Exploiting Indoor Mobile Laser Scanner Trajectories for Semantic Interpretation of Point Clouds

S. Nikoohemat [a*], M. Peter [a], S. Oude Elberink [a], G. Vosselman [a]

[a] Dept. of Earth Observation Science, Faculty ITC, University of Twente, Enschede, The Netherlands - (s.nikoohemat, m.s.peter, s.j.oudeelberink, george.vosselman)@utwente.nl

Commission IV, WG IV/5

KEY WORDS: Indoor Point Clouds, MLS Trajectory, Indoor Reconstruction, Opening Detection, Occlusion Reasoning, 3D Model

ABSTRACT:

The use of Indoor Mobile Laser Scanners (IMLS) for data collection in indoor environments has been increasing in the recent years. These systems, unlike Terrestrial Laser Scanners (TLS), collect data along a trajectory instead of at discrete scanner positions. In this research, we propose several methods to exploit the trajectories of IMLS systems for the interpretation of point clouds. By means of occlusion reasoning and use of trajectory as a set of scanner positions, we are capable of detecting openings in cluttered indoor environments. In order to provide information about both the partitioning of the space and the navigable space, we use the voxel concept for point clouds. Furthermore, to reconstruct walls, floor and ceiling we exploit the indoor topology and plane primitives. The results show that the trajectory is a valuable source of data for feature detection and understanding of indoor MLS point clouds.

1. INTRODUCTION

Indoor 3D models are required for navigation, building maintenance, disaster management and many other applications. However, manual creation of indoor 3D models is an expensive and cumbersome process for large buildings such as airports, shopping malls and hospitals. Indoor mobile laser scanning (IMLS) is a popular choice for data acquisition and for the generation of 3D models. Terrestrial laser scanners (TLSs) suffer from limitations such as the need for many scanning stations, manual alignment of point clouds and occlusion; in contrast, mobile laser scanners (MLS), because of their mobility, offer more coverage of indoor environment with less occlusion. IMLS systems, in addition to the point clouds, provide a continuous trajectory of device locations instead of few discrete station points in TLS. Current methods for indoor reconstruction and semantic labelling use mainly TLSs (Becker et al., 2015; Mura et al., 2014a; Oesau et al., 2014) or RGB-Depth data (Armeni et al., 2016; Khan et al., 2015). If MLS data is used as in (Xiao and Furukawa, 2014), the benefit of trajectory data is not exploited. In spite of the flexibility of MLS systems, the presence of occlusions caused by clutter (e.g. furniture) in the final point clouds is hindering indoor reconstruction. It is specifically challenging to distinguish automatically between occluded parts and genuine openings. We apply a method similar as Adan and Huber (2011) did in combination with the trajectory knowledge for detection of openings and labelling of gaps in the data. Another challenge of complex indoor environments is the presence of transparent and specular surfaces such as glass and mirrors that cause reflection and interference in point clouds. Such erroneous points - when they occur inside the permanent structure - cause the presence of "ghost walls" and surfaces which automatic methods are not capable to recognize. We offer a new method based on the trajectory and time stamps to detect and remove these reflections. For the detection of permanent structure such as walls, floor and ceiling, we use a method that exploits the topology of planar primitives. Current indoor reconstruction methods are often limited to Manhattan-World structure (Budroni and Boehm, 2010) or employ horizontal floor/ceiling and vertical wall assumption (Ochmann et al., 2016; Oesau et al., 2014; Xiao and Furukawa, 2014). Other methods generate 2.5D models (Oesau et al., 2014; Turner and Zakhor, 2014) or do not consider the detection of openings and addition of semantics (Mura et al., 2014a; Oesau et al., 2014; Xiao and Furukawa, 2014). Unlike the mentioned methods, our method is capable of dealing with highly cluttered environments, slanted walls and non-Manhattan-World structures. In order to partition the space into corridors, rooms and navigable space we introduce a new method that uses the concept of volumetric empty space by using voxel space. An analysis in voxel space allows space partitioning regardless the planar primitives and segmentation limitations. Additionally, we use a combination of voxels and trajectory to detect open and closed doors intersected by the trajectory. The result of this research is not a watertight reconstructed model but a representation of a coarse 3D model including permanent structure, openings and space partitions. The result is represented as a labelled point cloud, which will be used as input for further investigations. The main contribution of our work is in the exploitation of IMLS trajectories as a valuable source to interpret indoor scenes.

2. RELATED WORK

Indoor models have been investigated from two main perspectives: indoor model reconstruction and indoor model applications. While the former studies the methods for building 3D models (façade and indoor) from the data such as RGBD, LiDAR and imagery, the latter tries to improve indoor model standards such as BIM (Building Information Modelling) and CityGML (LoD4, LoD3). Our work is under the umbrella of the first group and we mainly focus on related work in the domain of indoor reconstruction. Most of the recent research on indoor modelling uses TLS data, synthetic data or Kinect data (Mura et al., 2014a; Ochmann et al., 2016; Oesau et al., 2014; Xiong et al.,

* Corresponding author

2013). There are few works that use IMLS data for indoor reconstruction (Sanchez and Zakhor, 2012; Xiao and Furukawa, 2014) but they are not using trajectory knowledge for better understanding of the indoor space. Sanchez and Zakhor (2012) propose an automatic method for planar 3D modelling of range scanner point clouds. Their data is not cluttered and the result does not have semantic attributes and openings. Budroni and Boehm (2010) apply a plane sweeping method to detect vertical and horizontal segments in a Manhattan-World case. The plane sweeping algorithm is a discrete method characterized by steps determined according to the points density.

