Polygonal Reconstruction of Building Interiors from Cluttered Pointclouds

Inge Coudron(✉), Steven Puttemans(✉), and Toon Goedemé(✉)

EAVISE Research Group, KU Leuven,
Jan Pieter De Nayerlaan 5, Sint-Katelijne-Waver, Belgium
{inge.coudron,steven.puttemans,toon.goedeme}@kuleuven.be

Abstract. In this paper, we propose a framework for reconstructing a compact geometric model from point clouds of building interiors. Geometric reconstruction of indoor scenes is especially challenging due to clutter in the scene, such as furniture and cabinets. The clutter may (partially) hide the structural components of the interior. The proposed framework is able to cope with this clutter by using a hypothesizing and selection strategy, in which candidate faces are firstly generated by intersecting the extracted planar primitives. Secondly, an optimal subset of candidate faces is selected by optimizing a binary labeling problem. We formulate the selection problem as a continuous quadratic optimization problem, allowing us to incorporate a cost function specifically for indoor scenes. The obtained polygonal surface is not only 2-manifold but also oriented, meaning that the surface normals of each polygon are consistently oriented towards the interior. All adjacent and coplanar faces that were selected, are merged into a single face in order to obtain a final geometric model that is as compact as possible. This compact model of the room uses less memory and allows for faster processing when used in virtual reality applications. The method of Nan et al. was used as a starting point for our proposed framework. Finally, as opposed to other state-of-the-art interior modeling approaches, the only input that is required, is the point cloud itself. We do not rely on viewpoint information, nor do we assume constrained input environments with a 2.5D or, more restrictively, a Manhattan-world structure. To demonstrate the practical applicability of our proposed method, we performed various experiments on actual scan data of building interiors.

1 Introduction

As we live in a 3D world, performing most of our activities in indoor environments, indoor scenes are familiar and essential in everyone's life. In the virtual world, 3D models of indoor scenes are used everywhere from 3D games to interior design. With the fast development of various augmented and virtual applications, the demand for realistic 3D indoor models is growing rapidly. However, obtaining such 3D models is quite difficult and challenging as opposed to outdoor scene modeling. The amount of clutter in outdoor scenes hiding the geometry of a

L. Leal-Taixé and S. Roth (Eds.): ECCV 2018 Workshops, LNCS 11129, pp. 459–472, 2019.
https://doi.org/10.1007/978-3-030-11009-3_28

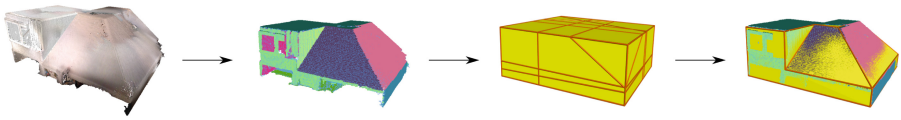


Fig. 1. Overview of the proposed pipeline. First the planar primitives are extracted from the pointcloud (Subsect. 3.1). Then candidate faces are generated (Subsect. 3.2). Finally an optimal subset of candidate faces is selected (Subsect. 3.3).

Fig. 2. Example of clutter inside a room.

building is rather low compared to indoor scenes. In indoor scenes, the clutter ranges from planar surfaces such as closets and furniture to highly irregular objects such as plants. Therefore, it is often not relevant to apply outdoor reconstruction algorithms to indoor scenes [2]. Nonetheless, we were able to adapt an outdoor reconstruction algorithm for the robust geometric modelling of indoor scenes (Fig. 2).

In this work, we focus on the reconstruction of a single room. We assume each room has been scanned separately. Therefore, when processing a complete building, each room will be processed sequentially. Hence, the problem of room segmentation is not considered here. Our goal is to reconstruct the basic geometry of the room from a point cloud obtained from a 3D consumer camera. The fact that we work with consumer cameras imposes a few extra challenges. First of all, the quality of the point clouds obtained with these scanners is often rather noisy. Secondly, the use of these consumer cameras may result in pose estimation errors during tracking which produce ghost-like double walls. Hence, the reconstruction algorithm must be robust enough to be able to cope with these imperfections. Furthermore, the scans contain a lot of clutter such as tables, doors etc. These surfaces should be ignored as well, as we are only interested in the outer geometry of the room as if it was an empty room. To obtain a compact geometric model, we approximate the room by as few piecewise planar surfaces as possible. A more compact model of the room uses less memory and allows for faster processing when used in virtual reality applications.

