Towards applicable Scan-to-BIM and Scan-to-Floorplan: An end-to-end experiment

Fiona C. Collins*, M. Saeed Mafipour*, Florian Noichl*, Yuandong Pan*, Miguel A. Vega*

TU Munich, Chair for Computational Modeling and Simulation
*in alphabetical order, authors have contributed equally

Abstract

Many research efforts in the fields of Scan-to-BIM and Scan-to-Floorplan have been focusing on automating the solutions in a stepwise manner. These limited approaches lead to improving the quantitative results of a single step, not the overall process. The challenge of covering a large variety of use-cases with a vast diversity of building element classes requires robust, modular, and configurable techniques. This paper proposes an end-to-end method covering all the steps from raw input scan data to two different output formats: A smart 2D instance floor plan and a 3D model. As a use case, the floorplan and the 3D model of typical office space are reconstructed and all the various, yet modular techniques along the way are presented. The results of the paper are demonstrated as metrics for a publicly available benchmark dataset, along with a discussion and an outlook for possible further development of each component.

Keywords: Scan-to-BIM, Scan-to-Floorplan, point cloud, digital twin, laser scan

1 Motivation

As-is BIM as a step towards full Digital twins can provide the digital replica of existing assets in which all the gathered data from the construction site can be imported. These models also facilitate the operation and management process of existing structures. Despite these advantages, digitalization still requires spending plenty of time. This is mainly due to the long lifecycle of the existing assets and the challenges of the manual digital twinning process. Laser scanning and photogrammetry are methods that can capture the building assets with high accuracy in little time [1, 2, 3]. However, using scans and transferring the resultant point cloud data to useful digital content is tedious and labor-intensive. The research field of Scan-to-BIM aims to investigate potentials of automating different parts of this process. In some cases, the technical limitations of either the processing toolchain or the input data might let the stakeholders favor a smart 2D representation rather than a 3D BIM format. PDF or SVG floor plans where each building element instance is denoted and callable over a unique identifier can, in this case, offer a good alternative to a full 3D BIM model. In other cases, the 3D as-is BIM models can provide a better basis for digitalization and visualization. With the dataset of the CV4AEC challenge[1] recently made available, we make use of one of the most extensive publicly available datasets. The variety of input data in its sheer extent made it possible for us to combine diverse approaches. Figure 1 shows the guiding thread of this paper and summarizes the end-to-end workflow.

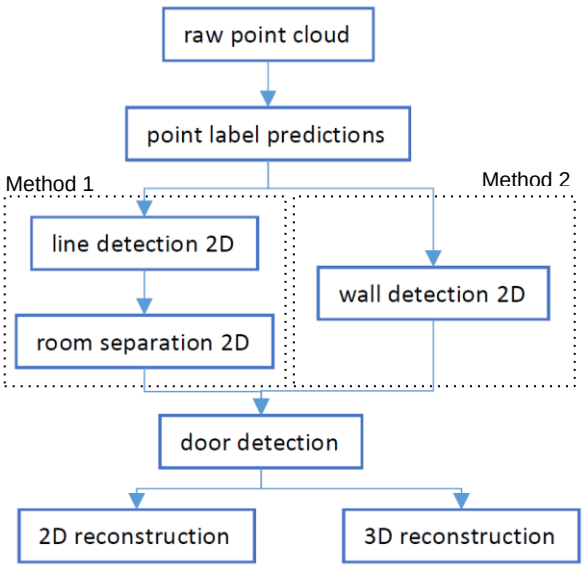


Figure 1: High-level workflow overview

[1] https://cv4aec.github.io/

2 Overview

Thanks to the variety of input data available in the dataset, we were able to test a diverse selection of approaches for every single step in the toolchain. We introduce the choices we made in this project and include possible alternatives and potentials for improvement for each one in the discussion section. This project was conducted in the framework of an official workshop on the topic of Scan-to-BIM and Scan-to-Floorplan, therefore we limited our data usage to the data provided by the hosts and formulated our targets in accordance to the given rules, which are introduced in short in the paragraph below.

