AUTOMATIC GENERATION OF INDOOR NAVIGABLE SPACE USING A POINT CLOUD AND ITS SCANNER TRAJECTORY

B. R. Staats[a], A. A. Diakité[b], R. L. Voûte[c,d], S. Zlatanova[b,*]

[a] Master of Science Geomatics, Faculty of Architecture and the Built Environment, TU Delft, The Netherlands – b.r.staats@student.tudelft.nl
[b] 3D Geo-Informatie, Faculty of Architecture and the Built Environment, TU Delft, The Netherlands - (a.a.diakite, s.zlatanova)@tudelft.nl
[c] Department of Urbanism, Faculty of Architecture and the Built Environment, TU Delft, The Netherlands - r.voute@tudelft.nl
[d] CGI Nederland BV, The Netherlands – robert.voute@cgi.com

KEY WORDS: Navigation space, MLS, Floorplan, Indoor, Trajectory, Voxel, Dynamic objects, 3D laser scanning

ABSTRACT:

Automatic generation of indoor navigable models is mostly based on 2D floor plans. However, in many cases the floor plans are out of date. Buildings are not always built according to their blue prints, interiors might change after a few years because of modified walls and doors, and furniture may be repositioned to the user's preferences. Therefore, new approaches for the quick recording of indoor environments should be investigated. This paper concentrates on laser scanning with a Mobile Laser Scanner (MLS) device. The MLS device stores a point cloud and its trajectory. If the MLS device is operated by a human, the trajectory contains information which can be used to distinguish different surfaces. In this paper a method is presented for the identification of walkable surfaces based on the analysis of the point cloud and the trajectory of the MLS scanner. This method consists of several steps. First, the point cloud is voxelized. Second, the trajectory is analysing and projecting to acquire seed voxels. Third, these seed voxels are generated into floor regions by the use of a region growing process. By identifying dynamic objects, doors and furniture, these floor regions can be modified so that each region represents a specific navigable space inside a building as a free navigable voxel space. By combining the point cloud and its corresponding trajectory, the walkable space can be identified for any type of building even if the interior is scanned during business hours.

1. INTRODUCTION

Navigation from a room inside a building to a room inside another building across the street consists of three parts: a first indoor part in the building where you start your journey, an outdoor part and a second indoor part in the destination building (Thill et al., 2011). In the outdoor environment, a navigation aid is well implemented and used in all kinds of applications but a navigation aid inside a building is still lacking. This aid is not necessary in small, simple buildings, but it is required in more complex ones like hospitals, airports, conference venues and large shopping malls. To aid visitors inside these buildings, an indoor navigation system is required. These systems exist of several elements like an indoor positioning system, an indoor navigable map, specific destinations (points of interest) and an appropriate guidance to follow the path (Brown et al., 2013, Boguslawski et al., 2016). An important aspect is the creation of an indoor map that can be displayed and used to plan possible routes throughout the building. Since the indoor environment is far more complex than the outdoor environment, automating the generation of indoor maps is challenging and time consuming (Zlatanova et al., 2014, Diakité and Zlatanova, 2016).

Most research on automatic generation of indoor maps is focused at the available 2D floor plans and only a few of them use the complex 3D representations (Zlatanova et al., 2014). Using 2D floor plans has its limitations. First of all, the 2D maps are a simplification of the complex 3D environment which can lead to difficulties in representation. Secondly, the connectivity between different floor plans can be difficult (Zlatanova et al., 2014). Thirdly, the maps that are available do not always contain furniture and most existing methods only focus on the reconstruction of the indoor space as an empty hull with less attention to obstacle detection (Díaz Vilariño et al., 2016). At last, as discussed by Turner et al. (2015), most floor plans are out of date. Buildings are not always built according to their blue prints, interiors might change after a few years by modification of walls and doors and furniture may be repositioned to the users preferences. Therefore, new approaches for the efficient 3D recording of indoor environments should be investigated. This paper concentrates on the automatic generation of indoor navigable space for pedestrians based on laser scanning with a Mobile Laser Scanner (MLS) device. These devices scan the environment continuously along a trajectory which makes them more time efficient than static terrestrial laser scanners (Holenstein et al., 2011).

The output of a MLS device is a point cloud. To aid pedestrians during their indoor navigation, features needed for path computation such as floors, stairs, walls and furniture objects, need to be extracted. Many other approaches are built on constraints, like a Manhattan World or a flat/planar/horizontal surface constraint (Fichtner, 2016, Macher et al., 2016, Khoshelham and Díaz-Vilariño, 2014, Anagnostopoulos et al., 2016, Budroni and Boehm, 2010). These constraints and limitations are not problematic for a regular office building but they are in more complex environments, so a focus on a method with less or without constraints is needed.

If a building is scanned during business hours, or a building cannot be closed like hospitals or airports, the captured point cloud contains dynamic elements like pedestrians or small vehicles. These dynamic elements do not represent any type of building element like furniture and thus need to be identified and removed.

Beside the point cloud, the MLS also stores the trajectory that the

*Corresponding author

MLS device took during the scanning procedure. If the MLS device is operated by a human, the trajectory contains information, which can be used to distinguish between different surfaces. The points, which are directly below the trajectory, indicate pedestrian walkable areas (Yan et al., 2016, Li, 2014). The height difference between neighbouring trajectory points can be used to detect stairs, slopes and flat surfaces. The trajectory also provides connection information and represents the complexity of the building when data is captured according to certain rules.

In this paper a method is presented for the identification of walkable surfaces based on analysis of the point cloud in combination with the trajectory of the MLS scanner. The method consists of two stages: a surface reconstruction phase and a cleaning phase. By combining the point cloud and the trajectory, the walkable spaces can be identified for any type of room without any constraints because the complexity of the building is already present in the trajectory of the MLS.