Adan and Huber (2011), building on an approach by Okorn et al. (2010), try to reconstruct walls and openings under the presence of clutter and occlusion. Xiong et al. (2013) improve this approach by reconstructing a semantically rich 3D indoor model. The authors, by means of ray casting, propose a so-called occupancy map to label wall surfaces as empty (e.g. windows), occupied (e.g. wall surface) and occluded. However, their method is not capable of detecting closed doors or windows with high reflection. Mura et al. (2014b) developed a cell decomposition method of a cluttered point clouds from a non-Manhattan-World environment. Their approach can be considered as a combination of planar and volumetric based methods. In their approach, they do not detect openings or focus on semantic labelling. Furthermore, slanted walls and non-horizontal ceilings cannot be reconstructed.

Becker et al. (2015) use an L-System combined with a shape grammar to reconstruct interiors from point clouds. The authors define interior structure as two main subdivisions: rooms and hallways. Per each structure, they apply a specific grammar to reconstruct interior environment. Another example of using grammar is Ikehata et al. (2015). They encode indoor space first into a structure graph and then transform it to a structure grammar. Their work is a practical representation of inverse procedural modelling for indoor environments but limited to Manhattan-World structures.

In contrast to the mentioned methods that are mainly based on planar structures and are not capable of defining rooms and interior spaces as 3D volumes, volumetric based approaches represent a 3D semantic model of indoor environments. Volumetric based reconstruction generates a 3D model instead of 2.5D models that provides more information about rooms and their topology. Xiao and Furukawa (2014) propose iterative Constructive Solid Geometry (CSG) to generate a 2D model and with stacking 2D models on top of each other, a 3D model will be generated. Since the authors use rectangles and cuboids as primitives, their approach has a lack of efficiency for non-Manhattan-World cases. Ochmann et al. (2016) apply energy minimization for detection of wall with arbitrary orientation to reconstruct volumetric walls and rooms from point clouds. In their model, topological relations between spaces and rooms are reconstructed in a graph structure, which simplifies to update the model in the future.

In a recent work by Mura et al. (2016), the authors encode the indoor components in an adjacency graph and - by finding the path from ceiling to floor - try to detect and reconstruct the main components. Armeni et al. (2016) parse point clouds collected with a Matterport system into indoor components for large-scale indoor scenes. The authors use walls as space divider to semantically generate space subdivisions and use this information to detect other structure elements. However, their work is not an effort for reconstruction but more towards scene understanding.

One specific challenge when using mobile laser scanners in environments with many glass walls is the large number of reflections and points measured through transparent surfaces. There are a few works in the robotics domain that investigate this problem (Foster et al., 2013; Koch et al., 2017). The authors try to tackle scanner failures by understanding the relative geometry of real wall, glass wall and reflected wall. To our knowledge there is no research until now which investigates indoor environments using the IMLS trajectories.

3. PERMANENT STRUCTURE DETECTION

The input of our method can be mobile laser scanner point clouds acquired by trolley systems (e.g. NavVis M3, VIAMETRIS iMS3D), backpack (Google-Cartographer, Chen et al., 2010) or hand-held systems (e.g. ZEB1, ZEB-REVO). The examples in this paper are acquired by ZEB1 handheld laser scanner and NavVis M3 Trolley. Both systems use LiDAR sensors (e.g. Hokuyo UTM-30LX) with 3 to 5 cm noise in 10 to 30 m range according to the specifications. In this research, the assumption is that the point clouds are already registered in a common coordinate system and all points contain time stamps in both the trajectory and point clouds. Although some of the MLS data contain RGB information, in this research the color information is not exploited. The trajectories that we use in most algorithms consist of a set of ordered points of scanner positions with time stamps. To make a connection between each point in the point clouds and its scanner position in the trajectory we use time stamp in both datasets.

3.1 Removing Reflected Points

In the first step of our workflow, we filter noisy points, specifically points that are reflected from glass surfaces inside the main structure and cause "ghost walls" (s3 in Figure 1). We remove reflected points in a segment wise process. Our approach starts with the surface growing segmentation described in (Vosselman et al., 2004) followed by the removal of unsegmented points.

A point is labelled as reflected when the nearest trajectory point is more than 150 seconds (time lag parameter) earlier or later (Figure 2). If a specific percentage of points (e.g. 70%) in a segment is labelled as reflected points, the whole segment will be labeled as reflected segment and will be excluded from further processing (Figure 3).

3.2 Generating Surface Patches

Surface patches are composed of merging segments based on some rules. For subsequent steps, the level of details of segmentation result is too high. Generalization is the transition from one level of details to another. Thus, we use a generalization approach to merge segments, build on a method explained by Haala and Kada (2010). Two segments will be merged and generalized based on three criteria: (i) planes of two segments should be within a specific distance (generalization distance = d), (ii) planes normal should be almost parallel (angle threshold = θ), (iii) bounds of two segments should have overlap or be within a distance from each other alongside their planes (proximity distance = d'). Using the third criterion, we avoid generalizing segments whose planes are coplanar but their bounds are far from each other. Generation of surface patches is not limited to vertical segments but can be carried out on segments with arbitrary angle. Figure 4 shows one example of surface patch generation.

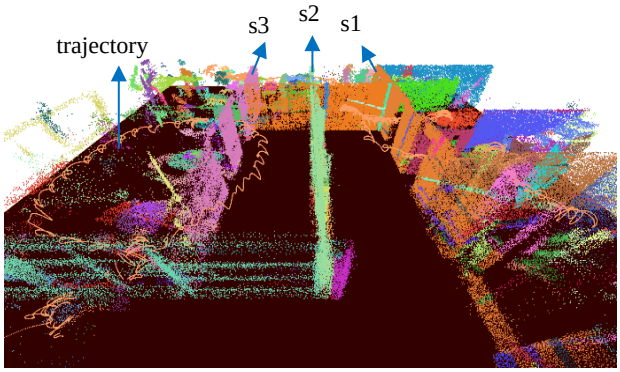


Figure 1. The trajectory (the zigzag orange line) and segments; s1, s2 and s3 are real surface, glass surface and reflected surface respectively. The data is acquired by ZEB1.