Our pipeline considers as input a 3D point cloud from a single room and produces as output a lightweight watertight 2-manifold oriented mesh. It requires

no further information such as viewpoints. An overview of the proposed pipeline is shown in Fig. 1. Firstly, candidate faces are generated by intersecting the extracted planar primitives. Secondly, an optimal subset of candidate faces is selected. The selection of the optimal subset of candidate faces is in fact a binary labeling problem, but we cast it as a continuous quadratic programming problem in order to be able to incorporate an indoor specific cost function. The result is a 2-manifold and oriented mesh that can compactly describe the outline of a cluttered indoor scene even with slanted walls or sloped ceilings.

The remainder of this paper is organized as follows. In Sect. 2 we give an overview of the current state-of-the-art in 3D interior reconstruction and geometric modeling from point clouds. Next, in Sect. 3 we explain our proposed pipeline. The results are discussed in Sect. 4. Finally, a conclusion and future work is given in Sect. 5.

2 Related Work

Most of the research done on reconstructing building interiors from point clouds follows a 2D approach. They either assume simple vertical walls [3,4,6] or, even more stringent, a Manhattan world [5,7–9]. In either case, the final model is produced by extruding a 2D floorplan with respect to the ceiling height. Some of these methods are restricted to piecewise linear floorplans [5], while others are able to capture rounded walls as well [7]. These assumptions limit the number of real-world architectures that can be reconstructed significantly. Therefore, more and more research is being done on interior reconstruction in 3D instead of 2D. One of the most promising techniques, is the work of [2]. They first determine whether the detected planar patches belong to permanent components (e.g. walls, ceiling or floor) by reasoning on a graph-based scene representation. Then the permanent components are used to build a 3D linear cell complex that is partitioned into separate rooms through a multi-label energy minimization formulation. However, this requires the prior knowledge of the scanning device poses, which is not always available. Therefore, in our approach we do not want to rely on this kind of prior knowledge.

In this paper, we want to create a geometric model of a scanned room. The model must be simple while being powerful enough to explain the scanned point cloud data. As interior rooms are mostly planar, we choose to fit a piecewise planar model to the room. In literature, reconstructing piecewise planar models from building exteriors is a well-studied problem [1,10,11]. However, methods such as [11] do not result in a watertight mesh, which is in fact necessary for correct shadow mapping in virtual reality. Furthermore, many of these methods ignore the problem of occlusions and missing regions due to clutter and are therefore not suited for indoor scenes. However, classifying planar patches as either clutter or structural components and simply removing them is not an option. Sometimes the structural components are completely occluded by clutter, so removing these planar patches would result in missing planes making it impossible to reconstruct a closed surface. Therefore we choose not to remove

the clutter and adapt the algorithm from [1] to better handle the clutter. The main reason why this algorithm is not directly suited for indoor scenes, is that it assumes that all points belong to a piecewise planar object. However, the indoor scenes also contain clutter, which should be ignored as much as possible. Furthermore, their hypothesis generator was not robust enough for indoor scenes. In the next section we will explain how we adapted this algorithm.

3 Proposed Method

Our proposed method takes as input a point cloud of an entire room, including furniture and other objects and outputs a piecewise planar model of the room. As shown in Fig. 1, first the planar primitives are extracted from the pointcloud. Then, candidate faces are generated. Finally, an optimal subset of candidate faces is selected.

3.1 Plane Extraction

To detect planar primitives in the 3D point cloud, we use the standard RANSAC based shape detection from [12]. As the computational cost of the algorithm is relatively high for large point clouds, we first downsample the input cloud using a voxelgrid filter at a resolution of 5 cm. The result from this RANSAC based shape detection is a set of planar patches $P = \{p_i\}$. Each patch p_i consists of a set of points which lie within a distance ϵ from the best fit plane through these points.