2. RELATED WORK

An indoor navigable space can be defined as the free surfaces that are used to navigate inside a building without bumping into any obstacles. To identify the indoor navigable space, building surfaces need to be constructed. The reconstruction of indoor environments is researched a lot and different approaches exist. In this section, a few examples of different techniques in the field of 3D reconstruction are discussed. How to filter humans and other dynamic obstacles in point clouds has also been investigated. A short overview on the research in the detection of dynamic objects in point clouds is also presented.

A common approach for surface reconstruction is the calculation of the normal of each point in the point cloud. By grouping the points with the same normals, smooth surfaces can be defined (Rabbani et al., 2006). Díaz Vilariño et al. (2016) proposed a method which segments a point cloud into regions based on these normals. A plane is fit to each region and these planar regions are then intersected and classified as walls, ceilings and floors based on the tilt and the position regarding to the centre of the room. The resulting unclassified points are marked as furniture. After classification of doors, obstacle aware navigation can be implemented by combining the planar regions and the furniture points.

Budroni and Boehm (2010) developed a method to reconstruct buildings from point clouds by using plane sweeps. First, ceilings and floors are found by horizontal plane sweeps. If the amount of points inside a specific plane reaches a threshold, floors or ceilings are detected. The next step consists of a few vertical plane sweeps at random locations. For each location histograms are computed by using the information of the vertical pane sweeps. These histograms are then combined which results in the main direction of walls in the entire building. The last step is cell decomposition of the entire space and extrusion of walls into the final building model. This method focusses on perpendicular, parallel walls and flat surfaces (Budroni and Boehm, 2010).

A similar approach is introduced by Fichtner (2016). This method uses an octree structure proposed by Broersen et al. (2016) to structure the input point cloud into voxels. The use of a voxel model has several advantages. Firstly, the number of voxels is much smaller than the number of points which makes storage, retrieval and processing far more efficient. Secondly, the voxel grid possesses a spatial structure, which makes searching for neighbour voxels quick and easy (Vo et al., 2015, Suzuki et al., 2010). Since the voxelization is a generalization of the point cloud, the representativeness of the voxelized point cloud heavily depends on the voxel size. The method of Fichtner (2016) first replaces and rotates the entire point cloud so that the walls and floors in the model are aligned with the x, y and z axis. After the voxelization, the method continues by deriving histograms of the occupied leaves (voxels) in the z direction. If a peak is detected, it indicates a possible floor. The next step consists of wall identification by applying histograms in the x and y direction. Furthermore, stairs are detected using a filter based approach. This method only works on Manhattan worlds and small floors cannot be defined in larger point clouds.

The proposed method by Vo et al. (2015) combines the voxel approach of Fichtner (2016) with the smoothness approach of Rabbani et al. (2016) . In approach, the point cloud is structured in an octree/voxel model. For the points in each voxel, the normals are calculated. These voxels are then grouped according to the normal classification. This method improves the speed and accuracy of the classification of the final result.

Turner et al. (2015) uses a ray tracing method through a voxel space with an ambulatory scanning platform. If a laser beam crosses through a voxel it is marked as an interior voxel. The final voxel is marked as a boundary voxel. By partitioning the voxel faces in planar regions and representing these regions as a quad tree, a triangulation is performed. In this way, a building representation which contains furniture is created. A second model without furniture is created based on a 2D floor plan. These floor plans are found by the creation of a histogram of the input point cloud. As discussed earlier, the use of a histogram method limits the detection of small floor areas across the building.

The 3D reconstruction of the indoor free space is a complex problem because of the variety of shapes of spaces and objects, a high level of dynamic objects like walking humans or small moving vehicles, occlusions and cluttered surfaces. Therefore, most of the 3D reconstruction approaches introduce assumptions like Manhattan worlds or planarity of surfaces and are avoiding the reconstruction of furniture. Our approach concentrates on the specific objects used for reconstruction which are of interest for the navigation of humans: stairs, flat, sloped surfaces and furniture objects.

2.1 Detecting dynamic objects

Commonly, public buildings like hospitals or airports cannot be closed during the data capture. Therefore, the generated point cloud contains dynamic objects like people or small moving vehicles. These elements have effect on the final output and should be filtered before further processing. The detection of moving objects is largely investigated in robotics and mostly done using specific technologies like range scans, stereo cameras, monocular cameras and recently also with consumer RGB-D cameras.

However, identifying dynamic objects based on point clouds is not yet extensively explored (Litomisky and Bhanu, 2013). The detection of dynamic elements is even more difficult because they are only present on a specific position at a given time, which makes the dynamic elements long-drawn shadows in the final registered point cloud (Józsa, 2012).

The approach of Litomisky and Bhanu (2013) to detect dynamic objects is based on splitting the point cloud into two data frames.

These data frames should have a significant overlap and a time difference. Large planes inside these data frames are subtracted, which is followed by the segmentation of the remaining points. The individually segmented clusters are then compared to the clusters of the other data frame and the dynamic elements can be removed.

A different approach is presented by Holenstein et al. (2011). They also work with a ray tracing method to divide the entire space in free and occupied voxels, just like Turner et al. (2015). By combining rays of multiple measurements, points from dynamic objects are identified and removed. Ray tracing methods are costly because of their computation time. As described by Holenstein et al. (2011), a point cloud of 5 million points with a voxel size of 5 centimetres took 8 hours and 45 minutes to process. Ray tracing methods are therefore not advised to use.
The detection of dynamic elements in this paper is based on the notion of Józsa (2012). A dynamic object is only present on a specific position at a given time. The SLAM algorithm that is used to form the final output point cloud of the MLS devices gives a more reliable result when the data is captured in closed loops (GeoSLAM, 2016). This is why multiple rooms are scanned multiple times. These different scanning times are split into different data frames as proposed by Litomisky and Bhanu (2013).