Data input & requirements

Table 1: Overview of the utilized training dataset

building area	MedOffice F3	MedOffice F2	OfficeLab	Facility	MedOffice B	MedOffice F1	total
area m^2	2'284	3'804	2'082	7'198	5'963	4'080	25'411
no. of points	$101x10^6$	$112x10^6$	$98x10^6$	$116x10^6$	$123x10^6$	$137x10^6$	$571x10^6$

The training set comprises 6 areas of laser scanned, unlabeled point clouds (see Table 1), 3D CAD models in DXF format[1] and JSON floorplans. Models and point clouds for these building areas are aligned in the same coordinate systems, all point clouds include RGB information. Beyond the mere geometric information, the provided models contain semantic information via their layered structure. Two separate requirements were set in this project. For the 2D floor plan reconstruction, the final result is a JSON file that contains the outlines of all objects in the floorplan as start and end coordinate points in 2D. Since the methods of this article allow us to, we go one step further and can output not only lines, but callable instances. This feature is of particular relevance when speaking about a Scan-to-Floorplan process, was however not demanded by the workshop. For the 3D model reconstruction, the submission must be in OBJ or DXF files, representing the full 3D geometries of the scanned facilities, including semantics.

3 Implementation

3.1 Preprocessing, training data

To ensure good predictions for this specific dataset, instead of working with one of the well-established benchmark datasets (such as the S3DIS dataset [4] the 3D models of the provided data sets were leveraged to create a tailor-made ground truth dataset. The approach used in this is similar to the ones used in the DURAARK project[2] and [5]. In this work, all geometries belonging to the same semantic layers in the provided 3D models were first exported as single 3D models. To transfer the semantic information from model to point cloud, the distance to the closest surface is calculated per point and stored with the semantic label of the according 3D object. Then, each point is automatically annotated with the class label that it was found closest to. To minimize mislabeled points, a threshold of 10 cm is enforced. Thus, all points that do not lie within the defined threshold proximity to the next model surface are labeled with the 'rest' class. As the models are imperfect both regarding content and alignment, and, as standard architecture models, do not contain any furniture or other equipment, the resulting distribution across all classes (depicted in Figure 1 and Table 2) is biased toward the 'rest' class. Furthermore, to reduce overall file size and improve homogeneous data quality, all point clouds were down sampled using a voxel grid filter with a voxel size of 5cm.

Table 2: Overall class split of the model-based ground truth point clouds

class	wall	ceiling	floor	door	window	column	rest
overall %	23.9	17.1	16.0	1.1	1.4	1.0	39.4

3.2 Point cloud segmentation

In this step, based on the results of our automatically annotated data, we decide to implement point cloud segmentation by deep learning. The point cloud we want to validate on is mainly representing indoor spaces of buildings. In large-scale point cloud

[1] AutoCAD Drawing Interchange Format, https://www.loc.gov/preservation/digital/formats/fdd/fdd000446.shtml
[2] https://github.com/DURAARK/IFCPointCloud

segmentation, the S3DIS dataset [4] is widely used and evaluated on. In Semantic Segmentation on S3DIS [6], the performance of different neural networks on S3DIS dataset is listed and evaluated.

We chose the KPConv, one of the best performing architectures [7] to train our data on and later to perform inference. The performance comparison of KPConv and PointNet [8] for different categories in the S3DIS dataset, is listed in Table 3 and justifies our choice. The performance of KPConv is significantly better but quite different across categories. For ceilings, floors, and walls, the prediction is more precise than those categories of furniture. As the mean IoU (intersection over union) for ceilings, floors, and doors is more than 80%, we think the semantic information performed by KPConv could be usable in the further reconstruction process.

Table 3: Semantic segmentation IoU scores on S3DIS k-fold

	mIoU	ceiling	floor	wall	beam	column	window	door	chair	table	book	sofa	board	clutter
KPConv	70.6	93.6	92.4	83.1	63.9	54.3	66.1	76.6	57.8	64.0	69.3	74.9	61.3	60.3
PointNet	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	42.0	54.1	38.2	9.6	29.4	35.2

We project all points of the point cloud into the X-Y plane, assuming that the walls' locations will show a high point density. Figure 3a illustrates the high point density areas (using a filter for visualizing purposes only), thereby showing that they coincide mainly with the proper wall locations. Figure 3b on the other hand shows the projection of only the points predicted as walls by our trained KPConv algorithm. Figure 3 as a whole shows the comparison between the two described point cloud versions. The result in Figure 3b is much cleaner and less noisy when using only the predicted wall points, especially when there is lots of furniture present in the indoor spaces. The furniture surfaces make it hard to extract geometric information from the unprocessed point cloud for geometric reconstruction. The downstream algorithms will only use the predicted wall points and in the projection in Figure 3b.