One of the most common problems in indoor scene reconstruction from planar patches is that some of the boundary walls are undetected due the presence of clutter, door openings etc. Part of this problem can be alleviated by adding the planes from the bounding box around the point cloud to the set of planar patches. The bounding box planes are defined as the front, back, top, bottom left and right side of the bounding box. If we cannot find a plane that is close to the bounding box plane, the latter is added to the set of detected planes. By adding these planes to the set, we can ensure that it is always possible to at least generate a watertight mesh.

To check if a plane is close to a bounding box plane, we use the algorithm proposed in [10]. First we compute the angles between the detected planes and the bounding box planes. Then, starting from the plane with the smallest angle, we test if two conditions are met. On the one hand, the angle between the planes must be lower than some threshold θ_t. On the other hand, more than a specified number of points n_t must lie on the bounding box plane. If these two conditions are met we identify the detected plane as the corresponding bounding box plane. If not, we add the bounding box plane to the set. In our experiments we chose $\theta_t = 10°$ and $n_t = 20\%|p_i|$ where $|p_i|$ denotes the number of inliers of the planar patch p_i.

3.2 Candidate Face Generation

As explained by Nan et al., the hypothesis generator of [1] might introduce degenerate faces due to the limit of floating point precision. A face is called degenerate when one or more of its edges is no longer connected to an adjacent face. This degeneracy usually makes the manifold and watertight constraints impossible to be satisfied. When applying their hypothesis generator on our indoor scenes it became apparent that this is indeed a problem. Therefore, we implemented a new hypothesis generator that is able to cope with the limitations of floating point precision and does not suffer from these degeneracies.

From the previous step, we were able to identify each of our bounding box planes. These planes are used to build an initial polyhedron. As this polyhedron is always convex, the subsequent slicing of the polyhedron with the detected planes will again result in convex polyhedrons. To dynamically slice a convex polyhedron, we make use of the Sutherland-Hodgman clipping routine [13]. This routine is quite simple and very efficient. Furthermore, it can be extended to account for numerical robustness (see Fig. 3).

The two adjustments that are required to make the algorithm numerically robust are the following. First of all, the intersection point calculated from point A to point B will be slightly different than the calculation from point B to point A due to the limited precision of floating points. In order to avoid numerical issues because of this, we have to compute the intersection in a consistent manner. This is achieved by ordering the two points lexicographically: first x-coordinates are compared, if they are equal, y-coordinates are compared. Secondly, another source of numerical issues arises when checking if a point lies in front or behind the plane. Therefore we slice the plane with so-called thick planes. When a vertex of the polygon lies within a certain distance of the plane, it is as if the plane cut the polygon at this vertex.

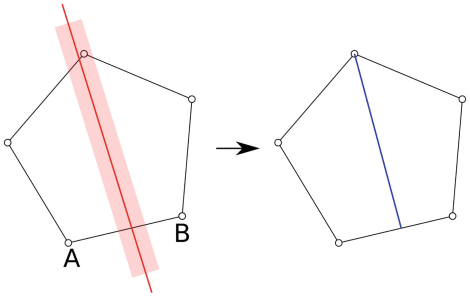


Fig. 3. Clipping a convex polygon by a plane.

To construct all candidate faces, we slice the polyhedron subsequently with each of the detected planes. Each face of the polyhedron is cut by the plane using the previously mentioned clipping routine. The result is a collection of faces

belonging to the polyhedron on the positive side of the cutting plane and a set of faces belonging to the negative side. The cross section from this cut is added to both the front and back polyhedron. Each of the polyhedrons that is constructed as a result of this cut, will be subsequently cut by the next detected plane. In the end, we obtain a collection of polyhedrons that represent the candidate faces.

As opposed to the hypothesis generator from [1], we also do no longer have the problem of generating long and very thin candidate faces. By changing the distance to determine if a vertex is on the plane, we can control how long the smallest edge will minimally be. Furthermore, Finally, we do not create any degenerate faces. A face is degenerate when it contains an edge that is not connected to any other face. Selecting such a face can never lead to a manifold mesh and is therefore considered redundant. Adding these faces to the optimization problem, makes the problem size unnecessary larger.