3. METHOD

The proposed method is based on the two datasets of the MLS device that are simultaneously produced during the data acquisition: the trajectory and the point cloud, as can be seen in figure 1. The proposed method starts with the voxelization of the point cloud. As described by Vo et al. (2015), voxelizing a point cloud has two advantages. Firstly, it introduces a spatial structure and secondly, it decreases the amount of data points which improves the processing speed. After the voxelization, dynamic objects that were present during the data capture are detected and removed. The next step consists of the classification of the trajectory into three types: stairs, flat and sloped surfaces. This is done by using different angle parameters between the successive points. The correctness of this classification depends on the changes in height of the MLS device during the data capture and therefore this classification is only a first indication of the type of floor element. The analysed trajectory is then voxelized to the same voxel space as the voxelized point cloud. By projecting these trajectory voxels on the voxelized point cloud, seed voxels are identified. The next step is the identification of doors. Doors form one of the two stopping criteria in the region growing process. This is important because users will navigate to a specific room inside a building and not to multiple rooms at once. The seed voxels and identified doors are further processed by a region growing algorithm implemented in a PostgreSQL database to form floor regions in every room. It is assumed that all the remaining voxels above the floor regions represent furniture objects. To derive the navigable voxel space inside a building, these furniture objects need to be marked as non-navigable spaces.

The last step consists of several checks of the identified navigable voxel space. First, the classification into flat floors, stairs and slopes is checked by analysing the ordered seed points through the floor regions. Second, small gaps in the final navigable space are filled to form the final output data. These steps provide the final navigable voxel space.

The point cloud and trajectory are captured using a Zeb Revo laser scanner. The method is implemented using Python 2.7 and PostgreSQL and tested on a DELL laptop running windows 7 with an Intel(R) Core i7-6820HQ CPU at 2.70 GHz, 16.00 GB RAM and a 250 GB solid state disk (SSD).

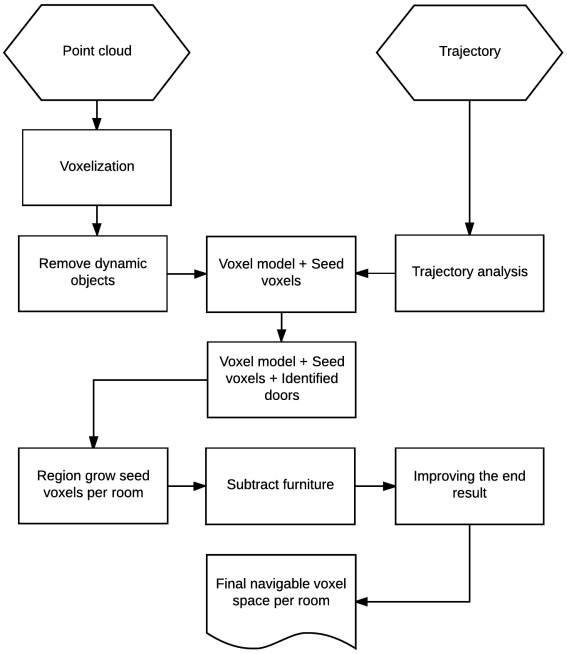

Figure 1. Overview of the proposed methodology

4. IMPLEMENTATION

This section introduces the proposed methodology step by step. This methodology is split into two parts. The first part discusses the reconstruction of navigable voxel space per room, classified as the three types: stairs, flat and sloped surfaces. The second part presents the different cleaning methods that are necessary to arrive at the final navigable voxel space. After these two steps, the final navigable voxel space is identified.

4.1 Surface reconstruction

The surface reconstruction process of the proposed method starts with the voxelization of the point cloud. Afterwards, the trajectory is analysed to identify stairs, flat and sloped surfaces. Next, seed voxels are identified and these seeds are region grow into floor regions.

4.1.1 Voxelization: As discussed in chapter 2, the voxelization of a point cloud creates a spatial structure and reduces the amount of data points which increases the processing speed (Vo et al., 2015). The voxelization is implemented using the octree method introduced by Broersen et al. (2016). The smallest octree leaves are saved in the final voxel model. The size of this smallest leave depends on the amount of subdivisions of the octree structure. The method of Broersen et al. (2016) automatically scales and replaces the point cloud so that the entire point cloud fits to a dimension array of 0 to 2 times the power of the number of subdivisions. The point cloud is scaled until the smallest voxel has a size of 1. The voxel size of 1 is an integer value which is used for calculation purposes. This value corresponds to a specific voxel size in cm. If a smaller voxel size is required, the octree needs to

be subdivided once more. The whole point cloud is translated so that all the points are located in the quadrant where both x and y are positive. This method is scalable to any size of point cloud with an almost linear increase of processing time (Broersen et al., 2016). The smallest none-empty octree leaves are stored in a PostgreSQL database.

The voxel size has a large influence on the representation of the indoor space and the final product. If a threshold below a door needs to be distinguished from the rest of the floor, the voxel size should be very small. In this research the different risers of a stair need to be identified and therefore, the voxel size can be bigger. According to the ISO 21542 (2011) standard, the risers of a step of a stair should not be higher than 15 cm. If a riser of a stair and the voxel size are both 15 cm, a riser is represented by one voxel. If noise is present before the step, a voxel is added and the riser is instantly deformed. Since the risers should always be identifiable, a smaller voxel size is needed. To identify risers of 15 cm, a voxel size of 5 to 7, cm is chosen. Defining a specific voxel size is currently not possible due to the automatic scaling of the point cloud as described above. This parameter can vary with respect to the type of stair.