(a) (b)

Figure 2: Comparison between projected wall points into the X-Y plane. (a) all points (with the applied point density filter for visualization), (b) only predicted wall points

3.3 Method 1: Wall and room detection with line processing

Line detection and space separation in 2D

The predicted wall points are used to fit optimal lines using random sampling and a voting scheme. This method commonly known as RANSAC relies on defining a threshold to set inliers and outliers before extracting a final set of lines. We use the MLESAC algorithm [9], one variant of RANSAC provided by the Point Cloud Library [10] to detect lines in 2D plane. The lines span the underlying 2D space into an oversegmented irregular grid structure.

Then a ray-tracing method [11] is used to determine if adjacent grid elements are separated by point groups (walls). Same labels are assigned to the grid elements for which the ray was not intersected by predicted wall points. The grid elements are then merged based on the assigned label and assembled to polygons. This process is illustrated in Figure 3. The merged polygons can either coincide with the underlying rooms or represent the void space between two wall surfaces. As can be see, the result of merging polygons for room spaces looks good. However, it is hard to merge the small polygons correctly. There are two main reasons here: 1) not all wall points are captured when laser-scanning because of occlusion; 2) the prediction of wall points is not perfect. The

wall points in 2D are still noisy and this makes it difficult to fit proper lines. Therefore, our results would need further refinement in future works to get more precise room and wall polygons. Following the aim we set out to attain, we nevertheless use the results for the next steps.

We notice that in Figure 3g the lines resulting from wall polygons are a part of a single uniquely identified wall instance (smart wall lines). The additional lines (from the room polygons) are however not part of any wall instance. The process could be enhanced in future to convert the room outer bounds to smart wall instances too. As shown in Figure 3h the midlines are only generated for the wall instance polygons. Due to this the outer walls are missing in the majority of the cases. Further algorithm tuning in upstream components is needed to extend these results.

