. But usually, these void spaces have different shapes from windows and doors. Moreover, on the boundary of the window void area and the wall area, there are some points predicted as window points by deep learning, like the orange points in 10c. The semantic information of window points can be used to distinguish the void space of a window opening from that caused by furniture. Then all void voxels (as shown in Figure 10d) in the plane are clustered into different point clusters based on the Euclidean distances between a point to its neighbours, as shown in Figure 10e. The next step is to remove the outliers in each cluster by applying the statistical outlier removal filter in Point Cloud Library (Rusu and Cousins, 2011) (shown in Figure 10f). At last, the minimal rectangle that can enclose all points in the cluster is fitted to the point cluster, as shown in Figure 10g. Because the larger rectangle in Figure 10g does not have any predicted window points on the boundary (not a window based on semantic information) and the void points inside the rectangle do not form a rectangle shape (not a window based on geometric information), the larger fitted rectangle can be removed from a window opening directly. As the dimension of door and window openings is relatively small compared with the dimension of ceilings and walls, the chosen voxel size used to downsample the point cloud has a larger impact on the dimension of doors or windows. The detected rectangles are refined by fitting edges again in the original input point cloud, as shown in Figure 10h. In the original point cloud, the edges of rectangles are adjusted horizontally or vertically within a voxel range until they fit the boundary of points.



(a) A wall with window in original point cloud

(b) The plane where it stops growing

(c) Nonvoid points (green: wall; orange: window)

(d) Centre points of void voxels on the plane

(e) Clusters of void voxels (two clusters are marked)

(f) Clusters after removing outliers

(g) Two fitted rectangles (only the smaller one is a window)

(h) Refined window in original point cloud (red rectangle)

Figure 10. The process of extracting a window opening

The process of fitting rectangles to find a door opening on the wall is quite similar and illustrated in Figure 11. The main difference is that some points that are predicted as the door rather than the window are considered to detect the door opening. By considering whether there are window or door points on the boundary of the opening area and wall area, an opening can be classified as a door or window opening. If no semantic information is available, prior knowledge, such as that door openings are usually connected to the floor surface, is used to identify door openings from window openings.

3.7. Merge the connected cuboids

From the previous steps, cuboids for void volume inside rooms are extracted from the point cloud. Because there is at least one seed inside one room, one room could have multiple cuboids from different seed points. This usually happens in some complex rooms, like U-shape rooms, L-shape rooms, rooms with different ceiling heights, etc. Therefore, these cuboids should be merged into one. There are two circumstances in that two cuboids should be merged into one: a) one surface of a cuboid touches a surface of another cuboid: b) two cuboids have overlapped space.

3.8. Extract walls, floors, and ceilings

In this step, elements are reconstructed from the cuboids generated from the last step. Different strategies are used for different kinds of structural elements.

For ceilings, floors, and outer walls, it is common that only the inside surfaces are captured. For these elements that are only captured from one side, the surface where void volumes stop growing is considered the inner surface of these elements. The thickness of these elements is set to a default value because the information is not available in the point cloud.

(a) A wall with door in original point cloud

(b) The plane where it stops growing

(c) Nonvoid points (green: wall; blue: door)

(d) Centre points of void voxels on the plane

(e) Clusters of void voxels (three clusters are found)

(f) Clusters after removing outliers

(g) Three fitted rectangles (only the right one is a door)

(h) Refined door in original point cloud (red rectangle)

Figure 11. The process of extracting a window opening

In contrast, both sides of the inner walls usually are scanned when collecting data inside a building. That means two void cuboids are grown in the two adjacent rooms. The space between two void cuboids is considered the inner wall. The thickness of the inner wall is the distance between these two surfaces.

3.9. Extract windows and doors

In Section 3.6, the location and dimension of doors and windows can be extracted by fitting rectangles under the assumption that all doors and windows in the building are rectangles. Two rectangles extracted from the two sides of the wall should be identical in an ideal case. However, it is almost impossible to fit two exactly identical rectangles from the two sides. In order to get one rectangle that can represent the door or window openings, a new rectangle is computed from two old rectangles, whose width and height are the mean values of two old rectangles, and the new centre point is the middle point of the two old centre points.