4.1.2 Trajectory analysis: The classification of the trajectory forms the basis of the following processing steps. The trajectory is classified into three types of surfaces: stairs, flat and sloped surfaces. In order to identify these different elements, the angles between successive trajectory points are analysed. There are different angle parameters for stairs and slopes and for each surface type there are a number of fixed successive points that fulfil the angle criteria as described in table 1. These values are derived from numerous tests performed on the trajectories of different point clouds. The classification of stairs and flat surfaces is visualised in figure 2.

Type	Min. angle in degree	Max. angle in degree	Connecting elements
Stair	7,1	60	4
Slope	2,3	18,4	2

Table 1. Thresholds for the trajectory analysis

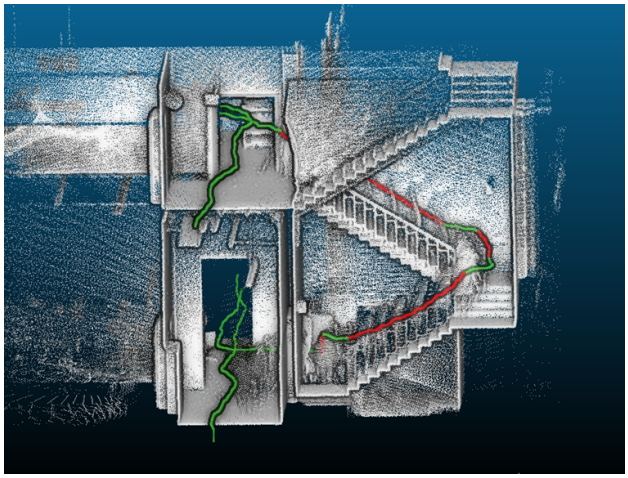

Figure 2. Analysed trajectory in the point cloud: green represents a flat surface and red represents a stair

4.1.3 Voxel model + seed voxels: The identification of seed voxels is done by voxelizing the classified trajectory in the same spatial structure as the voxelized point cloud. Seed voxels are identified by projecting the voxelized trajectory onto the voxelized point cloud. The records of these seed voxels are updated with the notation of being a seed voxel combined with the trajectory type of: flat, stair, slope or mixed as illustrated in figure 3. The mixed class is introduced because one voxel of the voxelized trajectory can contain multiple types of points. If halve of the points in a voxel are classified as a stair and the other half as a flat, it is unclear what the voxel type should be. In that case the voxel is marked as a mixed class.

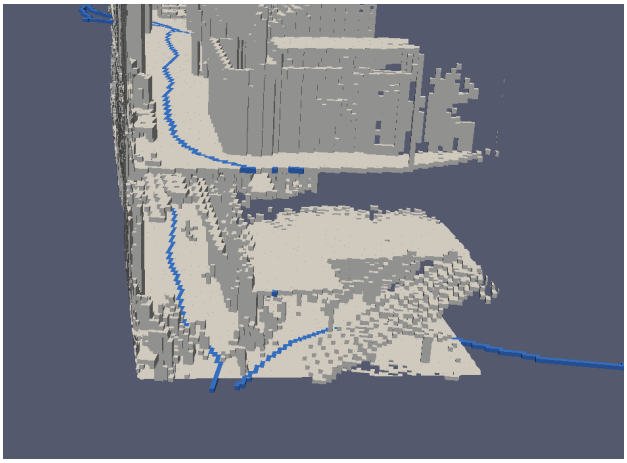

Figure 3. Identified seed voxels in the voxelized point cloud: blue corresponds with a flat surface

4.1.4 Voxel model + seed voxels + identified doors: The destination of an indoor navigation activity is most of the time located inside a specific room in a building. Therefore, it is important to distinguish different rooms in the final output data model. Doors form the connecting elements between different rooms. When the doors are identified, these rooms can be split during the region growing process. The doors are detected by the height difference between the trajectory seed voxels and the ceiling. A door is found if there is a sudden drop in the distance between the seed voxels and the corresponding ceiling voxel. Furthermore, the distance to the right and left in the x and y plane, first needs to be big first, than small and then big again for a few voxels. Both criteria are calculated based on the trajectory seed voxels, by computing the distance to the ceiling or the left and right voxels. The identified doors are a stopping criteria in the region growing process.

4.1.5 Region grow seed voxels per room: The final stage is the region growing process in which the floor regions will be generated from the identified seed voxels. The region growing process is implemented in the PostgresSQL database using the ST_ClusterDBSCAN algorithm (PostgresSQL, 2017). This algorithm only processes 2D clusters in one x,y (or x, z) plane. Therefore, the x,y regions are obtained per unique seed voxel height. Only the regions that contain trajectory points are saved. The region growing process is implemented to consider the eight neighbour adjacency of a seed voxel at one height value (so only on the x, y plane). There are two stopping criteria for the region growing process. First, the region growing stops when a door voxel is reached. Second, voxels with 2 voxels above the possible region voxel are not added to the current region.

4.2 Cleaning

The cleaning consists of four steps. First, dynamic objects that were present during the data capture are detected and removed.

