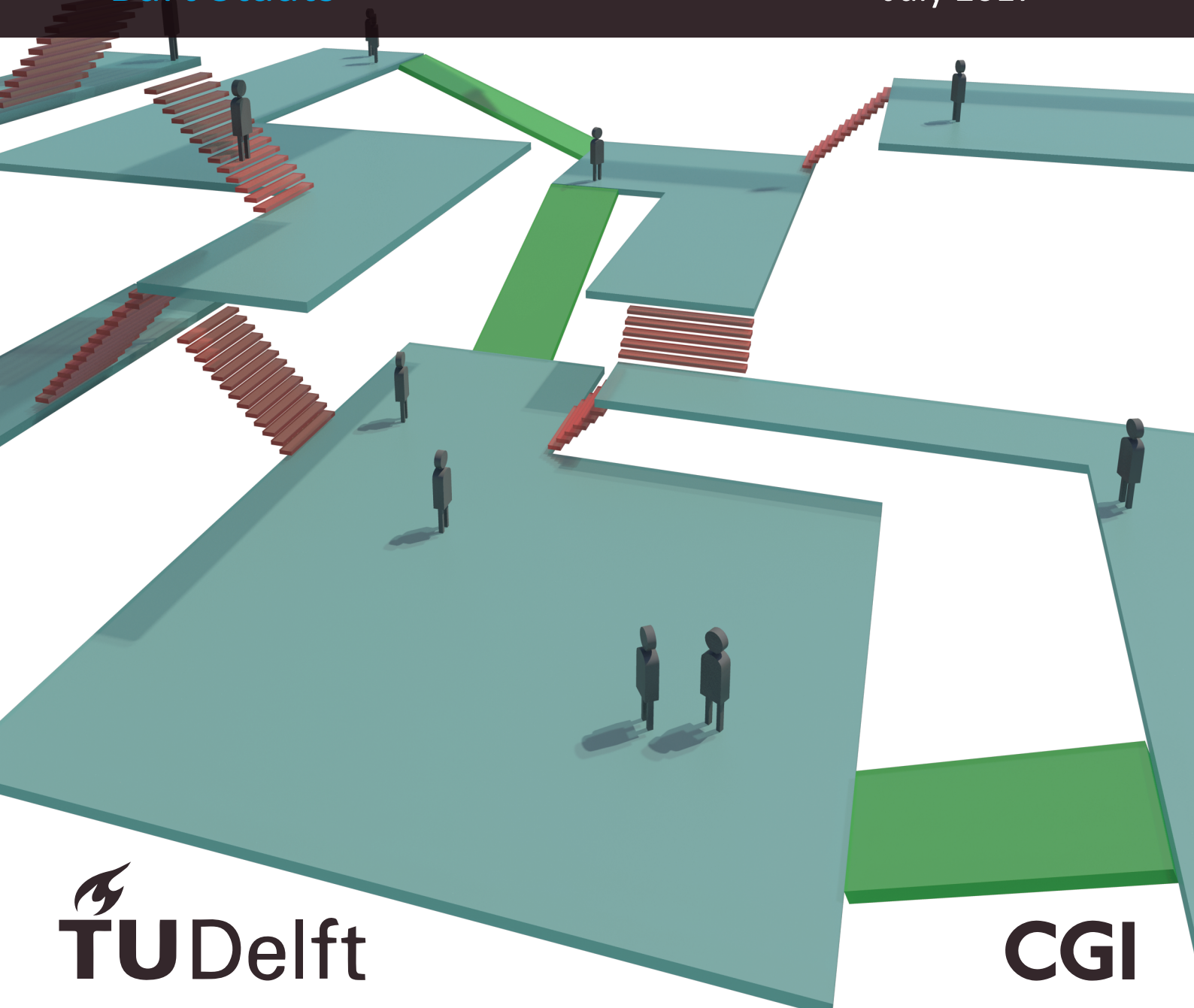
Master of Science Geomatics

Identification of walkable space in a voxel model, derived from a point cloud and its corresponding trajectory

Bart Staats

July 2017



TU Delft

CGI

Identification of walkable space in a voxel model, derived from a point cloud and its corresponding trajectory