A comparison between a set of candidate faces generated by the method in [1] and ours is shown in Fig. 4. From left to right right this image shows the input cloud, the candidate faces generated by [1], the reconstructed model from their hypothesis, the candidate faces generated by our algorithm and the reconstructed model from our hypothesis. As we can see in the first row of this figure, the top and front plane were not detected by the RANSAC approach as the number of inliers was too low. With our hypothesizing strategy we were able to cope with these missing planes. As apposed to [1] we do not only rely on the planes detected by RANSAC, but we start from a bounding volume which we subsequently slice with the detected planes. Therefore an approximation of these missing planes is added automatically to the candidate set. In Fig. 4(b) a circle is drawn around a face that was incorrectly cut, resulting in a reconstructed model that is not able to capture the actual geometry of the room.

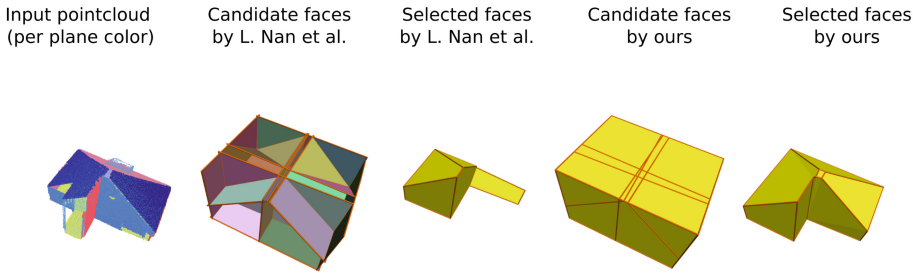
3.3 Optimal Face Selection

Energy Terms. Each face f_i is described by two variables x_i and x_{ir}. The subscript r stands for reverse, meaning that we either select the face in its counter clockwise orientation (i.e. $x_i = 1$) or its clockwise or reversed orientation (i.e. $x_{ir} = 1$). If the face is not selected, both variables must be zero. Our objective function consists of three energy terms: data selection, model complexity and interior coverage.

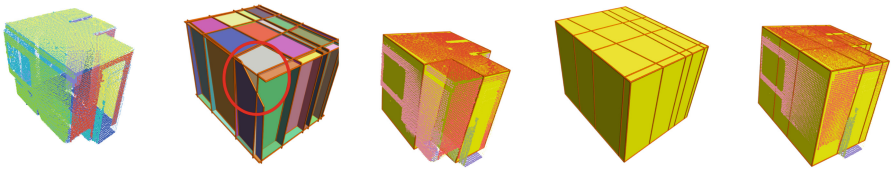
First of all the data selection term evaluates how many of the points, that belong to the planar patches, are selected:

$$E_d = -\frac{1}{|P|}\sum_{i=1}^{N}(x_i + x_{ir}) \cdot inliers(f_i), \tag{1}$$

where P is the total number of inliers from the detected planar patches and $inliers(f_i)$ is the number of inliers in face f_i and N is the total number of faces that was generated. This term favourizes selecting as many faces where we found planar patches as possible. However, these planar patches also contain clutter such as closets, tables, doors etc. The data fitting term will try to select the



(a) Missing planes.



(b) Degenerate clipping.

Fig. 4. A comparison between the candidate face generated by [1] and our hypothesis generator. From left to right right: the input cloud, the candidate faces generated by [1], the reconstructed model from their hypothesis, the candidate faces generated by our algorithm and the reconstructed model from our hypothesis.

faces containing these objects as well. Therefore, we need to define some other energy terms to counterbalance this effect.

The second energy term we define, is a measure for the complexity of the resulting model. Selecting faces corresponding to clutter often results in gaps or protrusions. Therefore, we define the complexity of the model as the number of boundary edges that were selected. An edge is a boundary edge if the two faces adjacent to this edge do not lie on the same plane. The less boundary edges are selected, the simpler the model will be. Hence, this term discourages selecting the faces corresponding to clutter as that introduces more boundary edges. The complexity term can be written as follows:

$$E_c = \frac{1}{|E|} \sum_{i=1}^{N} \sum_{j=1, i \neq j}^{N} (x_i + x_{ir})(x_j + x_{jr}) \cdot corner(f_i, f_j), \tag{2}$$

where E is the total number of edges and $corner(f_i, f_j)$ indicates whether the edge formed by the faces f_i and f_j is a boundary edge (i.e. $corner(f_i, f_j) = 1$) or not. However, in indoor scenes this is not sufficient because the clutter often occludes the actual room structure. As a result, the faces corresponding to the underlying structure do not contain any inliers while the clutter does. And so the first energy term will still dominate.