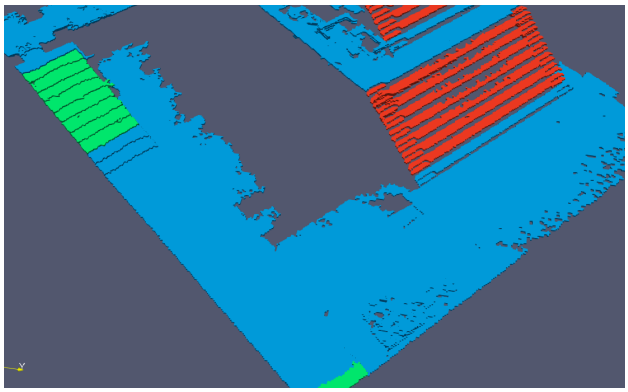

Figure 4. Regions in the voxel model: green represents a slope, blue represents a flat surface and red represents a stair

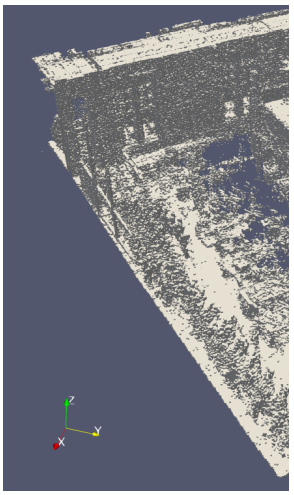

Figure 5. Uncleaned voxel model

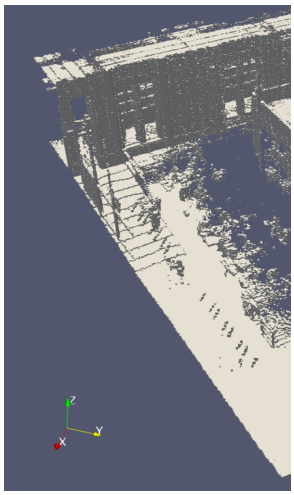

Figure 6. Cleaned voxel model

Second, the classified types and generated surface regions will be checked on their correctness. After this, small gaps in the floor regions are filled with new voxels. The final relates to the subtraction of furniture from the floor regions. In this way, the final navigable voxel space, in which pedestrians can navigate without any obstructions, is realised. The following sections provide further details on the individual steps.

4.2.1 Remove dynamic objects: The Zeb Revo, which is used for the data capture, uses a SLAM algorithm to register the different scans in one point cloud. Each point in the point cloud contains its original scanning time. A dynamic object is only present on a specific position for a short amount of time. Therefore, the points that represent the pedestrian form a long-drawn shadow (Józsa, 2012). Each voxel in the voxelized point cloud contains a different number of points. To detect the dynamic objects, the points inside a voxel are grouped by each second. If a voxel contains less than 2 scanning seconds, the voxel is marked as a dynamic object and removed from the voxelized point cloud. This value is based on various tests. These tests showed that whole dynamic objects were removed when this value of 2 was used. Increasing this threshold resulted in the partly detection of dynamic objects. When the parameter was increased parts of the model that were not scanned thoroughly, like the borders of the point cloud, were also removed. To be able to keep the most data, this threshold needs to be as low as possible. Cleaning the voxelized point cloud removes most dynamic objects in the voxel model, as can be seen in figure 5 and 6.

However, there is still some residual noise left, as illustrated in figure 6. This is caused by dynamic objects that were on the same position during the data capture. When a a dynamic object stays at the same position during the data capture, it is better represented in the final point cloud. A better representation means that more points per voxel were captured. Therefore, these objects are not detected by the described cleaning process.

4.2.2 Surface identification check: Sometimes elements are wrongly classified. The first riser and part of a slope can be defined as flat surfaces, see figure 4. The MLS device is held before the data gatherer. If a stair is entered, the scanner can already be above the second riser before the data gatherer enters the first riser. This results in a rise of the trajectory above the second riser. Based on the trajectory, this first riser cannot be detected. Moreover, furniture objects can be marked as flat surfaces, see figure 7. Furniture objects can be wrongly classified because the MLS is held above furniture objects during the data capture. If the trajectory is projected down on top of the voxelized point cloud, the furniture voxels are marked as seed voxels and region grown which turns them into floor regions.

To detect these wrongly classified elements, the change in height of the identified seed voxels is checked. If there is a sudden height change which is larger than the ramp and stair parameters, which are defined in table 1, it represents a wrongly classified element which can be detected and corrected. If the change is smaller than the ramp and stair parameters, risers or the first part of the slope can be detected.

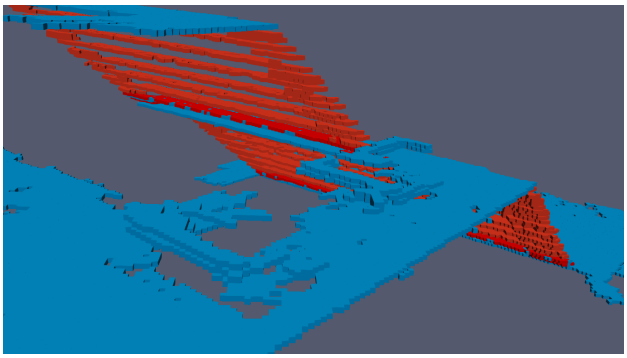

Figure 7. Region grown surface containing furniture as a floor

4.2.3 Repair gaps in the voxel model: Due to the occlusion of dynamic objects, missing data or the removal of some voxels during the cleaning, described in section 4.2.1, gaps can emerge in the floor regions. This is by ordering the individual floor regions and group them by the y-axis in the database. This way, the x-axis can be checked. If the distance between two following x positions is less than two voxels, it represents a gap and will be filled with new voxels. This process is repeated for the y-axis. The filling distance of two voxels depends on the voxel size and the filling requests of the operator. In this research this parameter is set manually but can be calculated in future developments.