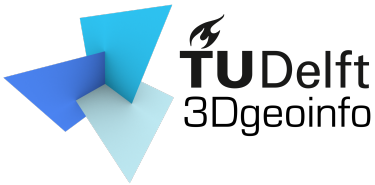
Delft University of Technology

Master of Science Geomatics for the Built Environment

Bart Staats

2016-2017

The work in this thesis was made in cooperation with:



3D geoinformation group
Department of Urbanism
Faculty of Architecture and the Built Environment
Delft University of Technology

CGI Nederland

Geometius

Supervisors:	Prof.Dr. Sisi Zlatanova
	Dr. Abdoulaye Diakité
	Robert Voûte (CGI Nederland)
Co-reader	Ir. Edward Verbree
Data collection:	Rob Bik (Geometius)

Abstract

Navigation from a room inside a building to another room inside a building which is across the street consists of three parts: first, the indoor part at the building where you start your journey. Secondly, the outdoor part and thirdly, another indoor part inside the building of destination. As regards to the outdoor environment, a navigation aid is well used and implemented in many types of applications. However, it is not common to use an aid to navigate inside a building. While such an indoor navigation aid is not necessary in small buildings, it is a necessity in more complex buildings like hospitals, airports, conference venues and large shopping malls. The indoor navigation aid can help visitors finding their way inside these, for them, unknown locations. The systems behind a navigation aid consist of several elements like an indoor positioning system, an indoor navigable map, specific destinations (points of interest) and an appropriate guidance throughout a building. This research focusses on the creation of indoor navigable maps that can be displayed and used to plan possible routes throughout the entire building. The indoor environment is far more complex than the outdoor environment. First, most people lose their orientation inside a building after they change their direction several times. Second, since there are no pre-existing routes inside a building, there are many different possible ways to arrive at a destination. Third, there is a large variety of associated spaces which all have their own unique interior design. Therefore, automating the process of making an indoor map is more challenging and time consuming than generating an outdoor map.

Most research in the field of automatic generation of indoor maps is focused on the already available 2D floorplans and only few of them use the more complex 3D representations. Using a 2D floorplans for the purpose of indoor navigation has its limitations because of various factors. Firstly, the 2D maps are already a simplification of the complex 3D environment which can lead to difficulties in representation. Secondly, the connectivity between different floor plans can be difficult as each floor plan is a separate entity. Thirdly, the maps that are available do not always contain furniture. Fourthly, in addition to the third one, most existing methods only focus on reconstructing the indoor space as an empty hull which results in a navigation aid which has does not emphasize the attention to obstacle detection. At last, most floorplans are out of date because some buildings are not built according to their blue prints. Their interiors might change after several years through the modification of walls and doors and furniture may be repositioned to the users preferences. Therefore, information about the indoor environment must be updated in most cases. This research concentrates on the automatic generation of indoor navigable spaces 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 terrestrial laser scanners.

To aid pedestrians in their indoor navigation, features needed for path computation such as floors, stairs, walls and furniture elements, need to be identified. These must be extracted from the point cloud which is generated by the MLS. How to identify these elements, like walls and doors, is investigated a lot. These researches are built on a set of constraints, like a Manhattan World or a flat surface constraint. These constraints are not problematic for a regular office building but they will provide difficulties in more complex buildings. This means that it is important to focus on a method with less or without constraints.

Scanning the indoor environment of an building often happens during business hours which automatically

leads to the inclusion of dynamic objects like pedestrians or small vehicles in the final point cloud. These dynamic elements do not represent any type of building elements (like furniture) and thus need to be identified and removed.

Beside the point cloud, the MLS also stores the trajectory of the MLS device. This trajectory contains three types of valuable information. The points directly below the trajectory indicate areas where pedestrians can walk, since the MLS device was operated by a pedestrian. The height difference between neighboring trajectory points can be used to detect stairs, slopes and flat surfaces and the trajectory also provides information about the connection of different surfaces and represents the complexity of the building.

In this research, a method for the identification of walkable surfaces based on the analysis of a point cloud and the corresponding trajectory of the MLS is developed. First, the point cloud is voxelized. Second, the trajectory is analyzed to detect the three different types of navigation surfaces: stairs, slopes and horizontal surfaces. This classified trajectory is projected vertically on the voxel model to acquire seed voxels. These seed voxels are then used to create areas by using region growing. These areas can be modified by identifying dynamic objects, entryways and furniture elements so that each area represents a specific navigable voxel space inside a building.