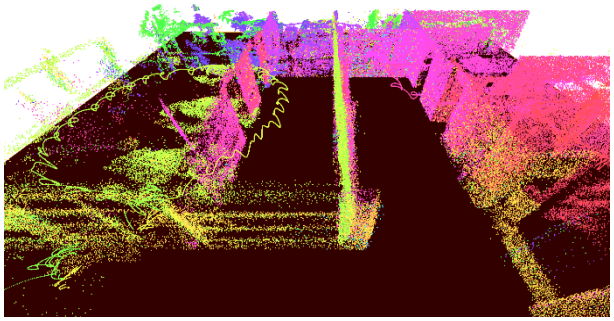


Figure 2. Trajectory and segments are colored by time. The different color of reflected surface and nearby trajectory is because of time difference.

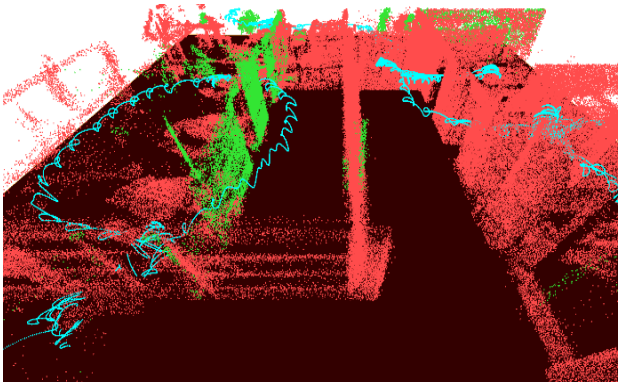
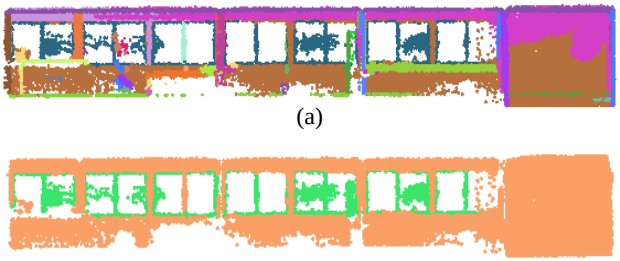


Figure 3. Reflected surfaces are detected and labeled to be removed (green). Trajectory is cyan and correct points are in coral.



(a)

(b)

Figure 4. All segments from figure (a) are generalized in two surface patches in figure (b). For this example, we used parameters: *d=40cm*, *θ=10 degree, d'=40cm.* Noise of this data is 5-8 cm.

3.3 Detecting Walls, Floors and Ceilings

In order to label each surface patch as wall, floor and ceiling we use the adjacency of extracted planar surfaces. Even in complex indoor environments, elements of the permanent structure often share common edges. For example, walls are connected to the floor and ceiling. However, this connection may not be detected because of clutter and gaps in the data. Especially at the connection between wall and floor, there is usually a lot of clutter (e.g. furniture). We define an adjacency graph as $G = (V, E, w)$ where V is representing each surface and E is a set of graph edges representing the connection and w is a set of attributes (e.g. intersection type). Encoding indoor planar surfaces and their connections in a graph has several advantages: 1. Processing graphs is computationally faster than processing segments and lines. 2. We can interpret graphs for extraction of primitives, and patterns. 3. It is possible to modify surfaces and their connections by adding/removing edges to the graph (e.g. removing furniture nodes).

3.3.1 Constructing Adjacency Graphs: To construct edges of the graph, we check if two surfaces intersect and if the intersection line is within a distance (e.g. 10 cm) of both surfaces. For each pair of adjacent surfaces one intersection line is generated (Figure 5, right). We classify surface patches to almost vertical and almost horizontal, using an angle threshold (e.g. 45 degree). Based on this classification of surfaces, one class represents the walls' connections (almost vertical) and the other wall and ceiling/floor connections (almost horizontal). In order to generate an adjacency graph, one edge E between each pair of intersecting surfaces is added to the graph G where vertices of E are the centers of gravity of each surface (Figure 6). We label (w) each edge of the graph based on following conditions:

$w = wallwall$ *iff* S_{vi} && S_{vj} *intersect*
$w = wallceiling$ *iff* S_{vi} && S_{hi} *intersect* && $S_{hi}(z) > S_{vi}(z)$
$w = wallfloor$ *iff* S_{vi} && S_{hi} *intersect* && $S_{hi}(z) < S_{vi}(z)$

Where S_{vi} is an *almost-vertical surface*,
S_{hi} is an *almost-horizontal surface*, and
$S(z)$ is the z value of surface's *center of gravity*.

3.3.2 Labeling Surfaces: After generating the adjacency graph, we are able to analyze each node ($v \epsilon V$) in the graph based on connections ($e \epsilon E$). Each node, which is representing a surface (S), takes one of the labels wall, floor, ceiling or no-label based on the following conditions.

(1) iff $No_{wc} \geq 1$ && $S \epsilon Svi$ then S is ***wall***
(2) iff $No_{wc} > 2$ && $No_{ww} == 0$ then S is ***ceiling***
(3) iff $No_{wf} > 2$ && $No_{ww} == 0$ then S is ***floor***

Where No_{wc} is the number of ***wallceiling*** labels,
No_{ww} is the number of ***wallwall*** labels, and
No_{wf} is the number of ***wallfloor*** labels in each node.

In labeling process, we do not include *wallfloor* connections as a valid condition for labeling a wall, because many features (e.g. furniture) are connected to the floor. The labeling process may assign a ceiling or floor label to any almost horizontal surface connected to other almost vertical surfaces. To modify these incorrect labels we need to estimate floor and ceiling height after the process and prune the results.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-IV-2-W4-355-2017 | © Authors 2017. CC BY 4.0 License.