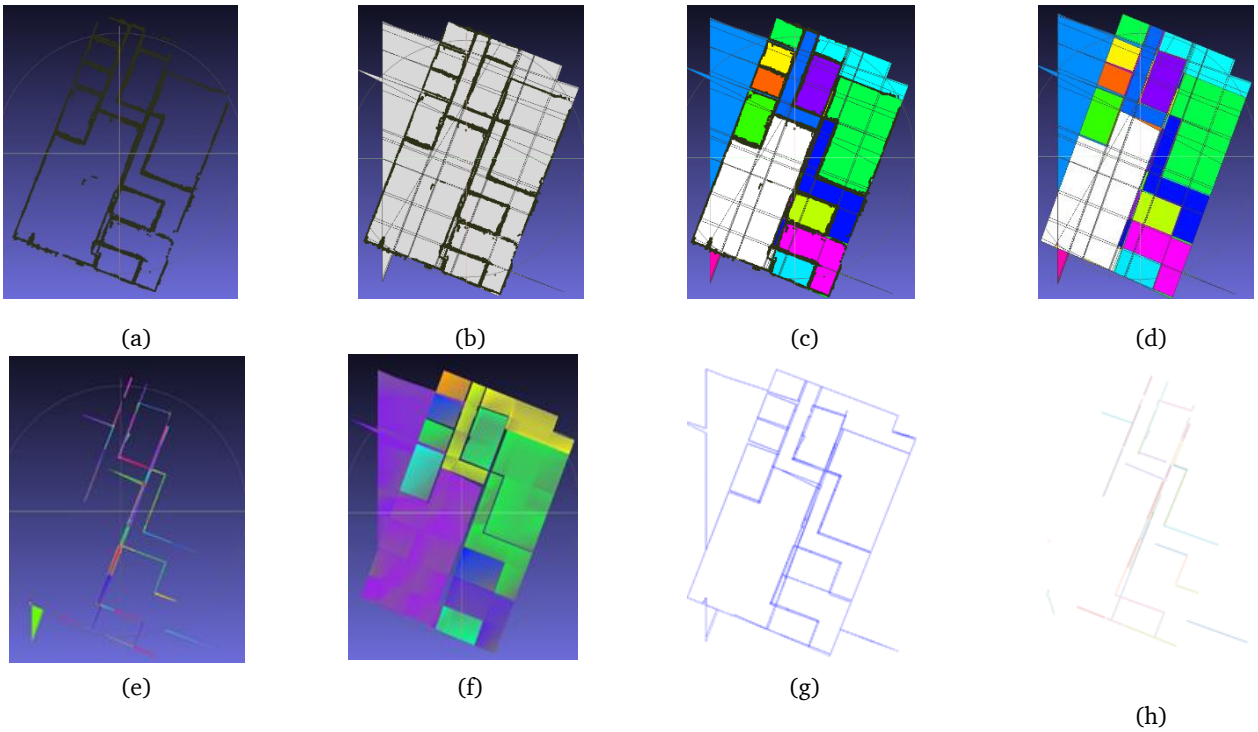


Figure 3: Line detection and space separation in 2D. (a) input point cloud of walls in 2D; (b) oversegmented plane; (c) polygon merging by ray-tracing; (d) polygon result (e) filtered and merged wall polygons based on a threshold of 5m^2 (f) Filtered and merged room polygons (g) resulting floor plan from wall and room polygons bounds (h) wall polygon midlines for 3D reconstruction

The technique of separating walls from rooms relies on the hypothesis that polygons in/between and around walls are considerably smaller than polygons spanning rooms. A threshold value is set to separate the two polygon classes based on their area. A visual optimization for the threshold value was performed and yielded best classification results for a 5 m^2 threshold (see Figure 3 e and f).

Floorplan generation and polygon midline extraction

The polygons from both walls and rooms (see Figure 3 e and f) are used in different ways for 1. a smart instance floor plan generation and 2. an output usable for the 3D reconstruction (midlines with start and end point). For the first task the outer bounds of the polygons are of interest to give an instance view of the walls and their thicknesses. For that, the wall polygons are merged based on adjacency and alignment along the principal component direction and then their outer boundaries are printed as instances into a vector format. SVG or PDF formats offer the attachment of limited semantics thereby offering a reasonable smart format for information transfer in 2D. Since not all walls are perfectly detected/merged the outer bounds of the room polygons are printed as well (see Figure 3 g). These additional lines serve as an improved visualization only and are not smart wall instances.

The 3D reconstruction used in this article relies on knowing the midline path of the wall instances. Similarly, to the 2D process, the adjacency and alignment along the principal component direction, serve as criteria for extracting the major midlines for the

polygons. The midline computation is limited to rectangular and triangular polygons only. Considering more complex polygon shapes would require more intensive computing. The midlines are encoded as start and end points.

3.4 Method 2: Wall detection with image processing

Point cloud rotation

Alternatively, to method 1 the wall midlines serving a subsequent 3D reconstruction can be extracted not via line processing but via image processing. The resultant predicted walls from KPConv, should be segmented into single surfaces. This step is conducted by projecting points on the horizontal plane (*XY*), creating the binary image of the points, and applying a kernel to detect the vertical and horizontal lines (walls). Since these kernels have been specified for detecting horizontal and vertical lines, the corresponding walls of the point cloud should be also aligned in this direction. Hence, a pre-processing step is required to make the point cloud align coordinate axes. Note that in the point cloud of buildings, the variance of points in all directions are close, and principal component analysis (PCA) cannot be simply used. To address this issue, an optimization problem is defined to determine the minimal axis-aligned bounding box. As the first step, the point cloud is de-noised by the k-d tree and k-nearest neighbor search to ensure that the bounding box is calculated correctly.

Figure 4 a and Figure 2 b show the point cloud of walls after projection on the *XY* plane and their corresponding axis-aligned bounding box (AABB). As can be seen, the AABB has the minimum area only if the point cloud within its bounding box is aligned with coordinate axes. Based on this observation, the rotation angle for which the area of the AABB is minimized is the angle that makes the point cloud align coordinate axes. The objective (loss) function of this optimization problem can be defined as below:

$$To\ minimize: A(\theta) = l * w \qquad 0 \leq \theta \leq \pi/2 \tag{1}$$

where l and w are the length and width of the AABB after a rotation of θ.