Experiments shows that by applying this method, it is possible to create a continuous navigable space in buildings including several floors, stairs and elevations. Data can be captured during opening hours because the method detects and removes dynamic objects in the final result. 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.

Preface

The MSc thesis which is lying here before you could not have been produced without the help of various people. Hereby I especially would like to thank all of my mentors who taught me during my time at the TU Delft, but in particular: Sisi Zlatanova and Abdoulaye Diakité, for helping me during the difficult steps which happened along the process of writing this thesis and for being my guides during my graduation. I would also like to thank my company mentor Robert Voûte, who was always enthusiastic, full of energy and had some great ideas about possible applications of my graduation research. You had brilliant ideas on how to improve the developed method and I am very grateful to had you as my mentor. I also enjoyed all the opportunities you gave me and the projects I was able to be part of during my time at CGI. Furthermore, I would also like to thank Edward Verbree for being my co-reader during my P4 and Rob Bik for generating the input data of this MSc thesis. I also would like to thank Svenja Staats for helping me by proofreading this thesis and developing it into this great end result. I would also like to thank Fanny Bot and Birgit Ligtvoet for working with me in our Inside-room at the Faculty of Architecture every Tuesday. I already miss these working days. Furthermore I would like to thank Lucía Díaz-Vilariño for the brainstorm sessions about various topics throughout the year. Also a special thanks to my lovely family who always supported me during my time at the TU Delft and the production of this MSc thesis. I could not have done it without you.

Contents

List of Figures

List of Tables

Abbreviations

GNSS	Global Navigation Satellite System
MLS	Mobile Laser Scanner
TLS	Terrestrial Laser Scanner
SLAM	Simultaneous localization and mapping
IMU	Inertial Measurement Unit
ISO	International Organization for Standardization
IFC	Industry Foundation Classes
OGC	Open Geospatial Consortium
MMS	Mobile Mapping System
CAD	Computer-aided Design
EPS	The desired distance between two points in the ST_ClusertDBSCAN algorithm

1 Introduction

Navigation from a room inside a building to another room inside a building which is across the street consists of three parts: first, the indoor part at the building where you start your journey. Second, the outdoor part and third, another indoor part inside the building of the destination (Thill, Dao, & Zhou, 2011). As regards to the outdoor environment, a navigation aid is well used and implemented in many types of applications. It is impossible for GNSS signals to be received inside a building, which is the reason why navigation aid is not widely available in an indoor environment. While such an indoor navigation aid is not necessary in small buildings, it is a necessity in more complex buildings like hospitals, airports, conference venues and large shopping malls. The indoor navigation aid can help visitors find their way inside these, for them, unknown locations. The indoor navigation systems behind a navigation aid consist of several elements like an indoor positioning system, an indoor navigable map, specific destinations (points of interest) and an appropriate guidance throughout a building (Boguslawski, Mahdjoubi, Zverovich, & Fadli, 2016; Brown, Nagel, Zlatanova, & Kolbe, 2013). Indoor positioning systems are developed by companies and are often based on new infrastructure systems. Therefore, these positioning systems are only available in certain buildings whereas a satellite system produces a worldwide positioning at once.
This research focusses on the creation of indoor navigable maps that can be displayed and used to plan possible routes throughout an entire building. The indoor environment is far more complex than the outdoor environment (Zlatanova, Liu, Sithole, Zhao, & Mortari, 2014). First, most people lose their orientation inside a building after they change their direction several times. Second, since there are no pre-existing routes inside a building, there are many different possible ways to arrive at a destination. Third, there is a large variety of associated spaces which all have their own unique interior design. Therefore, the production of an indoor navigation map is challenging and time consuming. A 3D model will solve most of these issues but the creation of a 3D model is also time consuming and requires an experienced operator (Gunduz, Isikdag, & Basaranera, 2016).