4. Experiment and result

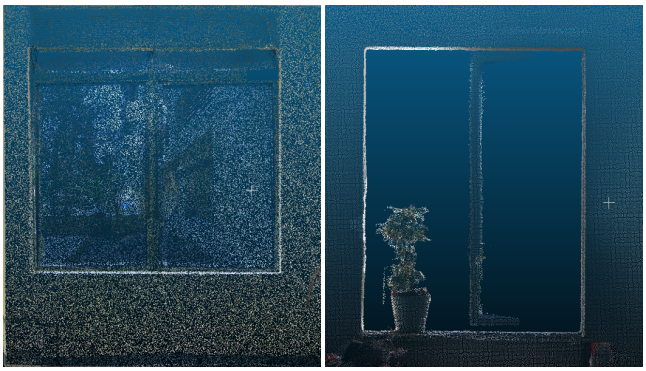
The code of void-growing approach is written in C++ by using Point Cloud Library (PCL) 1.9.1 (Rusu and Cousins, 2011) and the Computational Geometry Algorithms Library (CGAL) 5.1 (The CGAL Project, 2020).

The point cloud that is used in the reconstruction process in this paper is a laser scanning point cloud with 5mm resolution and captured in the office space at the Chair of Computational Modelling and Simulation (CMS) at the Technical University of Munich (TUM). This dataset is also the indoor environment of an office building, which is similar to that of the S3DIS dataset. In this paper, the dataset captured at TUM is called the TUMCMS dataset in the following sections. The TUMCMS dataset is labelled manually and split into the training and validation set (the training set is around 70% of the total area and the validation set is around 30%). Three training set settings are made: trained on the S3DIS dataset only, TUMCMS training set only, as well as S3DIS and TUMCMS training set. In addition, the resulting improvement when applying door and window points refinement described in Section 3.9 for two different models is also shown. The results of different models are listed in Table 2.

It is obvious to see that the model trained on both the S3DIS dataset and TUMCMS training set performs best in wall and floor class. However, it performs worse in window class compared with training solely on the TUMCMS dataset. The main reason is that two datasets

model	training set	test set	ceiling	wall	floor	window	door
KPConv	TUMCMS(train)	TUMCMS (test)	94.7	81.8	96.6	54.4	48.8
KPConv	S3DIS	TUMCMS (test)	93.6	73.9	88.3	15.3	17.3
KPConv without refinement	S3DIS and TUMCMS (train)	TUMCMS (test)	93.5	82.9	97.3	48.7	47.1
KPConv with refinement	S3DIS and TUMCMS (train)	TUMCMS (test)	93.5	82.7	97.2	68.4	60.2
PointTransformer (no refinement)	S3DIS and TUMCMS (train)	TUMCMS (test)	94.4	83.1	97.8	52.2	50.1
PointTransformer with refinement	S3DIS and TUMCMS (train)	TUMCMS (test)	94.4	83.0	97.7	70.2	64.4