4.2.4 Subtract furniture: The MLS device scans the whole interior of a room. Therefore, also points below furniture objects are present. A navigable space is the space where a pedestrian can navigate without bumping into any obstacles. For this reason, the space below furniture objects needs to be marked as an

unnavigable voxel space. After cleaning the dynamic voxels in the voxelized point cloud, the resulting voxels above the floor are assumed to be furniture objects. Dynamic objects that were on the same position during the data capture are for now also marked as furniture objects. Such furniture objects will be removed from the floor regions to get the final navigable voxel space.

4.3 Final navigable voxel space

After the surface reconstruction and the cleaning, the final navigable voxel space is identified. The wrongly classified elements are corrected see 8, as can be seen in the image, the navigable voxels are removed from the green slope where residual dynamic objects were present, see figure 6. Further research is needed to detect these residual dynamic objects and to prevent that these objects are removed from the navigable space. Furthermore, the wrongly classified furniture elements should be removed.

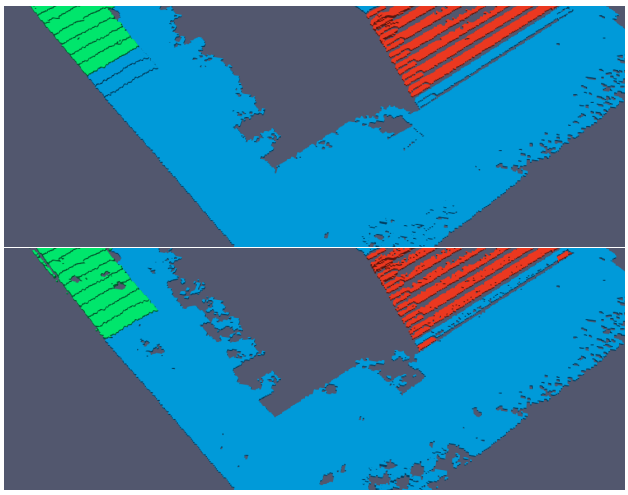

Figure 8. The regions of figure 4 (above) and the final navigable voxel space (below)

If the m^2 of the modelled final navigable voxel space is compared to the actual m^2 in the real world situation, the m^2 accuracy can be computed. The m^2 of the model is calculated based on the amount of floor voxels of its floor region. The m^2 of the real world situation is measured in a from furniture cleaned computer aided design (CAD) drawing of the same space. This is done for two different spaces in the indoor environment: a corridor and a part of the first floor. The results are illustrated in table 2. The difference between these two models is around 10%.

Checking type	Hallway in m^2	First floor Orange rock in m^2
CAD model	74.0	68.0
7.3 cm voxel model	67.7	61.3
Difference between CAD and 7.3 cm	-8.5 %	-9.9 %

Table 2. Area of the CAD model and the identified floor spaces with an voxel size of 7.3 and 3.7 cm

5. DISCUSSION

The performed tests have clearly illustrated that by combining the trajectory with the point cloud of the MLS important information can be derived to identify surfaces for navigation and that dynamic objects can be removed. It should be noted that in this implementation the voxel size depends on the number of subdivisions of the octree, as described in section 4.1.1. Therefore, each point cloud will have a different voxel size. Before point clouds of the same environment can be compared, they need to be rescaled to their original sizes. It would be useful to ask the operator for the preferred voxel size.

Furthermore, it is currently only possible to detect objects which moved during the data capture. Dynamic objects which remained static at the same position during the data capture cannot be distinguished yet. An example of such objects are humans waiting in queue near the coffee machine. Therefore, such objects are classified as furniture elements and removed from the final navigable voxel space. It is also possible that a room is scanned multiple times, see figure 9. If a dynamic voxel contains a dynamic object both times, the voxel contains points from both scanning times. In this case, the voxel contains more than two scan seconds and is therefore marked as static. To solve this problem, the individual points inside a voxel should be split into different time zones. The points are in a different time zone when the time difference of successive points is more than for example twenty seconds. In this case, each time zone inside the voxel needs to be checked again for the detection of these dynamic elements.

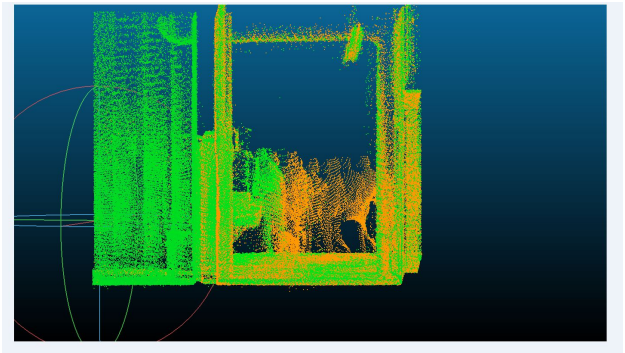

Figure 9. Multiple scanning times: orange is scan time one and green is scan time two

The classification of a stair, flat and sloped surface type is based on the trajectory analysis described in section 4.1.2 combined with the surface identification check described in 4.2.2. If the trajectory voxels with the seed voxels are compared, the x and y location are the same which results in a 2D image illustrated in figure 10. The z-value of the trajectory voxels is based on the original changes in height of the MLS device during the data capture (red line), whereas the z of the seed voxels (blue line) is based on the actual geometry of the point cloud itself. The green box in figure 10 is actually a small stair. The exact start and end point are not clearly identifiable in the trajectory voxels but they are identifiable in the seed voxels below. Furthermore, the green line shows that the trajectory rises later on a stair than the seed voxels do which results in the wrong classification of stair risers or slope parts. Therefore, the surface identification check is implemented. This requires a specific amount of processing time twice to get the right classification. If the classification is directly based on the seed voxel themselves, the classification and checking is done at once which is probably quicker than the currently implemented method.