Since the parameter θ cannot be directly seen in the objective function, derivative-based optimization algorithms cannot solve this problem. Hence, particle swarm optimization (PSO) [12], as a metaheuristic and derivative-free algorithm, is used.

(a) (b)

Figure 4: The projected point cloud of walls: (a) the point cloud is not aligned with the coordinate axes (larger AABB); (b) the point cloud is aligned with the coordinate axes (smaller AABB)

Image processing step

Once the point cloud is rotated a binary image is generated with a grid side length of 5 mm by simply projecting the points of the predicted walls in the horizontal plane, as illustrated in Figure 6a.

Subsequently, and similar as done by [13], several iterations of a morphological dilation with a structural element (S) with a rectangular shape of size 10 x 10 are applied to join small blobs that are close to each other and which conform larger objects. This step leads to the results shown in Figure 6b, in which the two sides of the walls are now connected in a single element. Thereafter, and as the point cloud is already aligned with the XY axis, horizontal and vertical blobs are extracted applying a convolution with a vertical and horizontal kernel over the binary image. The resulted blobs, presented in Figure 6c, can be used as a mask to filter the separated instances of walls in the original point cloud, as shown in Figure 6d.

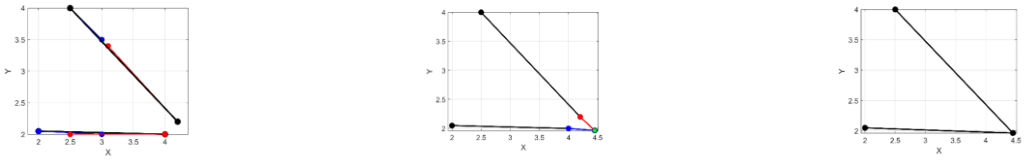
(a) (b) (c) (d)

Figure 5: Separating wall blobs: (a) Binary image result of the projection of the point cloud in the horizontal plane; (b) same image after dilation operation with the rectangular kernel; (c) separated horizontal (left) and vertical (right) blobs after linear kernel; (d) separated wall instances in different colors with their corresponding AABBs

This method assumes Manhattan World and disregards possible diagonal walls. However, after detecting the horizontal and vertical aligned walls, the rest of the walls can be filtered out and treated independently. A further improvement would also allow the method to distinguish the correct width of each wall, separate wall sections if they have different widths, and deal with walls that are only scanned by one side.

Line merging

The segmented walls from both the line and image processing step might have overlaps and not be completely connected as well. Therefore, a post-processing step is required to merge the walls on the same alignment and connect the neighboring walls. To this end, the AABB of each wall (line segment) is calculated and extended as much as a value of e_{bm}; using these bounding boxes, the collision of walls can be checked. If two walls have a collision, their normal vectors are almost in the same direction and have a distance lower than a value of threshold t_d, those walls are selected for merging. The merging process is conducted by calculating a new AABB. The new wall will be the diameter of this bounding box which normal is in the same direction as the original walls. To connect walls, the collision of walls is checked again, however, the walls that have a collision and are not almost parallel are considered. Next, the intersection point of the walls is calculated and new walls are created. As the last step, these lines are merged again to form single lines. This process can be briefly seen in Figure 6.



(a) (b) (c)

Figure 6: Post-processing: (a) merging walls; (b) connecting walls; (c) merging walls after connecting

3.5 Door and window detection

During the process of laser-scanning facilities, doors are usually open. This and the fact that the glass window surfaces do not reflect the laser beam, leads to special characteristics in the point cloud; windows and doors are void point areas in walls and their detection must rely on alternative processing steps than what was considered up until now in this paper. The door and window detection in this approach starts by using the wall coordinate from the previous step for prior knowledge. In the original point cloud, we find the points whose distance to the wall candidate is smaller than a threshold. In our experiment, we use the threshold 0.5m, assuming there is not any wall that is thicker than 0.5m in these facilities. Then we project the found points to the vertical plane of the wall. In the plane, doors and windows should identifiable by a human eye as holes. Apart from doors and windows, if the wall is occluded by some furniture, there would also be some furniture holes. But usually, these holes have different shapes than windows and doors. In our experiment, we consider all the windows and doors in the facility to be rectangular. The next step involves the calculation of a negative space on this plane. Then we cluster the points on this plane and fit rectangles to these point clusters. By filtering the non-rectangular point cluster out, we can get the window and door on the plane. We can also use more prior knowledge to filter more rectangles out, like doors should be higher than 2m, doors are connected to the floor, etc. The whole process is illustrated in Figure 7.