Table 2. Segmentation IoUs (%) on TUMCMS test set

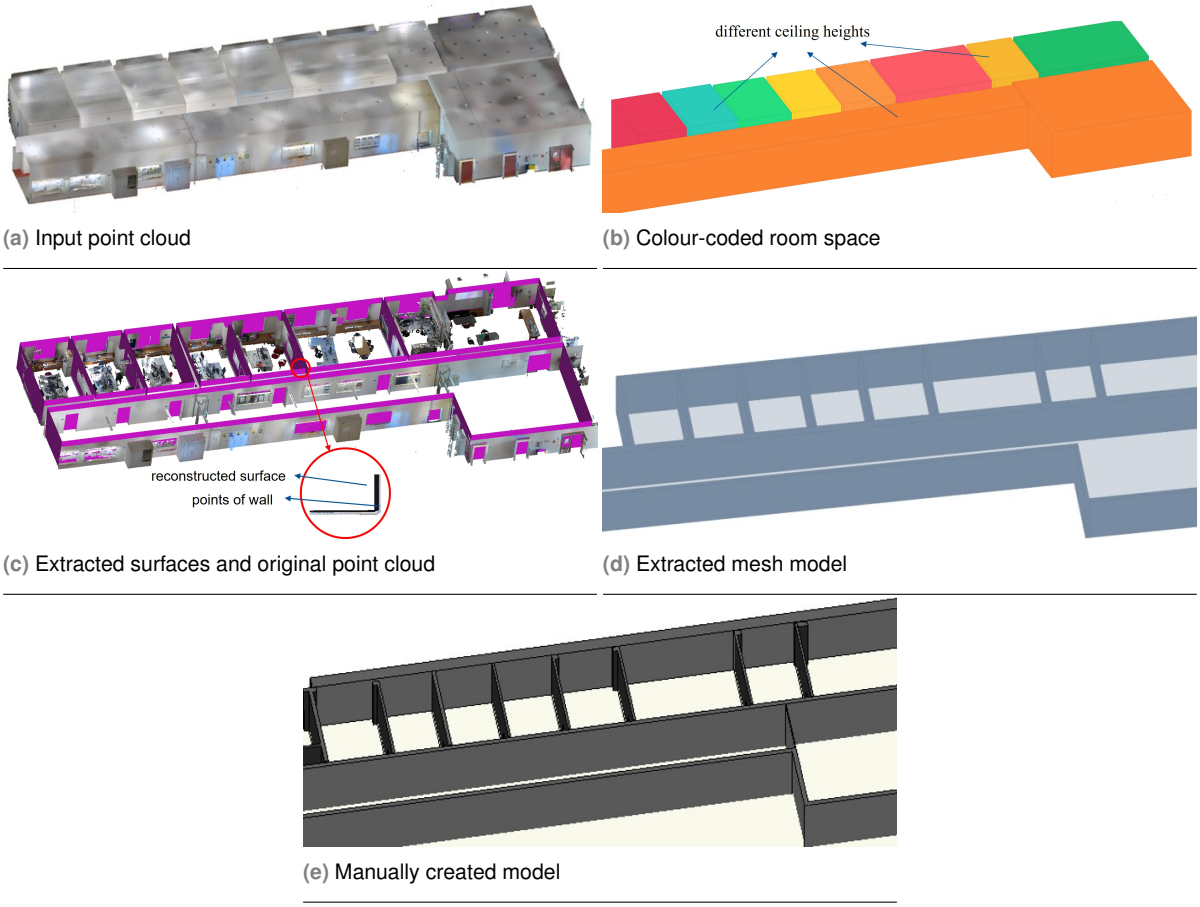


(a) Window in S3DIS dataset **(b)** Window in TUMCMS dataset

Figure 12. Windows in S3DIS and TUMCMS dataset

are captured by different technologies (S3DIS used RGBD camera and TUMCMS used laser scanner), which makes the property of point clouds non-consistent for different objects. One example is shown in Figure 12. While windows in the S3DIS dataset contain points and show outside scenes, windows in the TUMCMS dataset have no points and are actually ”void” on the wall. Therefore, adding TUMCMS data to the training set can improve the result of the window class. But the result is not as good as that if using laser-scanned point cloud only (54.4% and 48.7%, respectively). In order to improve the prediction results of the window prediction, it would be better to conduct training and test on the data coming from the same source. In contrast, objects like walls, ceilings, and floors in the two datasets are quite similar, which makes the training on both datasets able to improve the performance of the network.

The qualitative evaluation result on the part of the test set in the TUMCMS dataset is illustrated 13. The extract room spaces are illustrated in Figure 13b. Each colour represents a void volume inside a room. It can be seen that not only standard cuboid rooms but also complex rooms can be detected. For example, the hallway can be seen as an L-Shape room. Furthermore, if focusing on the ceiling of the input cloud, the ceiling heights of some rooms and the hallway are not identical because of suspended ceilings which are quite common in the building industry nowadays. The different ceiling heights in the input point cloud can be clearly identified in the grown void volumes. In Figure 13c, it can be seen that the alignment of reconstructed surfaces and the original point cloud. The points representing the ceiling are removed to depict rooms of the input point cloud more clearly. It is obvious that the shift is small, and the quantitative evaluation is discussed later in this section. Based on the volume spaces found in previous steps, walls, floors, and ceilings can be extracted. In the experiment, if only one surface of the elements is scanned, the default thickness of the element is set to 30cm. The reconstructed 3D model created by the proposed approach and the BIM model created manually are shown in Figure 13d and 13e.