1.1 Problem statement

Most research in the field of automatic generation of indoor navigation maps is focused on the already available 2D floorplans and only few of them use the more complex 3D representations (Zlatanova et al., 2014). Using 2D floorplans to generate indoor navigation maps has its limitations because of various factors. Firstly, the 2D maps are already a simplification of the complex 3D environment which can lead to difficulties in representation. Secondly, the connectivity between different floor plans can be difficult as each floor plan is a separate entity (Zlatanova et al., 2014). Thirdly, the maps that are available do not always contain furniture. Fourthly, in addition to the third one, most existing methods only focus on reconstructing the indoor space as an empty hull. This results in a navigation aid which does not take obstacle detection into account (Díaz Vilariño et al., 2016). At last, as discussed by Turner, Cheng, and Zakhor (2015), most floorplans are out of date. Buildings are not always built according to their blue prints. Their interior might change after several years through the modification of walls and doors and the designed location of furniture elements may be repositioned to the users preferences. A possible solution for this problem is to update all the changes in the original map by hand. Obviously, this method

is very time consuming, inefficient and expensive. Therefore, up-to-date information about the indoor environment needs to be collected, which can be done by using a Terrestrial Laser Scanner (TLS) device. The interior of a building can now be captured more quickly (Díaz Vilariño et al., 2016). Because of the number of rooms and the occlusion of elements, capturing the whole building with a TLS is still time consuming and requires a lot of scan positions (Gunduz et al., 2016). Therefore, recent developments introduce the MLS device. This device scans the environment continuously along the trajectory of the operator which makes it more time efficient (Holenstein, Zlot, & Bosse, 2011). This way, scanning the inside of a building is now a matter of hours instead of days. Therefore, this research concentrates on the automatic generation of indoor navigable spaces for pedestrians based on laser scanning with a MLS device.

1.1.1 From point cloud to map

To aid pedestrians in their indoor navigation, features needed for path computation such as floors, stairs, walls and furniture elements, need to be identified. These must be extracted from the point cloud which is generated by the MLS. The identification process of these elements, like walls and doors, is investigated a lot. These researches are mostly built on a set of constraints, like a Manhattan World or a planar surface constraint (Anagnostopoulos, Pătrăucean, Brilakis, & Vela, 2016; Budroni & Boehm, 2010; Fichtner, 2016; Khoshelham & Díaz-Vilariño, 2014; Macher, Landes, & Grussenmeyer, 2016). These constraints are not problematic for a regular office building but they will cause difficulties in more complex buildings. This means that it is important to focus on a method with less or without constraints.

Scanning the indoor environment of an building often happens during business hours; especially in buildings that are never closed like hospitals or airports. This leads to the inclusion of dynamic objects like pedestrians or small vehicles in the final point cloud. These dynamic elements do not represent any type of building elements (like furniture or floor) and thus need to be identified and removed.

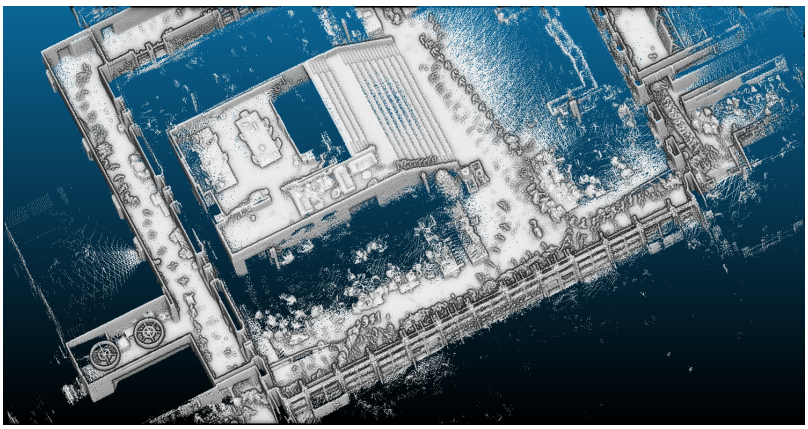
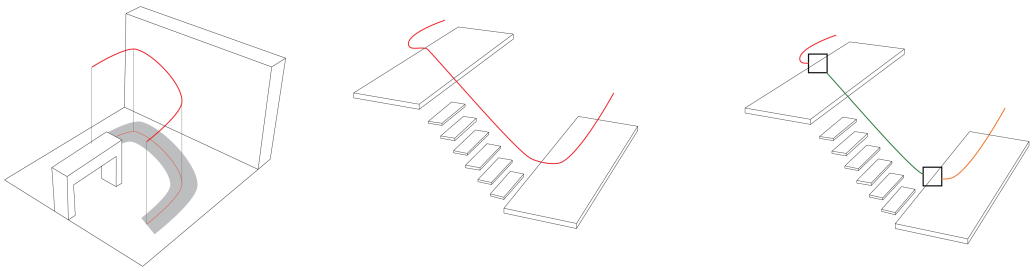