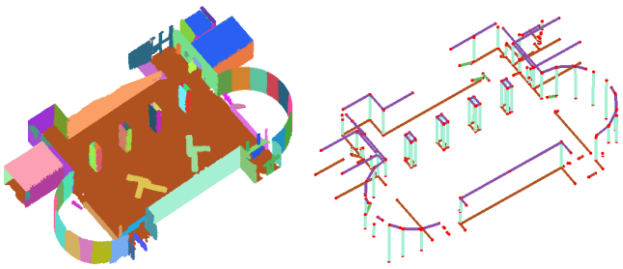


Figure 5. Left: surface patches from generalization process. Right: intersection lines between pair surfaces. For simplicity of figures, we chose an area with minimal clutter. The data is acquired by a trolley system.

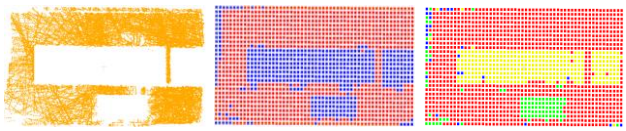


Figure 7. Left: a wall surface containing holes, orange points are point clouds. Middle: the same wall surface in a voxel space, red voxels represent *occupied* areas and blue voxels are *empty*. Right: voxels after *occlusion* test, yellow voxels are *openings* and green are *occlusions*.

This implies that nodes with floor/ceiling label will be removed from the graph when the distance to the correct floor/ceiling height surpasses a threshold. Consequently, we remove also connected nodes with wall label. The results contain surfaces with wall, floor, ceiling and unknown labels.

3.4 Detecting Openings Using MLS Trajectory and Occlusion Tests

We use the detected wall surfaces in combination with the trajectory to detect openings (windows and doors). If we consider each opening as a hole in the wall surface, we need to discriminate between genuine openings and gaps because of occlusion. Therefore, we are not focusing on extracting the exact border of openings, but on labelling the holes in the point clouds. Adan and Huber (2011) use ray-tracing and occlusion labelling to detect openings. In their example they use ray-tracing for a single scanner position. However, in our case this is not applicable because each surface can be seen from many points on the trajectory.

To implement occlusion test, we need the information from which scanner position in the trajectory, a point in the point cloud was observed. As explained in section 3.1, this information is available using time stamp in both data sets. Additionally, we reconstruct a 3D voxel space for each wall surface. Hence, in the position of holes, there are voxels with *empty* label and in the position of points there are voxels with *occupied* label. The occlusion test will label empty voxels further. One advantage of voxel space is speeding up the labelling process. Therefore, the occlusion test process needs three inputs: the whole point cloud, a set of voxels per surface and the trajectory. Per point in the point clouds, we reconstruct a ray to its corresponding scanner position for each time stamp (every 0.01 seconds depending on the data) and then check for occlusion by checking the intersection of the ray with other surface planes. If there is an intersection, the intersected voxel can take three labels: 1. *occlusion* if the intersection point is in front of the surface, 2. *opening* if the intersection point is behind the surface, 3. *occupied* if the intersection point is close to the surface and remains unchanged if there is no intersection (Figure 7). One drawback of the occlusion test is that during wall detection some surfaces are incorrectly labelled as walls, which consequently result in labelling fake openings (see black solid ovals in Figure 8b).

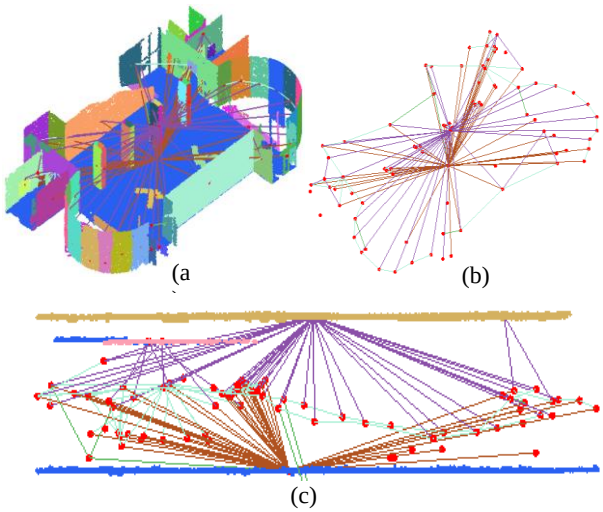


Figure 6. (a) Shows surfaces with connected edges. (b) Shows edges and nodes of the graph. Each node is representing a surface. (c) Shows the floor and ceiling surfaces and edges of the graph. Three different colors of edges representing three types of connection for *wallwall* (cyan), *wallceiling* (purple) and *wallfloor* (brown).

3.5 Modified Point Clouds and Second Iteration

One of the challenges of indoor reconstruction is removing points and surfaces outside the building footprint (Figure 8a). *Modified point clouds* are one of the side products of the occlusion test in which points behind transparent surfaces especially outside the building façade are removed. Then, by running wall detection method (section 3.3) for second time on *modified point clouds*, we remove excessive walls outside the building footprint (see black dot ovals in Figure 8b); we refer this process as *second iteration* for wall detection (Figure 8c). However, modifying point clouds in this way causes removing some of the points that belong to the main structure, but points that are measured through a transparent surface are expected to be noisier than point clouds and we prefer to eliminate them from the point cloud. The resulting dataset is essentially useful in buildings with glass walls.

4. VOLUMETRIC SPACE PARTITIONING AND DOOR DETECTION IN VOXEL SPACE

Partitioning the space to meaningful areas such as rooms and corridors is referred to room segmentation, space subdivision or space partitioning (Krūminaitė and Zlatanova, 2014; Mura et al., 2016; Turner and Zakhor, 2014). By using mobile laser scanners, there is no information for the partitioning to individual rooms because we are not aware which points are captured in the same room. Therefore, we need a different approach than others who use TLS scans information for a coarse room segmentation (Ochmann et al., 2016).