The identification of doors is also based on the trajectory of the mobile laser scanner. In the currently implemented method, doors

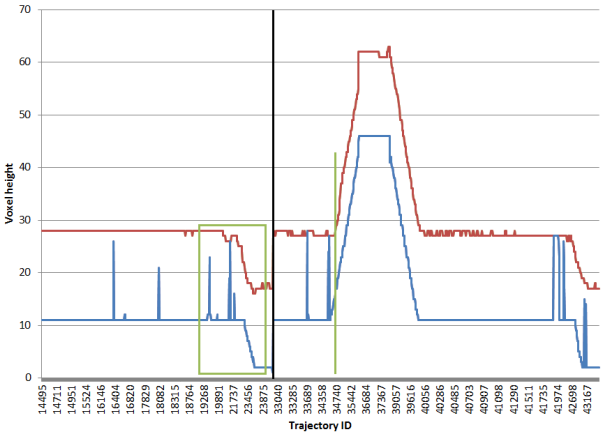

Figure 10. Trajectory voxels are represented in red, seed voxels are represented in blue per trajectory ID. The black line represents a split in the data for visualization purposes

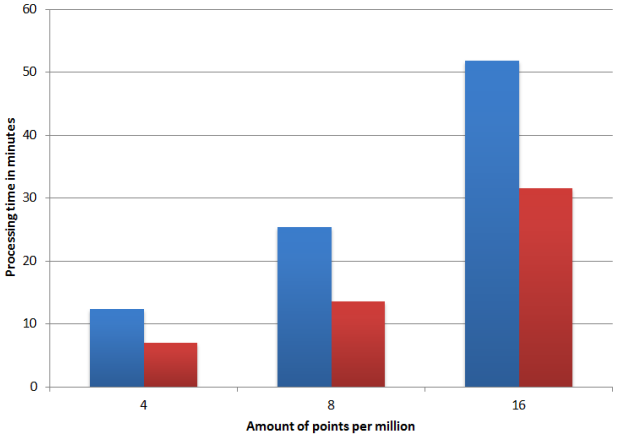

Figure 11. Processing time: blue is the total amount of time and red is the time which was necessary for the voxelization

can only be found when there is enough distance between the top of the door and the ceiling, is not yet implemented in an automatic way and doors can only be crossed during the data capture.

The region growing process is implemented in the PostgresSQL database using the ST_ClusterDBSCAN algorithm. This algorithm is quite fast and works fine for the voxelized point cloud because there is always one specific z-value for each floor region. The clustering algorithm only clusters the voxels of one z-value into regions. Therefore, the voxels that apply to the two stopping criteria need to be subtracted from the floor voxels before the clustering begins. Section 4.1.5 described the two stopping criteria: the first consists of the doors, which form the boundary of a room and the second consists of the subtraction of a voxel if it has 2 voxels above. In some cases, it is possible that these 2 voxels do not exist in the model but do exist in reality, for example in the case of a railing or a door of glass. In these cases, the stopping criteria is not applied and the region growing can continue underneath the railing or outside of a building or a room. This results in a over-estimation of the available navigable voxel space and the possibly merging of different rooms.

The cleaning parameter of less than 2 scanning seconds, as described in 4.2.1, is chosen after multiple tests. This parameter works for a 5 to 7.5 cm voxels but needs to be changed if a smaller or larger voxel size is chosen. If a smaller voxel size is used, a voxel contains less points which probably results in a smaller cleaning parameter. If the voxel size increases, the voxels contain more points which means that the cleaning parameter probably needs to be increased as well.

The processing time of the implemented method is illustrated in figure 11. When the amount of points is doubled, the processing time approximately doubles as well. As can be seen, the voxelization time represents more than half of the total processing time. The current method is tested on point clouds of 4, 8 and 16 million points. Future tests should be conducted with larger clouds. It is not suspected that this will cause any type of problem in the current implementation.

The indoor environment is highly cluttered with all kinds of furniture. If a good representation of the navigable voxel space is required, occluded areas behind chairs or tables also need to be scanned. If these spaces are not scanned, the navigable space cannot be detected behind these objects. This results in a lower amount of navigable space in the final model than it is in reality.

As discussed in 4.2.3, gaps in the floor regions are filled with new voxels if a gap is 2 voxels or smaller in length. This can lead to difficulties with boarding's of stairs because the space between a boarding and a risers can be filled up. In this case, the corners of the risers are not straight but rounded. A possible solution of this problem is to cross-check the repaired gaps with the original voxelized point cloud. If a match is found, the voxel is restored and otherwise it represents new data and can cause problems as described above. This can result in an over representation of the final navigable voxel space.

The difference in m^2 between the real world situation and the output of the model is around 10 %, as discussed in 4.3. This value is only based on two test cases of a floor region. To check its validity, more tests with different environments should be conducted.

Currently it is not possible to process different point clouds which have the same voxel size. As mentioned above, the voxel size cannot be controlled because it depends on the amount of subdivisions of the octree and the automatically scaling of the point cloud introduced by Broersen et al. (2016).

6. FUTURE WORK

This paper proposed a method to identify floors, stairs, slopes and furniture objects based on the point cloud and the trajectory of a MLS device. Future research could be done concerning the following topics:

1. Identification of walls: A stopping criteria during the region growing process is to stop adding voxels when 2 voxels directly above a voxel exist. This gives an indication for the location of possible walls. If the total number of voxels above the current voxel are counted until a specific height, it results in a heat map of the indoor space. By detecting linear elements, walls can be identified. A different approach for the detection of walls is by slicing the different rooms close to the ceiling in the x, y plane.
2. Identify furniture objects: All the resulting voxels above the floor regions are now marked as furniture objects. These voxels are just subtracted from the end result. It would be better to group these features and identify the type of furniture which is present.