Figure 1.1: Point cloud contains dynamic objects

The captured point cloud contains the local x, y and z coordinates of the points and are represented in an unstructured space. A voxel model will be used to structure this space. A voxel space is generated from point clouds using structuring algorithms which groups the points based on their neighborhood (Vo, Truong-Hong, Laefer, & Bertolotto, 2015). Structuring a point cloud in this way has two advantages. Firstly, the number of voxels is much smaller than the number of points which makes storage, retrieval and processing more efficient. Secondly, as discussed before, the voxel grid possesses a spatial structure, which makes it easier and less time consuming to search for neighbouring voxels (Suzuki, Kitamura, Amano, & Hashizume, 2010b; Vo et al., 2015).
The representation of the voxelized point cloud depends on the voxel size. A small voxel size improves

the representation but at the same time increases the number of voxels and the computation time. A good balance between these two factors is important to get the right results in a reasonable amount of time.

Beside the point cloud, the MLS also stores the trajectory of the MLS device. The position of the scanner along this trajectory is used in other research to clean point clouds, to determine the scanner position or to improve a delaunay triangulation (Holenstein et al., 2011; Verbree & van Oosterom, 2001; Yan et al., 2016). If the MLS device is operated by a human, the trajectory contains valuable information, which can be used to distinguish different surfaces. As can be seen in figure 1.2a, the points directly below the trajectory indicate areas where pedestrian can walk (Li, 2014; Yan et al., 2016). The height difference between neighbouring trajectory points can be used to detect stairs, slopes and flat surfaces; see figure 1.2b. The trajectory also contains information about the connection between different spaces and represents the complexity of a building; see figure 1.2c.



(a) Walkable point below the trajectory (b) Stair identification by height differences (c) Connection between different objects

Figure 1.2: Available information in a trajectory

All this information is already available in the scanner trajectory but is not yet used in existing methods. Therefore, this master thesis investigates if it is possible to identify navigable space based on the analysis of the voxelized point cloud and the corresponding trajectory of the MLS device.

1.2 Research question

To get an answer on this topic, the following research question will be answered:

Which indoor walkable space can be identified from a voxelized point cloud using the trajectory of a mobile laser scanner?

1.2.1 Sub questions

Before this research question can be answered, the following sub questions need to be investigated:

1. What are the characteristics of a walkable space?
   The concept of space is different for everyone. Therefore, it is important to clearly define a walkable space.
2. In what way can the trajectory of a mobile laser scanner be used to identify a walkable space?
   The trajectory of a mobile laser scanner contains three types of semantical information as described in § 1.1.1. This sub question investigates in which way the trajectory information can be used to identify different types of walkable spaces.

3. How can walkable areas, identified by a mobile laser scanner trajectory, be subdivided into different spaces?
   A navigable map is used to create a route to a specific room inside a building. When the indoor model is created, it is important to clearly identify these different rooms. When rooms are merged together, pedestrians cannot navigate to these single rooms because they are not identified in the final model. Therefore, it is important to identify single rooms.
4. In what way does the voxel size influence the accuracy of the generated walkable space?
   In this method, the voxel model is used as a representation of the point cloud; see § 1.1.1. The size of these voxels influence the representation of the end result. A large voxel size could lead to a smaller or larger navigable space compared to the real situation. A small voxel size could represent the navigable space more accurate but will probably slow the classification process down. A good balance between the results and the voxelsize is therefore important.

1.3 Scientific relevance

As described in § 1.1.1, many approaches are based on a set of constraints like a Manhattan World or a planar surface constraint. Because the trajectory of the MLS is captured at the same time as the point cloud, the complexity of indoor environment is already present in the trajectory itself and less constraints need to be applied. To the author's knowledge, there is no other research that uses the trajectory of a mobile laser scanner for this purpose. Therefore, this approach is a new method in the field of the automatic reconstruction of interiors of buildings based on point clouds.