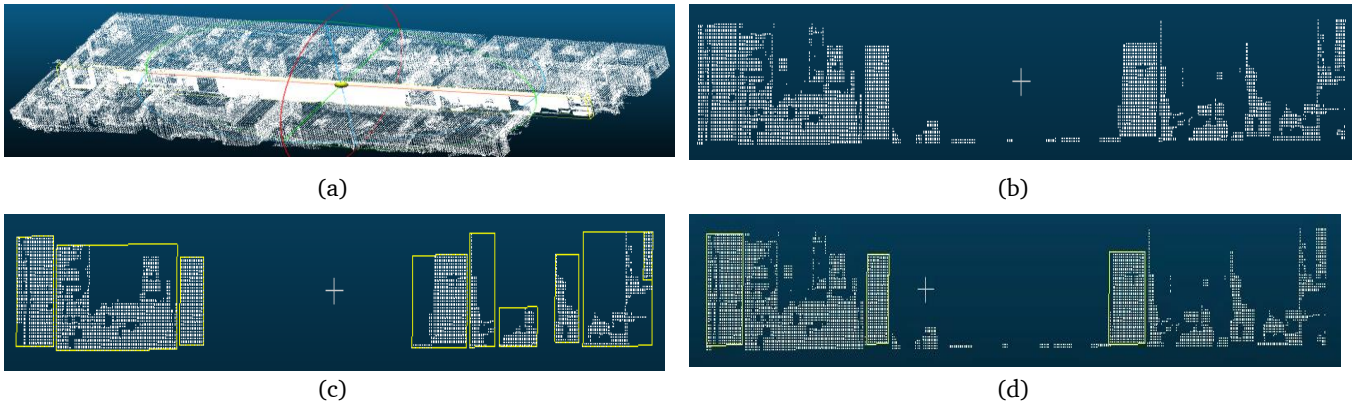


Figure 7: Window and door detection: (a) find wall in original point cloud; (b) calculate void space of the plane; (c) find point clusters of the void space; (d) remove outliers of each cluster and find the rectangles of window and door void space, in this case three doors are found

3.6 3D Reconstruction in Revit

To finally create the 3D model the modelling capabilities of Revit are used. With the 2D coordinates of the midline start and end points of the walls, doors, and windows, a program written with the Revit API allows the creation of a 3D model. Among the different possibilities for creating a 3D model (such as IfcOpenShell and xBIMtoolkit), Revit was chosen due to the possibility it offers to insert customizable family types, automatic wall joins (at intersections) and automatic DXF and IFC export. In Revit, warnings caused by overlap during wall creation are automatically disabled. To create the doors and windows, the method takes the middle point of the element and finds the closest wall element to it, assuring that the element has the same normal vector as the wall. Moreover, the doors and windows are not created if their position is close to a wall border. After the resulted model is automatically exported in DXF and IFC from Revit, IfcConvert is used to convert the IFC file to OBJ format with materials. Figure 8 gives an overview of the proposed 3D reconstruction pipeline.