A chair for example, can be located in the middle of the room but because it is more dynamic than a table it could also be replaced. This way, the furniture types and the usage of the space can be described in more detail.
3. Identification of dynamic objects, which do not move during the scanning: As discussed in section 5, static dynamic objects are now marked as furniture. Future research should be done on the identification and the removal of these elements from the voxelized point cloud since they can block a path which is normally available and thus should be included in the final result.
4. Generation of a node network: Navigation applications are mostly based on node networks. With the identification of doors, stairs, flat and sloped surfaces, such a network can be generated. It would be useful to extend this method with the automatic generation of navigation networks in a standard like IndoorGML.

7. CONCLUSION

The proposed method efficiently identifies three different kinds of navigable voxel spaces based on the trajectory and the point cloud of a MLS device. This method makes it possible to create a continuous navigable space in buildings including several floors, stairs and elevations. Data can be captured during business hours since the method detects and removes dynamic objects. The proposed method can be used for any type of room without any constraints because the complexity of the building is already present in the trajectory of the MLS.

ACKNOWLEDGEMENTS

This work has been supported by the Dutch project STW/M4S 13742 ‘Smart 3D indoor models to support crisis management in large public buildings’ (www.sims3d.net)

REFERENCES

Anagnostopoulos, I., Pătrăucean, V., Brilakis, I. and Vela, P., 2016. Detection of walls, floors, and ceilings in point cloud data. Construction Research Congress.

Boguslawski, P., Mahdjoubi, L., Zverovich, V. and Fadli, F., 2016. Automated construction of variable density navigable networks in a 3d indoor environment for emergency response. *Automation in Construction* 72(Part 2), pp. 115–128.

Broersen, T., Fichtner, F. W., Heeres, E. J., Liefde, I. d., Rodenberg, O. B. P. M., Verbree, E. and Voûte, R., 2016. Using a linear octree to identify empty space in indoor point clouds for 3d pathfinding. In: *19th AGILE Conference on Geographic Information Science*.

Brown, G., Nagel, C., Zlatanova, S. and Kolbe, T. H., 2013. *Modelling 3D Topographic Space Against Indoor Navigation Requirements*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–22.

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

Diakité, A. A. and Zlatanova, S., 2016. Valid space description in bim for 3d indoor navigation. *International Journal of 3-D Information Modeling (IJ3DIM)* 5(3), pp. 1–17.

Díaz Vilariño, L., Boguslawski, P., Khoshelham, K., Lorenzo, H. and Mahdjoubi, L., 2016. Indoor navigation from point clouds: 3d modelling and obstacle detection. Vol. XLI-B4, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences.

Fichtner, F. W., 2016. Semantic enrichment of a point cloud based on an octree for multi-storey pathfinding. Master’s thesis, Delft University of Technology.

GeoSLAM, 2016. Zeb-revo user’s manual v1.0.2.

Holenstein, C., Zlot, R. and Bosse, M., 2011. Watertight surface reconstruction of caves from 3d laser data. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3830–3837.

ISO 21542, 2011. Iso/fdis 21542: Building construction – accessibility and usability of the built environment. ISO.

Józsa, O., 2012. Analysis of 3d dynamic urban scenes based on lidar point cloud sequences. Master’s thesis, Budapest University of Technology and Economics.

Khoshelham, K. and Díaz-Vilariño, L., 2014. 3d modelling of interior spaces: Learning the language of indoor architecture. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*.

Li, G., 2014. Automatic detection of temporary objects in mobile lidar point clouds. Master’s thesis, University of Twente.

Litomisky, K. and Bhanu, B., 2013. *Removing Moving Objects from Point Cloud Scenes*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 50–58.

Macher, H., Landes, T. and Grussenmeyer, P., 2016. Validation of point clouds segmentation algorithms through their application to several case studies for indoor building modelling. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*.

PostgresSQL, 2017. 8.9. spatial relationships and measurements. $http://postgis.net/docs/manual-dev/ST_ClusterDBSCAN.html$ (7 April 2017).

Rabbani, T., Van Den Heuvel, F. and Vosselmann, G., 2006. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 36(5), pp. 248–253.

Suzuki, T., Kitamura, M., Amano, Y. and Hashizume, T., 2010. 6-dof localization for a mobile robot using outdoor 3d voxel maps. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5737–5743.

Thill, J.-C., Dao, T. H. D. and Zhou, Y., 2011. Traveling in the three-dimensional city: applications in route planning, accessibility assessment, location analysis and beyond. *Journal of Transport Geography* 19(3), pp. 405–421.

Turner, E., Cheng, P. and Zakhor, A., 2015. Fast, automated, scalable generation of textured 3d models of indoor environments. *IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING* 9(3), pp. 409–421.

Vo, A.-V., Truong-Hong, L., Laefer, D. F. and Bertolotto, M., 2015. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing* 104, pp. 88–100.

Yan, L., Liu, H., Tan, J., Li, Z., Xie, H. and Chen, C., 2016. Scan line based road marking extraction from mobile lidar point clouds. *Sensors* 16(6), pp. 903.

Zlatanova, S., Liu, L., Sithole, G., Zhao, J. and Mortari, F., 2014. Space subdivision for indoor applications. *Delft University of Technology, OTB Research Institute for the Built Environment*.