For detecting doors, we introduce a new approach that exploits the trajectory knowledge in combination with a voxel space. This method is capable of detecting most of the doors (either open or closed) intersected by the trajectory.

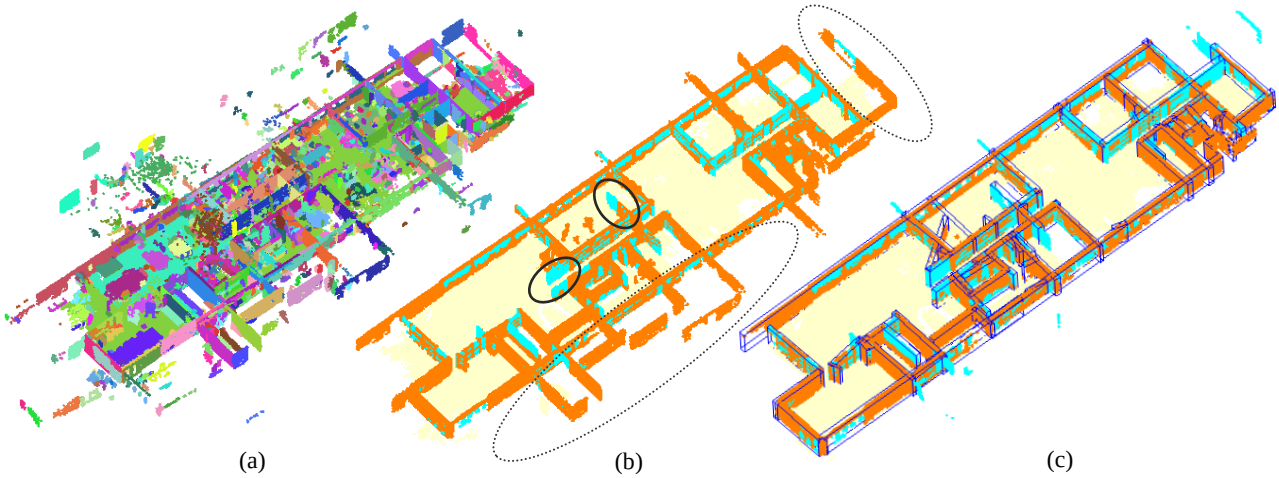


Figure 8. The images show the input and output of section three. The algorithm takes laser point clouds as input and exports labeled laser points including walls, openings, floor, ceiling and clutter. (a) Cluttered 3D point clouds of a building story acquired by Zeb1. (b) Detected walls (orange) and openings (cyan) in the first iteration. Black dot ovals show incorrect walls outside the building footprint. Black solid ovals show incorrect detected openings (fake openings). (c) Detected walls and openings in second iteration after automatically removing points outside the building footprint. Blue boxes show the extent and thickness of the walls.

4.1 Space Partitioning

Space partitioning is independent from results of previous steps and is not based on segmentation into planes. The process starts with reconstructing 3D voxels from point clouds. Each voxel takes an *occupied* label if there are laser points in it and *empty* otherwise. The algorithm tries to collect empty voxels within a margin of occupied voxels. Therefore, per each empty voxel, we search all neighboring voxels by a search window (e.g. 30 cm) and we store this empty voxel if a percentage (e.g. 70%) of its neighbors is empty voxels. If we apply this process for the neighborhood of each empty voxel in three dimensions, the result will be a cluster with all empty voxels within a distance of occupied voxels (clutter), see Figure 9a. Since there are openings and gaps between permanent structures, some of the partitions remain connected. To disconnect some lose connections that probably are connected through small windows or small gaps, a morphological erosion in three dimensions on the result will be applied. Erosion will also remove very small empty spaces for instance under a table.
Eventually, to have each individual empty space, we apply a connected component segmentation and generate various partitions (Figure 9a). As a result, each cluster of empty space could represent a room, corridor or empty space between furniture. One challenge in space partitioning is the presence of gaps and windows in the point clouds that causes the connection of partitions. In section 4.4, a solution for this problem is offered.

4.2 Navigable and non-Navigable Space

From the space partitioning results we extract those voxels which are just above the floor (e.g. 10 cm above the floor), and store them as navigable space (Figure 9d). This navigable area could be generated in various heights for the purpose of navigation for flying objects (e.g. drones). By using the original point clouds, it is likely that undesired space partitions and navigable space will be generated because of noisy points outside the building façade. These areas can be cropped by using the building footprint (as an external source) or using the method described in section 4.4.

4.3 Door Detection Using Trajectory

Doors are an important component for indoor reconstruction, especially for indoor navigation and room detection. During opening detection, we detected some of the doors as openings, in this step we specifically focus on detecting doors by using the trajectory. In the voxel space, each voxel can represent the *center of a door* if it satisfies three conditions:
(i) Above each door center there should be at least several occupied voxels representing top of the door (top-door).
(ii) Nearby (e.g. 15 cm) each door center there should be one or more trajectory points.
(iii) In a neighborhood (e.g. 50 cm) of a door center there should be void (empty voxels) for open doors and not void (occupied voxels) for closed doors.
If these three conditions are fulfilled, the voxel center is labelled as a door center candidate. In this article, we call the voxels and corresponding points above the door edge shortly top-door.
There may be several door center candidates per door that meet the three conditions. After finding door centers, points on top-door can be extracted, which gives us the position and orientation of the door and we can use this information to further look for door borders. In our examples, we just extract top-doors. The process needs the width and the height of the door, the search radius for the trajectory, the search radius for the void-hood as input, as well as a void-hood percentage (e.g. 70% empty voxels in the neighborhood). To speed up the process we only check voxels above a specific height, where a door center could be located. To search nearby trajectory points, the search area depends on the scanner device. For example, for a handheld laser scanner the search area should be different from a trolley laser scanner.