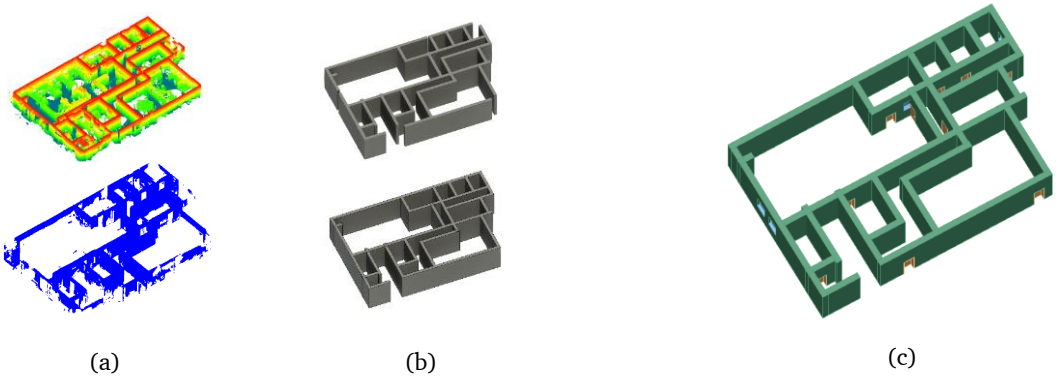


Figure 8: 3D Reconstruction Overview: (a) Original point cloud (top), point cloud after wall prediction and automatic rotation; (b) Wall instances after image processing (top), result after line merging (bottom); (d) Final 3D model after adding detected doors (in brown) and windows (in blue) with custom family type in Revit and exporting to DXF format

4 Conclusion

This paper presents a hands-on end-to-end workflow from raw point clouds to a usable format such as callable SVG instance floor plans and 3D IFC models for building engineers and architects. Despite being limited by accuracy and class diversity, results of the paper resemble manual reconstruction outcomes, thereby hinting at the numerous potentials full automatic Scan-to-BIM/Scan-to-floorplan can bring to the AEC community. A quantitative assessment of the usability of the results by the community is out of scope for this work.

Nevertheless, the necessity of articles such as ours becomes apparent when scrutinizing the limitations of the numerous technical components and their direct down-stream implications in the whole work chain. The final result of a 2D or 3D reconstruction is highly dependent on the accuracy and coverage of class diversity of each upstream component. End-to-end approaches without enough focus on either accuracy or class diversity are likely to be rejected by the AEC community for practical reasons. Despite the considerable attention AI methods have gained in the past years for point cloud semantic segmentation, the AEC community relies on the further development of domain-based algorithms to complete the work chain. Concluding this work, the authors would recommend developing additional end-to-end methods to prove the efficacy of automation in the AEC community.

References

[1] Bosché, F., M. Ahmed, et al. (2015). “The value of integrating Scan-to-BIM and Scan-vs-BIM techniques for construction monitoring using laser scanning and BIM: The case of cylindrical MEP components.” Automation in Construction 49: 201-213.

[2] Adán, A., B. Quintana, et al. (2018). “Scan-to-BIM for secondary building components.” Advanced Engineering Informatics 37: 119-138.

[3] Laing, R., M. Leon, et al. (2015). “Scan to BIM: the development of a clear workflow for the incorporation of point clouds within a BIM environment.” WIT Transactions on The Built Environment 149: 279-289.

[4] I. Armeni et al., "3D Semantic Parsing of Large-Scale Indoor Spaces," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1534-1543, doi: 10.1109/CVPR.2016.170.

[5] Collins, F. “Encoding of geometric shapes from Building Information Modeling (BIM) using graph neural networks”, Technische Universität München, 2021.

[6] Semantic Segmentation on S3DIS. https://paperswithcode.com/sota/semantic-segmentation-on-s3dis

[7] Thomas, Hugues, et al. "Kpconv: Flexible and deformable convolution for point clouds." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.

[8] Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[9] Torr, P. and Andrew Zisserman. “MLESAC: A New Robust Estimator with Application to Estimating Image Geometry.” Comput. Vis. Image Underst. 78 (2000): 138-156.

[10] Rusu, Radu Bogdan, and Steve Cousins. "3d is here: Point cloud library (pcl)." 2011 IEEE international conference on robotics and automation. IEEE, 2011.

[11] Laine, Samuli, and Tero Karras. "Efficient sparse voxel octrees–analysis, extensions, and implementation." NVIDIA Corporation (2010).

[12] Kennedy, J. and R. C. Eberhart (1997). A discrete binary version of the particle swarm algorithm. *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*, IEEE.

[13] Vega, M., Braun, A., Bauer H., Noichl F., & Borrmann, A. (2021) "Efficient Vertical Object Detection in Large High-Quality Point Cloud of Construction Sites", European Conference on Computing in Construction