(a) Input point cloud

(b) Colour-coded room space

(c) Extracted surfaces and original point cloud

(d) Extracted mesh model

(e) Manually created model

Figure 13. Qualitative evaluation on TUMCMS test set

By quantitative evaluation, the room areas in the TUMCMS test set are evaluated by firstly comparing area values, absolute and relative deviation, overlapped area, and Intersection over Union (IoU) of individual rooms in Table 3. Basically, if using R_i to denote the room space of the extracted method for room i and S_i to denote the room space of room i in the manually created BIM model, the IoU for the room i is computed as

$$Room^{IoU}\, i = \frac{R_i \cap S_i}{R_i \cup S_i}. \tag{5}$$

The performance of choosing different voxel sizes is also compared in the table, and it shows a smaller voxel size performs better than the larger one. In addition, the result on another dataset is also evaluated, which is captured in a different office building at the Technical University of Munich (TUM), as shown in Table 4, makred as Dataset 2. The input point cloud and corresponding created model of the approach for Dataset 2 are illustrated in Figure 14 for qualitative evaluation as well.

In addition, the thicknesses of some selected walls from the approach and manually created model from two datasets are shown in Table 5. The proposed algorithm could apply to other point clouds of buildings without any modifications or with slight modifications. In most cases, if the buildings have similar door and window openings, it can be applied directly to new datasets. However, if the door and window openings have different shapes (like circles), the model has never been trained on this kind of door and window. It is understandable that

Figure 14. The input point cloud and corresponding created model of the approach for Dataset 2

Table 3. Area comparison between the void-growing model and BIM model on TUMCMS test set

office No.	voxel size (cm)	void-growing (m^2)	BIM (m^2)	abs. dev. (m^2)	rel. dev. ($\%$)	overlap area (m^2)	IoU ($\%$)
office 1	5cm	46.62	49.22	2.60	5.28	45.55	94.24
office 2	5cm	26.15	26.82	0.67	2.50	25.85	93.15
office 3	5cm	23.30	24.11	0.80	3.32	23.13	92.15
office 4	5cm	25.20	25.88	0.68	2.63	24.92	95.03
office 5	5cm	23.75	25.20	1.45	5.75	23.23	94.58
office 6	5cm	23.75	24.90	1.15	4.62	23.12	93.18
office 1	3cm	47.69	49.22	1.53	3.11	46.90	94.74
office 2	3cm	26.35	26.82	0.47	1.75	26.22	74.02
office 3	3cm	23.64	24.11	0.47	1.95	23.48	92.78
office 4	3cm	25.30	25.88	0.58	2.24	25.20	94.64
office 5	3cm	23.82	25.20	1.38	5.48	23.41	95.23
office 6	3cm	24.10	24.90	0.80	3.21	23.85	93.54

Table 4. Area comparison between the void-growing model and grount truth on Dataset 2 (5cm voxel size)

office No.	void-growing (m^2)	BIM (m^2)	abs. dev. (m^2)	rel. dev. ($\%$)	overlap area (m^2)	IoU ($\%$)
office 1	17.75	18.79	1.04	5.53	17.02	93.29
office 2	26.46	28.56	2.10	7.35	25.88	92.55
office 3	21.65	22.87	1.22	5.33	20.03	91.06
office 4	22.58	23.36	0.78	3.34	21.56	93.74

the network cannot detect elements that it has never seen before. To find and fit these doors and windows, the training set can be enlarged or other predefined opening shapes can be used as available prior knowledge.