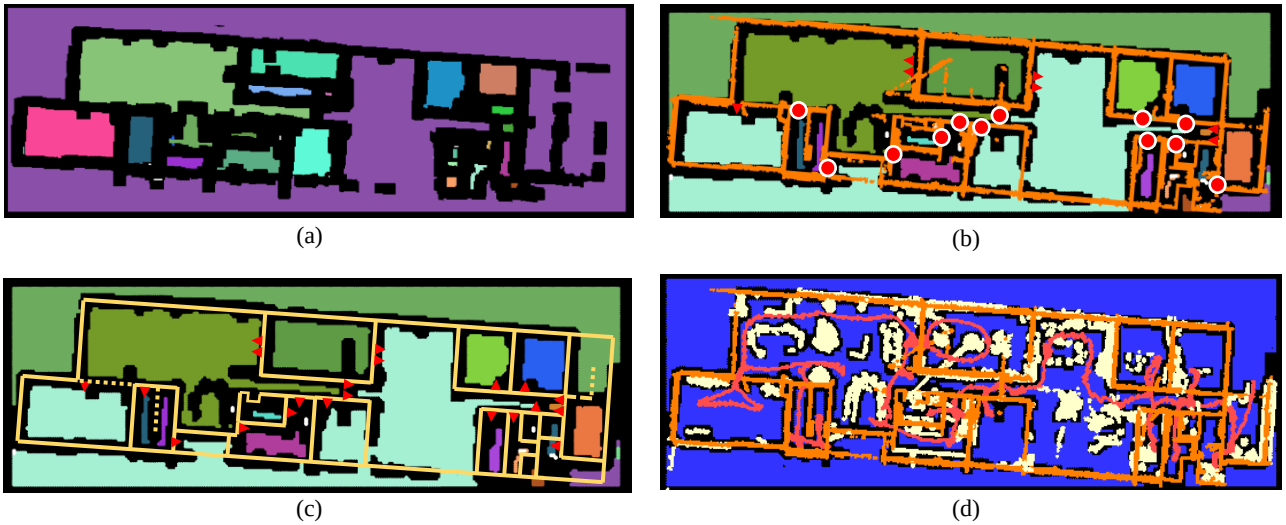


Figure 9. (a) Top view of space partitioning without using openings and wall information. Colored areas are empty spaces and black areas show the position of walls and clutter. (b) Space partitioning results after modification with openings and wall information. The overlaid orange layer show detected walls. Red dots show detected doors and red triangles show undetected doors. (c) Space partitioning and original floor plan. Light green area in the bottom of figure shows not correctly divided partitions. Small red triangles show the location of correct doors. (d) Navigable space (blue), clutter (yellow), and trajectory (coral).

4.4 Combining Permanent Structure Results and Space Partitioning

During the space partitioning process, some of the partitions could be connected to each other because of gaps and openings in the point clouds. To correct this problem and improve space partitioning, we use the detected openings and gaps resulting from the step described in section 3.4 and modify the partitions. Consequently, partitions that are incorrectly connected will be split (Figure 9b). However, this modification could result in over-segmentation when incorrect openings are involved. In the case of large gaps and openings that are not detected by the occlusion tests, the space partitions remain connected (light green area in the bottom of Figure 9c and Figure 10). It is possible to separate connected partitions alongside the wall plane by using the wall information and post-processing.

5. EVALUATION

We tested our algorithms on a dataset of a three stories building acquired with the ZEB1 handheld system. The selected story contains 16 million points, which is reduced to 2.1 million by subsampling. The noise is between 5 to 8 cm (technical details are in (Sirmacek et al., 2016)). The dataset in Figures 5 and 6 were acquired by a trolley system. The point clouds shown in other images were captured with the ZEB1 system. Figure 8 shows the input and output of our approach in section three, where we detect walls and openings.

5.1 Parameter Selection

The first set of algorithms explained in section 3 is highly depend on the segmentation quality. Our algorithms for wall detection and opening detection work better with large segments, which contain several planes composing a wall. Generating surface patches reduces the number of segments from approximately 1500 segments to 560 patches. For generating surface patches, we use the thresholds in Table 1. Usually, the thickness of a wall (including windows frames) varies from 40 to 80 cm, so the first threshold for planes distance should be in this range. The co-planarity angle also should not exceed 20 degrees, because then surface patches will be skewed. The third parameter is the distance between segments alongside their plane and it should not be larger than a narrow corridor size. Otherwise, co-planar segments from two sides of the corridor could be merged and this would be problematic for occlusion tests.

For the wall detection and the reconstruction of the adjacency graph, the intersection threshold of 0.10 m is appropriate for most kind of datasets. For the classification of planes to almost vertical and almost horizontal, we expect a fixed angle of 45 degree should result in precise outcome, but this has not been tested on various datasets. Additionally, during the wall detection we ignore small segments with less than 300 points (minimum segment size parameter). Other parameters (floor and ceiling height estimation) in wall detection are optional.

In the second set of algorithms in voxel space (section 4), the selection of the voxel size is crucial and it should not be smaller than the point spacing. We tried our algorithm with a 5 cm, 10 cm and 20 cm voxel size. With a 20 cm voxel size, the computation time is less but the results are sparse and less accurate especially in the border of openings. Therefore, 5 and 10 cm voxel size are better choices.

In the space partitioning and door detection, a larger search window size can significantly increase the computation time. The search window in space partitioning algorithms also defines the distance to the clutter.

The selection of parameters for the door detection algorithm needs more investigation, because the void-hood percentage and search window size should be matched to find enough candidate points in the door centers. For example, doors that are half opened in the point clouds are difficult to process as an open door or closed door. One reason is, near the door center at the same time there are both empty neighbors and occupied neighbors that belong to the door position during scanning.