We define a third energy term, namely the interior coverage, that tries to compensate for the fact that the clutter results in missing data. In indoor scenes, we want as many points on the inside of the obtained model. So the uncovered regions (i.e. regions with more points on the outside than on the inside) should be as low as possible. To measure the coverage of a face f_i we first project all the points that lie in front onto f_i. Hence for x_i we project the points on the positive side of the supporting plane onto f_i and for x_{ir} we project the points on the negative side onto f_i. Then, we calculate the 2D alpha shape [14] from the projected points. The alpha shape creates a bounding area that envelops the set of projected 2D points. By changing the alpha parameter, you can manipulate the alpha shape object to tighten or loosen the fit around the points to create a non convex region. The alpha shape provides a good measure for the coverage of the candidate face by the projected points. So even if a face f_i from a structural component has no inliers due to occlusion, there will be a lot of points in front of it. Therefore, it can still provide a high coverage and gets a higher chance at being selected as well.

$$E_i = -\frac{1}{|area(M)|}\sum_{i=1}^{N} x_i \cdot (area(f_i) - area_P(f_i)) + x_{ir} \cdot (area(f_i) - area_N(f_i)), \quad (3)$$

where $area(M)$, $area(f_i)$, $area_P(f_i)$ and $area_N(f_i)$ denote the surface areas of the bounding box, a candidate face f_i, and the area of the alphashape mesh constructed from the points on the positive or negative side of f_i respectively.

Optimization. By minimizing a weighted sum of the above mentioned energy terms, we can find the optimal subset of candidate faces. Remember that for the selection of each face f_i we defined two variables x_i and x_{ir}. They indicate in which orientation the face is selected. As we can select only one orientation, these variables are mutually exclusive. The mutual exclusion can be enforced by adding an extra term to the objective function, namely $x_i x_{ir}$. This term drives the solution towards either one or both variables of being zero. The objective function is thereby formulated as follows:

$$E = \lambda_d \cdot E_d + \lambda_c \cdot E_c + \lambda_i \cdot E_i + \sum_{i=1}^{N} x_i x_{ir} \quad with \quad 0 \leqslant x_i, x_{ir} \leqslant 1 \quad (4)$$

By defining two variables per face, we can ensure that the reconstructed model will have a consistent orientation. The orientation of two adjacent faces is consistent if the two vertices of the common edge are in the opposite direction (see Fig. 5). Note that the vertices of each edge are lexicographically ordered. So for each face f_i, adjacent to an edge e_j, we can determine its direction with respect to this edge. For this, we define a function $sign(x_i, e_j)$ and $sign(x_{ir}, e_j)$ as follows:

$$sign(x_i, e_j) = \begin{cases} 1 & \text{if } e_j \text{ and corresponding edge in } x_j \\ & \text{have the same direction} \\ -1 & \text{if } e_j \text{ and corresponding edge in } x_j \\ & \text{have opposite directions} \end{cases} \tag{5}$$

$$sign(x_{ir}, e_j) = -sign(x_i, e_j) \tag{6}$$

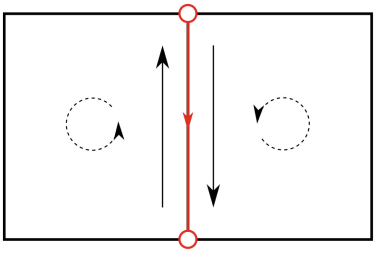


Fig. 5. Consistent ordering of the faces among an edge.

To ensure each edge will have consistently oriented faces, we define the following constraint:

$$\forall e_j : \sum_{f_i \in \mathcal{N}(e_j)} sign(x_i, ej) \cdot x_i + sign(x_{ir}, ej) \cdot x_{ir} = 0, \tag{7}$$

This constraint implies that when a face is selected with the edge in one direction, another face should be selected with the edge in the opposite direction.