In the literature about path planning algorithms like the A* or the Dijkstra algorithm plan the shortest path from A to B see figure 1.3. However, there are more types of paths; a path with the most free space, a path with the nicest view or walking experience, a human preferred path and several other paths. The shortest path of figure 1.3 is not the most human preferable route. A human will not walk close to a wall if there is more free space in the hallway, even though this is the shortest path. The trajectory can be used to predict this path because the MLS trajectory is captured by a human and therefore represents a human preferred navigation path. This notion could be implemented in the following way:
If the pedestrian navigable space is identified in a voxel model, the voxels below the trajectory get a more preferred navigable state. If this improved model is used for path planning algorithms, these preferred voxels will be taken into account. Therefore, the route from A to B will follow a more human preferred path.

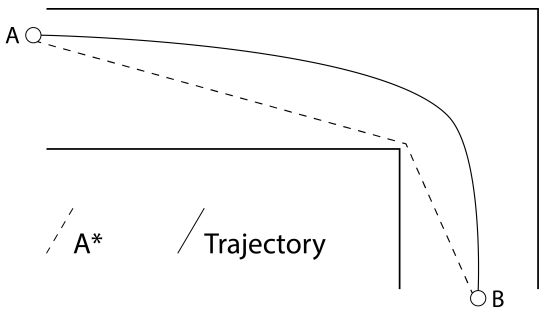


Figure 1.3: A* shortest path planning and a MLS trajectory

As described by Gilliéron, Büchel, Spassov, and Merminod (2004), significant advantages in navigation support is achieved by the generation of node networks in car navigation systems. These node networks or graph models can also be used to represent buildings where rooms are represented by nodes and hallways are represented by edges (Gilliéron et al., 2004). Nodes can represent among others stairs, doors and

elevators. Karas, Batuk, Akay, and Baz (2006) argue that the generation of node network models is even needed for optimal route calculation in indoor environments. By converting the identified navigable space into these nodes and edges, an indoor node network is generated which is suited for this type of navigation support. Furthermore, these node network systems require far less storage capacity than a voxelized model of the indoor space. This enables the creation of indoor navigation applications for devices with a limited storage capacity. These systems could also be suited for the guidance of multiple types of actors such as pedestrians, pedestrians with walkers, pedestrians in wheelchairs, cleaning cars and maintenance vehicles.

1.4 Scope

There are multiple fields of interest to investigate based on a voxelized point cloud and its corresponding trajectory. This MSc thesis investigates if the trajectory of a MLS can be used for the automatic generation of an indoor navigable voxel space which is used walking pedestrians.

1.5 Research goal

This MSc thesis focusses on the generation of a navigable voxel space. As discussed in § 1.1.1, features for path computation like floors, stairs, walls and furniture elements need to be identified. Walls do not represent an area which is walkable for pedestrians; they divide these walkable areas. Therefore, the identification of walls is not of interest for the identification of a walkable space. A lot of existing methods are based on constraints like a Manhattan World or a planar surface assumption. By combining the trajectory with the point cloud, it is assumed that these constraints should not be applied to this method. As earlier discussed, the detection of different spaces is also very important. Entryways form the connection between different spaces. Therefore, these entryways need to be detected and identified. If the data is captured during opening hours, dynamic objects can be present in the point cloud. These objects do not represent features needed for path computation and need to be detected and removed. These steps will be based, as far as possible, on the trajectory and the voxelized point cloud. Therefore, the goal of this research is to:

1. Identify different spaces by detecting the entryways of rooms
2. Detect dynamic objects from the point cloud that were present during the data capture
3. Identify three types of navigation surfaces: stairs, flat and sloped surfaces
4. Identify the navigable space by identifying and removing the furniture object above the navigation surfaces

1.6 Reading Guide

The research is described in different chapters. The theoretical background and the related work are introduced in chapter 2. The proposed method to derive the navigable voxel space can be read in chapter 3. After this, the introduced method is implemented and the results are discussed in chapter 4. In chapter 5, the research question will be answered. Besides the answering of the research question, the method will be discussed, conclusions will be drawn and the future work will be introduced.