5.2 Robustness

The quality of the results for section 3 (reflection removal, wall detection, opening detection) depends on the segmentation and generation of surface patches. An ideal surface patch contains segments that belong to the same wall and has a rectangular shape.

Despite being capable of dealing with non-Manhattan-World

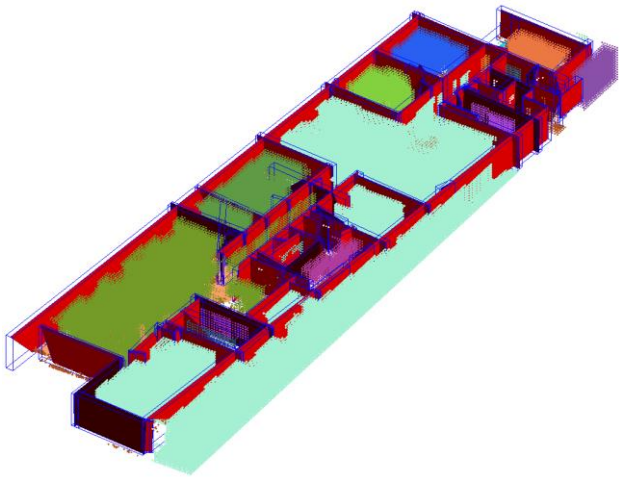


Figure 10. A 3D model of walls and space partitions. This model can be further processed to reconstruct a watertight model.

structures, our algorithm may fail in cases of loose connections or no connection to the ceiling.

The opening detection algorithm depends on the correctness of detected surfaces. More rectangular surfaces lead to better result, because of usage of bounding box for each surface to detect openings. In the case of walls which are not extended to the ceiling (e.g. kitchens island, staircases) excess openings will be generated (Figure 8b). Detecting excessive openings (false positive predictions) is especially problematic when we remove points behind them, because removed points might belong to the main structure. However, in our experiment the amount of these false positive openings is low and does not affect the overall accuracy significantly. Using the result of *modified points* from the opening detection step and running the wall detection algorithm for second iteration (section 3.5) considerably improves the wall detection results and removal of the incorrect walls outside the building footprint (Figure 8b).

The space partitioning algorithm in principle is independent from segmentation, the result is more promising after correction with wall and opening detection results (Figure 9b). Navigable space in 2D and 3D is a valuable information and is robust against parameter selection. The only affecting issues for the navigable space are points outside of the building footprint or connections through undetected gaps or openings. The door detection algorithm is sensitive to parameter changes and can be error prone concerning the position of the door during the scanning.

5.3 Accuracy

Since we present the result as a set of labelled point clouds, the correctness and completeness is measured point-wise, not object-wise (Tables 2 and 3). We calculated the accuracy of wall, floor and ceiling detection separately from opening detection. During the opening detection process, points (voxel centers) will be generated in the position of gaps and holes, but we just measure the correctness of the result of the generated points. The result is highlighted in Table 2, right. The lower precision of occluded area (gaps) is a consequence of detecting many false positives outside the building footprint. In Table 2, left and Table 3 the correctness (precision) of wall, floor, and ceiling in both iterations is promising, however, the completeness (recall) of classes in second iteration dropped significantly. This is a result of removal of points behind the surfaces during the occlusion test, which consequently in second iteration deteriorate the completeness of the results.

Regarding door detection algorithm, we detected twelve correct

Algorithm	Parameters	Value
SurfaceGrowing Segmentation	distance to surface seed search radius	0.10 m 1.0 m
Reflection Removal	time difference # of reflected points in a segment	150 s 70%
Surface Patch Generation	planes distance segments distance planes angle	0.60 m 0.40 m 10 degree
Wall/Floor/Ceiling Detection	intersection threshold surface angle threshold floor height estimation (optional) ceiling height estimation (optional) dist to floor, ceiling (optional)	0.10 m 20 degree - - 0.50 m
PruneWall Detection	dist to floor, ceiling	0.50 m
Occlusion Test (Opening detection)	voxel size closenees dist to surface	0.10 m 0.60 m
Space Partitioning	voxel size search windows size	0.10 m 5*voxel_size
Door Detection	voxel size door size (width, height) search windows size percentage of void_hood points trajectory search radius	0.10 9*21*voxel_size 5*voxel_size 70% 0.15 m

Table 1. Parameter settings per algorithm

Class	Precision	Recall	F1-Score
Wall	0.95	0.53	0.68
Floor	0.97	0.67	0.79
Ceiling	0.98	0.51	0.68

Class	Precision
Openings	0.73
Occluded	0.57
Occupied	0.89

Table 2. Accuracy results of 2nd iteration (section 3.5), left: permanent structure detection results, right: opening detection results. For opening detection because we do not have total number of points in openings and gaps (there is hole in the data), we just show the correctness (precision) of the detected results and not completeness (recall).

Class	Precision	Recall	F1-Score
Wall	0.88	0.95	0.91
Floor	0.93	0.98	0.95
Ceiling	0.93	0.98	0.95

Table 3. Accuracy result of 1st iteration (section 3.3).

hits out of seventeen doors (Figure 9b). Among detected doors, four doors are closed doors, which are cumbersome to detect without using the trajectory. Undetected doors are mainly because of sparse number of points on top of the door center or because the criteria for void hood search around the door center are not satisfied.

The quality of space partitioning result is highlighted and compared with original floor plan in Figure 9c. Except the hall in the middle of area that is connected to the outside through large windows (light green area), other partitions are mostly detected and divided correctly. It is possible to improve this result with further processing with wall positions.