4.1. Discussion

Most previous research starts with detecting elements by considering geometric information only and subsequently extracting room information. However, geometric information of target objects could be missing because of the occlusion, which makes these methods perform worse in an occluded environment. The void growing method detects spaces inside rooms first and then extracts building elements by considering geometric information and semantic information predicted from deep learning, which makes it possible to distinguish wall surfaces from furniture surfaces in an occluded environment, even when the furniture surfaces are large.

The voxel size used in the downsampling process limits the performance of the approach. When detecting objects with large dimensions, the impact is insignificant. However, it becomes vital to determine the thickness of elements because the thickness is usually not very large. A potential solution is to compute the mean points of all points inside a voxel by averaging each coordinate and then considering the mean point as the downsampled point. The results can be benefited by reducing the voxel size, but the computational effort would

Table 5. Wall thickness comparison between the void-growing model and BIM model of two dataset with different voxel sizes

Dataset No.	wall No.	voxel size (m)	void-growing (m)	BIM (m)	abs.dev. (m)	rel. dev. (rel.$\%$)
Dataset 1	wall 1	5cm	0.20	0.17	0.03	17.6
Dataset 1	wall 2	5cm	0.20	0.16	0.04	25.0
Dataset 1	wall 3	5cm	0.15	0.14	0.01	7.1
Dataset 1	wall 4	5cm	0.20	0.17	0.03	17.6
Dataset 1	wall 5	5cm	0.20	0.17	0.03	17.6
Dataset 1	wall 1	3cm	0.18	0.17	0.01	5.9
Dataset 1	wall 2	3cm	0.18	0.16	0.02	12.5
Dataset 1	wall 3	3cm	0.15	0.14	0.01	7.1
Dataset 1	wall 4	3cm	0.18	0.17	0.01	5.9
Dataset 1	wall 5	3cm	0.18	0.17	0.01	5.9
Dataset 2	wall 1	5cm	0.20	0.21	0.01	4.8
Dataset 2	wall 2	5cm	0.20	0.22	0.02	9.1
Dataset 2	wall 3	5cm	0.20	0.18	0.02	11.1
Dataset 2	wall 4	5cm	0.15	0.20	0.05	25.0
Dataset 2	wall 1	3cm	0.18	0.21	0.03	14.3
Dataset 2	wall 2	3cm	0.21	0.22	0.01	4.5
Dataset 2	wall 3	3cm	0.18	0.18	0	0
Dataset 2	wall 4	3cm	0.18	0.20	0.01	5.0

also be enlarged. Another limitation is that the proposed approach is suitable for laser scanning point cloud but not for point cloud from RGBD camera because window openings in the point cloud is used to detect windows and find the opening dimension and location of the windows. The window openings are actually void in the laser scanning point cloud but not void in the point cloud captured from the RGBD camera (as shown in 12).

5. Conclusion

In conclusion, the paper presents an automatic method based on the void-growing method that extracts void room space inside rooms first, which is proposed in the previous conference paper (Pan *et al.*, 2021). In addition, state-of-the-art deep learning technologies are added to the initial void-growing method to improve the performance of extracting walls, ceilings, and floors. Semantic information from deep learning is also used to extract doors and windows. However, the window and door recognition by point cloud deep learning is not improved when adding more training data from different sources. The final geometric digital twin includes point clusters with annotated semantic information, which shows the detailed information of captured surfaces and a simplified mesh model of the structural elements. As the detailed geometric information provides information that could be missed during the process of creating a simplified mesh, the final output can be applied in use cases that require detailed information, like asset condition monitoring. As the proposed method can automatically extract ceilings, walls with doors and windows, and floors in both simple cuboid rooms and rooms with complex structures, it can reduce a considerable amount of human effort to create a geometric digital twin of buildings. Human modelers only need to check the correctness and accuracy of the automatically created model and then add other objects into the model depending on the requirements.

In the future, apart from ceilings, floors, walls, windows, and doors, next step is to extend the current approach to detect more elements like columns, beams, staircases, etc. More complex buildings (like buildings with slanted ceilings, curved walls, etc.) should also be considered. Furthermore, many other elements are planned to be reconstructed in the digital twinning process, like light texture, furniture, and smoke alarms, which would make the digital twin more valuable in building management and maintenance.