2 Theoretical framework

In this chapter the theoretical background and the related work is discussed. First, the theoretical background about the production of the input point cloud and the several definitions of an indoor space will be introduced. Second, the related work of this thesis will be discussed.

2.1 Theoretical background

The first topic of the theoretical background discusses the algorithm that calculates the input point cloud. The second topic introduces diverse definitions of an indoor space.

2.1.1 SLAM

The point cloud of this research is generated using a Simultaneous localization and mapping (SLAM) algorithm. SLAM is developed in the field of robotics to generate a map of an unknown environment and to simultaneously locate a robot inside this map as well. Before SLAM, mapping and localization were studied independently. In a later stage, it was recognised that these two problems depend on each other. To be able to localize yourself in an unknown environment, a map is required and for the construction of a map which is precise, it is necessary to be properly localized (Fuentes-Pacheco, Ruiz-Ascencio, & Rendón-Mancha, 2015). That is why these two processes are happening simultaneously nowadays. The general approach of a SLAM algorithm consists of two basic elements (Durrant-Whyte & Bailey, 2006):

1. Detecting the change in position of a robot between the current and next position
2. Detecting the displacement of the corresponding landmarks between these two positions

Because of these basic concepts, poor featured environments like large empty spaces or tunnels with a smooth surface will cause errors in the final result. Environments with a lot of dynamic objects will also generate errors (Fuentes-Pacheco et al., 2015; *GeoSpatial SLAM*, n.d.). If there are enough static landmarks present, dynamic objects can be ignored without effecting the quality of the final point cloud (Bailey & Durrant-Whyte, 2006).

This method is implemented in the software which belongs to the MLS device; in this case the ZEB-REVO. This device contains a Lidar sensor to capture the environment and an Inertial Measurement Unit (IMU) to capture the movements of the device. The IMU is used to estimate the location roughly. The so called 'surfles' which represents shapes, are extracted from the corresponding point cloud. By combining the location and 'surfles' of the current and the next scanner position, the point cloud is extended and the position of the points is corrected (*GeoSpatial SLAM*, n.d.). To diminish drift in the final result, the data should be captured in closed loops (GeoSLAM, 2016). In this way, the start and finish 'surfles' can be matched and the accuracy of the point cloud of the entire loop will be improved (*GeoSpatial SLAM*, n.d.).
If a location is scanned multiple times, the SLAM algorithm combines all the measurements in one point cloud. The points in figure 2.1 all have a colour which corresponds with their scanning time. As can

be seen in the image underneath, the scanned environment exists of red, blue, light blue and yellow points. This indicates that the room is scanned four times and that all these scans are combined in one representation by the SLAM algorithm. As can be seen, the red coloured vehicle exists of only one colour and therefore was only present during the red scanning time. For each point, the scanning time is saved in the final point cloud.