5.4 Limitations

Although our method for wall detection is not limited to vertical walls or horizontal floor/ceiling, it needs to be improved in complex surfaces with several connection angles. Another limitation is the composition of structure near the ceiling. If walls are not connected to the ceiling, or heavily occluded near the

ceiling, the result is not promising. In spaces that there are structures not extended to the ceiling such as kitchen islands, staircases, bookshelves near the wall, the wall detection algorithm fails and opening detection can generate fake windows in such cases. Built-in bookshelves and cabinets will be detected as part of the wall structure, if during the patch generalization they are generalized as part of the surrounding wall.
Concerning reflected points from glass surfaces, currently we remove them, and the better solution is try to reconstruct them in the correct position by exploiting scanning geometry.
Among the algorithms, the occlusion test, generation of surface patches and space partitioning significantly take more time to process than the other algorithms.
In this work, we do not provide detailed timing process and computation environment because it is not the focus of this research. Algorithms optimization is part of the future work.

6. CONCLUSIONS AND FUTURE WORK

We presented several algorithms for the interpretation of interior space using MLS point clouds in combination with the trajectory of the acquisition system. We use trajectories for the detection of openings, doors, and reflected surfaces. Furthermore, our permanent structure algorithm is not limited to Manhattan-World and 2.5D assumptions. By applying the voxel space to 3D point clouds, we extract and partition empty spaces by using opening and wall detection results. Applying our methods on a complete building with slanted walls, extracting stairs, detection of openings borders and generate a watertight model are topics of future work.

7. ACKNOWLEDGEMENTS

This work is part of the TTW Maps4Society project Smart 3D indoor models to support crisis management in large public buildings (13742) which is (partly) financed by the Netherlands Organization for Scientific Research (NWO). We thank the Fire Brigade Rotterdam Rijnmond for making one of their buildings available for the test and data collection.

8. REFERENCES

Adan, A., 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* (3DIMPVT). pp. 275–281.

Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S., 2016. 3D Semantic Parsing of Large-Scale Indoor Spaces. Presented at the Proceedings of the *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1534–1543.

Becker, S., Peter, M., Fritsch, D., 2015. Grammar-Supported 3d Indoor Reconstruction From Point Clouds For "As-Built" Bim. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* 1, 17–24.

Budroni, A., Boehm, J., 2010. Automated 3D Reconstruction of Interiors from Point Clouds. *Int. J. Archit. Comput.* 8, 55–73.

Chen, G., Kua, J., Shum, S., Naikal, N., Carlberg, M., Zakhor, A., 2010. Indoor localization algorithms for a human-operated backpack system, in: *3D Data Processing, Visualization, and Transmission. Citeseer*.

Foster, P., Sun, Z., Park, J.J., Kuipers, B., 2013. VisAGGE: Visible angle grid for glass environments, in: *2013 IEEE International Conference on Robotics and Automation*, pp. 2213–2220.

Haala, N., Kada, M., 2010. An update on automatic 3D building reconstruction. *ISPRS J. Photogramm. Remote Sens., ISPRS* Centenary Celebration Issue 65, 570–580.

Ikehata, S., Yang, H., Furukawa, Y., 2015. Structured indoor modeling, in: Proceedings of the *IEEE International Conference on Computer Vision.* pp. 1323–1331.

Khan, S.H., Bennamoun, M., Sohel, F., Togneri, R., Naseem, I., 2015. Integrating Geometrical Context for Semantic Labeling of Indoor Scenes using RGBD Images. *Int. J. Comput. Vis.* 1–20.

Koch, R., May, S., Murmann, P., Nüchter, A., 2017. Identification of transparent and specular reflective material in laser scans to discriminate affected measurements for faultless robotic SLAM. *Robot. Auton. Syst.* 87, 296–312.

Krūminaitė, M., Zlatanova, S., 2014. Indoor Space Subdivision for Indoor Navigation, in: *Proceedings of the Sixth ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, ISA '14. ACM, New York, NY, USA, pp. 25–31.

Mura, C., Jaspe Villanueva, A., Mattausch, O., Gobbetti, E., Pajarola, R., 2014a. Reconstructing complex indoor environments with arbitrary wall orientations. *Proc Eurographics Posters Eurographics Assoc.*

Mura, C., Mattausch, O., Jaspe Villanueva, A., Gobbetti, E., Pajarola, R., 2014b. Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Comput. Graph.* 44, 20–32.

Mura, C., Mattausch, O., Pajarola, R., 2016. Piecewise-planar Reconstruction of Multi-room Interiors with Arbitrary Wall Arrangements. *Comput. Graph.* Forum 35, 179–188.

Ochmann, S., Vock, R., Wessel, R., Klein, R., 2016. Automatic reconstruction of parametric building models from indoor point clouds. *Comput. Graph.* Special Issue on CAD/Graphics 2015 54, 94–103.

Oesau, S., Lafarge, F., Alliez, P., 2014. Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. *ISPRS J. Photogramm. Remote Sens.* 90, 68–82.

Okorn, B., Xiong, X., Akinci, B., Huber, D., 2010. Toward automated modeling of floor plans, in: *Proceedings of the Symposium on 3D Data Processing, Visualization and Transmission.*

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

Sirmacek, B., Shen, Y., Lindenbergh, R., Zlatanova, S., Diakite, A., 2016. Comparison of ZEB1 and Leica C10 Indoor Laser Scanning Point Clouds. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* 143–149.

Turner, E., Zakhor, A., 2014. Floor plan generation and room labeling of indoor environments from laser range data, in: *Proceedings of International Conference on Computer Graphics Theory and Applications.*

Vosselman, G., Gorte, B.G., Sithole, G., Rabbani, T., 2004. Recognising structure in laser scanner point clouds. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* 46, 33–38.

Xiao, J., Furukawa, Y., 2014. Reconstructing the world's museums. *Int. J. Comput. Vis.* 110, 243–258.

Xiong, X., Adan, A., Akinci, B., Huber, D., 2013. Automatic creation of semantically rich 3D building models from laser scanner data. *Autom. Constr.* 31, 325–337.