6. Acknowledgements

The work in this paper is funded by the Institute for Advanced Study (IAS) at the Technical University of Munich. This research has been conducted by support from the NVIDIA Research Accelerator Program. The dataset used in this paper is collected on the main campus of the Technical University of Munich with the help of NAVVIS (https://www.navvis.com/).

REFERENCES

Adan A and Huber D (2011) 3d reconstruction of interior wall surfaces under occlusion and clutter. In *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, IEEE, pp. 275–281.

Agapaki E, Miatt G and Brilakis I (2018) Prioritizing object types for modelling existing industrial facilities. *Automation in Construction* **96**: 211–223.

Armeni I, Sener O, Zamir AR, Jiang H, Brilakis I, Fischer M and Savarese S (2016) 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

Brilakis I, Pan Y, Borrmann A, Mayer HG, Rhein F, Vos C, Pettinato E and Wagner S (2019) Built environment digital twining. International Workshop on Built Environment Digital Twinning presented by ....

Budroni A and Boehm J (2010) Automated 3d reconstruction of interiors from point clouds. *International Journal of Architectural Computing* **8(1)**: 55–73.

Chen M, Feng A, McAlinden R and Soibelman L (2020) Photogrammetric point cloud segmentation and object information extraction for creating virtual environments and simulations. *Journal of Management in Engineering* **36(2)**: 04019046.

Coughlan J and Yuille AL (2000) The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. *Advances in Neural Information Processing Systems* **13**: 845–851.

Engel N, Belagiannis V and Dietmayer K (2021) Point transformer. *IEEE Access* **9**: 134826–134840.

Fan S, Dong Q, Zhu F, Lv Y, Ye P and Wang FY (2021) Scf-net: Learning spatial contextual features for large-scale point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14504–14513.

Goodfellow I, Bengio Y and Courville A (2016) *Deep learning*. MIT press.

Guo MH, Cai JX, Liu ZN, Mu TJ, Martin RR and Hu SM (2021) Pct: Point cloud transformer. *Computational Visual Media* **7(2)**: 187–199.

Haghighatgou N, Daniel S and Badard T (2022) A method for automatic identification of openings in buildings facades based on mobile lidar point clouds for assessing impacts of floodings. *International Journal of Applied Earth Observation and Geoinformation* **108**: 102757.

Hu Q, Yang B, Xie L, Rosa S, Guo Y, Wang Z, Trigoni N and Markham A (2020) Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11108–11117.

Huang Q, Wang W and Neumann U (2018) Recurrent slice networks for 3d segmentation of point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2626–2635.

Krizhevsky A, Sutskever I and Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* **25**: 1097–1105.

Landrieu L and Simonovsky M (2018) Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4558–4567.

Li Y, Bu R, Sun M, Wu W, Di X and Chen B (2018) Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems* **31**: 820–830.

Lu R and Brilakis I (2017) Recursive segmentation for as-is bridge information modelling .

Macher H, Landes T and Grussenmeyer P (2017) From point clouds to building information models: 3d semi-automatic reconstruction of indoors of existing buildings. *Applied Sciences* **7(10)**: 1030.

Maturana D and Scherer S (2015) Voxnet: A 3d convolutional neural network for real-time object recognition. In *Ieee/rsj International Conference on Intelligent Robots and Systems*, pp. 922–928.

Mayer H and Reznik S (2005) Building facade interpretation from image sequences. In *Proceedings of the ISPRS Workshop CMRT*, Citeseer, pp. 55–60.

Monszpart A, Mellado N, Brostow GJ and Mitra NJ (2015) Rapter: rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph.* **34(4)**: 103–1.

Mura C, Mattausch O, Villanueva AJ, Gobbetti E and Pajarola R (2014) Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Computers & Graphics* **44**: 20–32.

Murali S, Speciale P, Oswald MR and Pollefeys M (2017) Indoor scan2bim: Building information models of house interiors. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 6126–6133.

Ochmann S, Vock R and Klein R (2019) Automatic reconstruction of fully volumetric 3d building models from oriented point clouds. *ISPRS journal of photogrammetry and remote sensing* **151**: 251–262.