To guarantee that the reconstructed model is 2-manifold either two or no faces must be selected. Therefore we define an additional constraint for each edge:

$$\forall e_j : \sum_{f_i \in \mathcal{N}(e_j)} x_i + x_{ir} \leqslant 2 \tag{8}$$

Thus the final optimization problem for selecting the best subset of candidate faces can be formulated as follows:

$$\min_{x_i, x_{ir}} \lambda_d \cdot E_d + \lambda_c \cdot E_c + \lambda_i \cdot E_i + \sum_{i=1}^{N} x_i x_{ir} \tag{9}$$

$$\text{s.t.} \begin{cases} \sum_{f_i \in \mathcal{N}(e_j)} sign(x_i, ej) \cdot x_i + \\ sign(x_{ir}, ej) \cdot x_{ir} = 0 \\ \sum_{f_i \in \mathcal{N}(e_j)} x_i + x_{ir} \leqslant 2 \\ 0 \leqslant x_i, x_{ir} \leqslant 1 \end{cases} \tag{10}$$

To obtain the final selection of candidate faces, we round the variables x_i and x_{ir} to the nearest integer. This is necessary because we are solving a continuous relaxation of the binary labeling problem. Therefore, the outcome will be close to

0 or 1 but not exactly. Each face for which either x_i and x_{ir} is 1 will be selected. The union of the selected faces comprises the final polygonal reconstruction.

Our method distinguishes from [1] in two aspects. Firstly, we implemented a hypothesis generator that is more robust for indoor scenes as explained the previous subsection. Secondly, our energy terms are able to better handle the missing or erroneous data as a result of occlusions and clutter in indoor scenes. A comparison between a reconstructed model obtained using the optimization problem of [1] and ours is shown in Fig. 6. As we can see despite the heavy clutter, our method is still able to detect the outer geometry of the room.

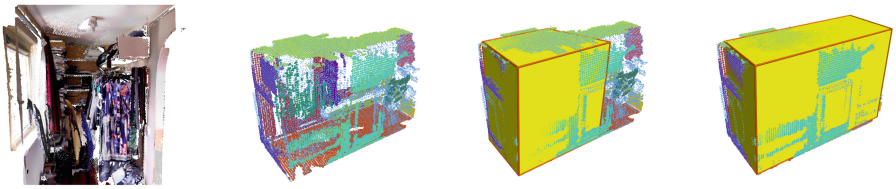


Fig. 6. Comparison between the optimization problem as defined in [1] and ours. From left to right: the cluttered indoor scene, the extracted planar primitives, the model reconstructed by [1], the model reconstructed by our method

4 Results

Our algorithm is implemented in C++ using the free Edition of the ALGLIB library for solving the quadratic optimization problem. To detect the planar primitives and construct the alpha shape meshes, we used the CGAL library [14]. We tested our pipeline on a set of 4 different real-world datasets from building interiors. All test were performed on a DELL XPS with an Intel Core i7 (1.8 GHz), 8 GB DDR3 RAM. Processing times for all models are given in Table 1 and vary from about 30 s to about 5 min for the complete pipeline. Note that our algorithm uses the free edition of ALGLIB and therefore runs on a single core. The planar primitive extraction is obtained using the RANSAC implementation in CGAL. For this we used the default parameter settings. The energy minimization depends on the weights λ_d, λ_c and λ_i that we fixed to 0.2, 0.3 and 0.3 respectively, which was determined empirically.

Table 1. Information on the environment (no. rooms) and overall processing time.

Dataset	#rooms	Processing time
House1	15	27 s
House2	17	313 s
Appartment1	12	38 s
House3	12	186 s

In Fig. 9 we show the output of our algorithm on 3 different datasets. The experiments show the advantages of our method. We were able to reconstruct the outer geometry of the rooms despite the clutter and missing or erroneous data. For example, if we take a closer look at the dining room from Appartment1 (see Fig. 7), we can see that the point cloud suffers from an extreme case of what we call a ghost-like double wall. Despite the erroneous data our method is able to correctly estimate the actual geometry of the room.

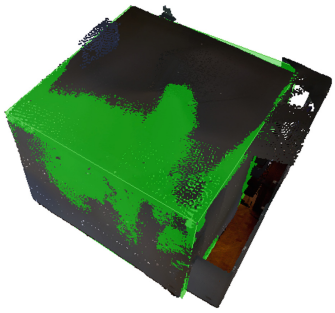


Fig. 7. A closeup from the dining room of Appartment1. The reconstructed model is shown in green. Despite the erroneous data the method is able to correctly estimate the geometry of the room. (Color figure online)

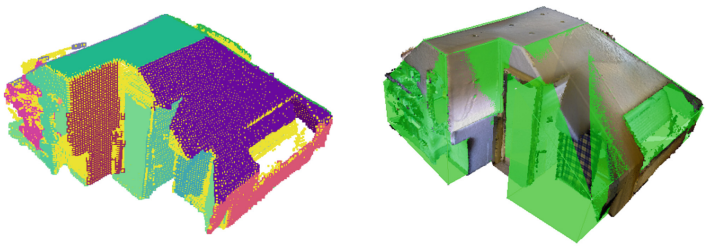


Fig. 8. A closeup from the bedroom on the second floor of House1. Due to a planar patch that was not detected, our method fails to reconstruct the correct geometry.

However, our method also has its limitations. We rely on the detection of planar patches from the structural components of the room to reconstruct its outer geometry. As we cannot guarantee that all planar surfaces are correctly extracted, our method will fail to correctly reconstruct the geometry in such cases as can be seen in Fig. 8. In this point cloud, the points that were not assigned to any planar patch are marked yellow. As we can see, there was no planar patch detected near the door of the bedroom. Therefore, our method was not able to correctly reconstruct the outer geometry of the room.

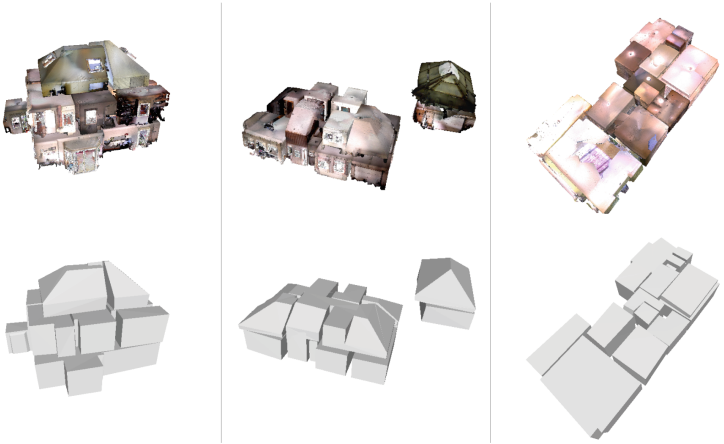


Fig. 9. Our method applied on different real life datasets: House1, House2 and Appartment1 respectively. From top to bottom: the cluttered indoor scenes, the model reconstructed by our method

In Fig. 10 we show a direct comparison between our method and the method from [1]. While their method is able to reconstruct most of the rooms, our method better describes the actual outer geometry. For example, in the close up from the living room, we can see that the method from Nan et al. Tries to overfit the clutter. The same problem was seen in the kitchen. Furthermore, some of the rooms were only constructed partially or not at all as a result of their non-robust hypothesis generator.

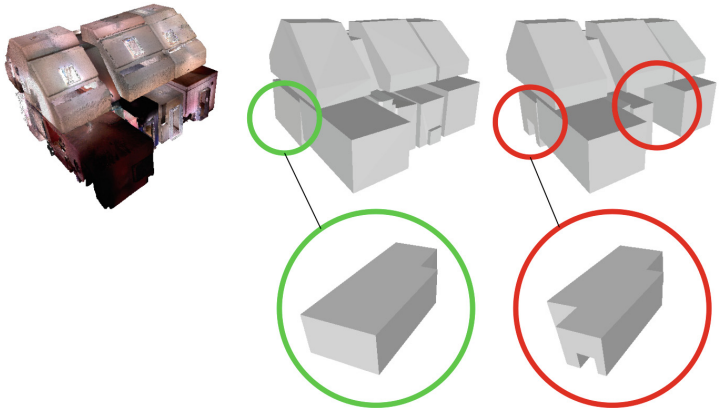


Fig. 10. Comparison between the results produced by our method and [1] on House4. From left to right: the cluttered indoor scenes, the model reconstructed by our method, the model reconstructed by [1]