Ochmann S, Vock R, Wessel R and Klein R (2016) Automatic reconstruction of parametric building models from indoor point clouds. *Computers & Graphics* **54**: 94–103.

Oesau S, Lafarge F and Alliez P (2013) Indoor scene reconstruction using primitive-driven space partitioning and graph-cut.

Pan Y, Braun A, Borrmann A and Brilakis I (2021) Void-growing: a novel scan-to-BIM method for manhattan world buildings from point cloud. In *Proceedings of the 2021 European Conference on Computing in Construction*, *Computing in Construction*, vol. 2, University College Dublin, Online, pp. 312–321, 10.35490/ec3.2021.162.

Perez-Perez Y, Golparvar-Fard M and El-Rayes K (2021) Scan2bim-net: Deep learning method for segmentation of point clouds for scan-to-bim. *Journal of Construction Engineering and Management* **147(9)**: 04021107.

Prokhorov D (2010) A convolutional learning system for object classification in 3-d lidar data. *Trans. Neur. Netw.* **21(5)**: 858–863, 10.1109/TNN.2010.2044802.

Pu S and Vosselman G (2007) Extracting windows from terrestrial laser scanning. *Intl Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* **36**: 12–14.

Pu S and Vosselman G (2009) Knowledge based reconstruction of building models from terrestrial laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing* **64(6)**: 575–584.

Qi CR, Su H, Mo K and Guibas LJ (2016) Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593* .

Qi CR, Yi L, Su H and Guibas LJ (2017) Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413* .

Qiu S, Anwar S and Barnes N (2021) Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1757–1767.

Ripperda N (2008) Determination of facade attributes for facade reconstruction. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* **37(B3a)**: 285–290.

Rusu RB and Cousins S (2011) 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

Sanchez V and Zakhor A (2012) Planar 3d modeling of building interiors from point cloud data. In *2012 19th IEEE International Conference on Image Processing*, IEEE, pp. 1777–1780.

Stambler A and Huber D (2014) Building modeling through enclosure reasoning. In *2014 2nd International Conference on 3D Vision*, vol. 2, IEEE, pp. 118–125.

The CGAL Project (2020) *CGAL User and Reference Manual*. 5.1.1 edn., CGAL Editorial Board, https://doc.cgal.org/5.1.1/Manual/packages.html.

Thomas H, Qi CR, Deschaud JE, Marcotegui B, Goulette F and Guibas LJ (2019) Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6411–6420.

Tran H, Khoshelham K, Kealy A and Díaz-Vilariño L (2019) Shape grammar approach to 3d modeling of indoor environments using point clouds. *Journal of Computing in Civil Engineering* **33(1)**: 04018055.

Truong-Hong L, Laefer DF, Hinks T and Carr H (2013) Combining an angle criterion with voxelization and the flying voxel method in reconstructing building models from lidar data. *Computer-Aided Civil and Infrastructure Engineering* **28(2)**: 112–129.

Truong-Hong L and Lindenbergh R (2022) Extracting structural components of concrete buildings from laser scanning point clouds from construction sites. *Advanced Engineering Informatics* **51**: 101490.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł and Polosukhin I (2017) Attention is all you need. *Advances in neural information processing systems* **30**.

Wang R, Xie L and Chen D (2017) Modeling indoor spaces using decomposition and reconstruction of structural elements. *Photogrammetric Engineering & Remote Sensing* **83(12)**: 827–841.

Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM and Solomon JM (2018) Dynamic graph CNN for learning on point clouds. *CoRR* **abs/1801.07829**, http://arxiv.org/abs/1801.07829, 1801.07829.

Xiao J and Furukawa Y (2014) Reconstructing the world's museums. *International journal of computer vision* **110(3)**: 243–258.

Xiong X, Adan A, Akinci B and Huber D (2013) Automatic creation of semantically rich 3d building models from laser scanner data. *Automation in construction* **31**: 325–337.

Zhao H, Jiang L, Jia J, Torr P and Koltun V (2020) Point transformer. 2012.09164.

Zhao H, Jiang L, Jia J, Torr PH and Koltun V (2021) Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268.