5 Conclusion and Future Work

In this paper, we proposed a framework for reconstructing a lightweight polygonal surface of building interiors from cluttered point clouds. The method uses a hypothesis and selection strategy, in which candidate faces are firstly generated by intersecting the extracted planar primitives. Secondly, an optimal subset of candidate faces is selected by optimizing a binary linear programming problem. In this paper, we adapted the pipeline from [1] to make it more suitable for indoor scenes. As a first step, we implemented a hypothesis generator that is more numerically robust. Secondly, we reformulated the selection problem as a continuous quadratic optimization problem. The reformulation allowed us to incorporate a different cost function relevant for indoor scenes. Furthermore, the obtained polygonal surface is not only 2-manifold but also oriented, meaning that the surface normal of each polygon is consistently oriented towards the interior. This is especially interesting for rendering, where the shading depends on the correct orientation of the normals. Finally, as opposed to other state-of-the-art interior modeling approaches, the only input that is required, is the point cloud itself. We do not rely on viewpoint information, nor do we assume constrained input environments with a 2.5D or, more restrictively, a Manhattan-world structure.

The main limitation of our approach is that some planar surfaces that define the outer geometry of the room might still be undetected. In future work, we would like to explore the possibility to make the detection of planar surfaces more robust. Since the scenes were captured using an RGBD sensor, we can use the RGB images to detect lines as well. Next we can apply a RANSAC based method to detect planar surfaces from these line segments. The is especially suited for the missing planes due to windows or door openings which result in missing data in the depth images. To be able to better compare the results of the different reconstructions quantitavely, we will define a new metric based on the Intersection over Union. For the groundtruth mesh as well as the reconstructed mesh, we can create an occupancy grid in which each voxel is 1 if it is inside the mesh or 0 when it is on the outside. The Intersection over Union of both occupancy grids provides a good measure of the similarity of the reconstruction. If the reconstructed mesh is the same as the groundtruth mesh, the IoU will be 1.

References

1. Nan, L., Wonka, P.: PolyFit: polygonal surface reconstruction from point clouds. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2372–2380, October 2017
2. Mura, C., Mattausch, O., Pajarola, R.: Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. In: Computer Graphics Forum (2016)

3. Turner, E., Zakhor, A.: Floor plan generation and room labeling of indoor environments from laser range data. In: Proceedings of the 9th International Conference on Computer Graphics Theory and Applications: GRAPP, VISIGRAPP 2014, INSTICC, vol. 1, pp. 22–33. SciTePress (2014)
4. Oesau, S., Lafarge, F., Alliez, P.: Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. ISPRS J. Photogramm. Remote Sens. **90**, 68–82 (2014)
5. Mura, C., Mattausch, O., Villanueva, A.J., Gobbetti, E., Pajarola, R.: Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. Comput. Graph. **44**(C), 20–32 (2014)
6. Ochmann, S., Vock, R., Wessel, R., Klein, R.: Automatic reconstruction of parametric building models from indoor point clouds. Comput. Graph. **54**(C), 94–103 (2016)
7. Turner, E., Zakhor, A.: Watertight as-built architectural floor plans generated from laser range data. In: 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission (3DIMPVT), pp. 316–323, October 2012
8. Ikehata, S., Yang, H., Furukawa, Y.: Structured indoor modeling. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1323–1331, December 2015
9. Murali, S., Speciale, P., Oswald, M.R., Pollefeys, M.: Indoor Scan2BIM: building information models of house interiors. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6126–6133, September 2017
10. Li, M., Wonka, P., Nan, L.: Manhattan-world urban reconstruction from point clouds. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 54–69. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_4
11. Monszpart, A., Mellado, N., Brostow, G.J., Mitra, N.J.: RAPter: rebuilding man-made scenes with regular arrangements of planes. ACM Trans. Graph. **34**(4), 103:1–103:12 (2015)
12. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
13. Sutherland, I.E., Hodgman, G.W.: Reentrant polygon clipping. Commun. ACM **17**(1), 32–42 (1974)
14. Da, T.K.F.: 2D alpha shapes. In: CGAL User and Reference Manual, 4.12 edn. CGAL Editorial Board (2018)