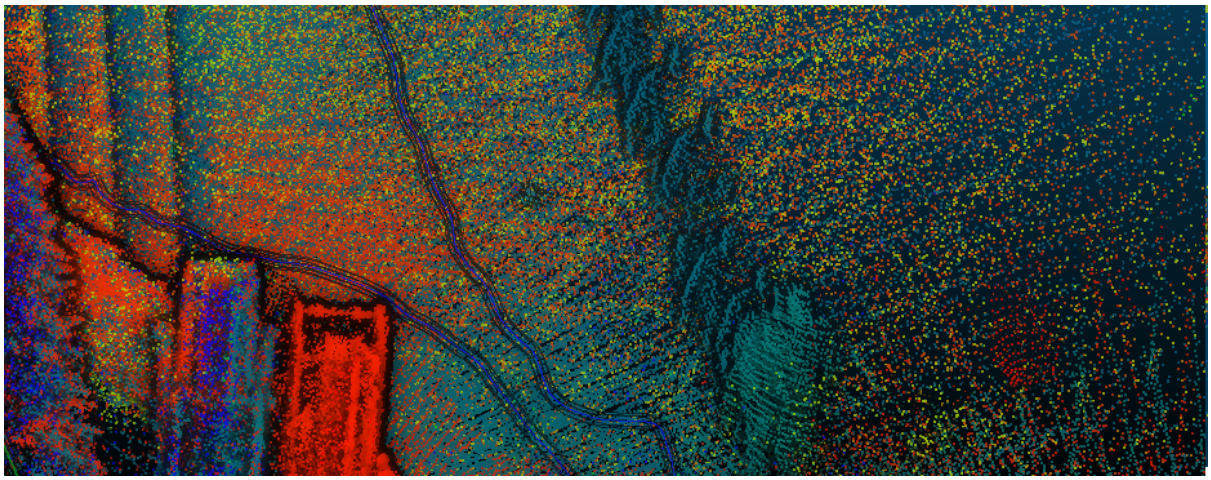


Figure 2.1: Point cloud coloured by scanning time

2.1.2 What is the definition of a space?

Most people understand the concept of space. However, forming a definition is far more challenging (Ekholm & Fridqvist, 2000). In this subsection, the different views of the concept of a space will be discussed. First, several definitions of a space will be introduced. Second, the different definitions will be compared.

A very conceptual description of a space is given by Ekholm and Fridqvist (2000): '*A space is by most of us thought of as an empty volume, enclosed in some respect - materially or experientially.*' In this context, a materially enclosed space means a space surrounded by physical objects like walls, ceilings or doors. Experientially enclosed spaces consists of non physical boundaries like a different colour of pavement or a different type of pavement.
The IFC standard, which is adopted in ISO 16739, describes a space as: '*A space represents an area or volume bounded actually or theoretically. Spaces are areas or volumes that provide for certain functions within a building.*'(building SMART, n.d.)
Ekholm (1996) describes a space for the construction sector as: '*A room in a building is a space with a free void that is large enough to accommodate users and equipment; the building parts that make up the space are enclosing, e.g., to climate, light, sound, or fire. A building space may also be experientially enclosing, i.e., dependent on a person's interpretation.*' (Ekholm, 1996; Ekholm & Fridqvist, 2000).
Eastman and Siabiris (1995) describes a space according to a conceptual framework. This framework can be extended depending on the desired goal and consists of the following four classes:

1. Building
   The Building class contains all the other three classes: Constructed Form, Bounded Space and Activity.

2. Constructed Form
   The Constructed Form describes all physical elements of a building like walls and floors but also electrical wires, plumbing, lighting and water distribution systems.

3. Bounded Space
   The Bounded Space is enclosed in the Constructed Form and represents the empty space inside a building.

4. Activity
   The Activity class describes which activity can take place in a specific Bounded Space. This definition depends on the goal of the model and research field

The IndoorGML standard created by the OGC is not concerned about architectural components or physical elements themselves: '*Indoor space is defined as space within one or multiple buildings consisting of architectural components such as entrances, corridors, rooms, doors, and stairs. In IndoorGML, we are not concerned about architectural components themselves (e.g. roofs, ceilings, walls) but instead the spaces (e.g. rooms, corridors, stairs) defined by architectural components, where objects can be located and navigate. IndoorGML is also concerned with the relationships between spaces.*' (Lee et al., 2016). In this definition architectural components are first described as entrances and corridors and later described as roofs, ceilings and walls. Entrances and corridors are then renamed as spaces defined by architectural components (roofs, ceilings and walls). There is a contradiction in this definition. Therefore it is assumed that architectural components are roofs, ceilings and walls and spaces describes rooms, corridors and stairs.
Zlatanova, Liu, and Sithole (2013) are describing a space with a focus on indoor navigation. According to them, a space is: '*the environment in which humans store resources (items of interest) and engage in navigation activities. ... A logical compartmentalisation of resources and navigation activities requires the creation of sub-spaces.*' The sub space is divided in two categories: 'Free sub-spaces' and (occupied) 'Inert sub-spaces' which are needed to enable indoor navigation (Zlatanova et al., 2013).

The concept of a space provided by Ekholm and Fridqvist (2000) is not specific which is why it forms the base for most other concepts. According to the concept of Ekholm and Fridqvist (2000), a space is bounded by structural and non structural elements. This definition of a space is also present in (building SMART, n.d.). The IFC concept of a space is extended by adding a specific function to each space. The addition of this function is also described in the conceptual framework of Eastman and Siabiris (1995). This framework forms a basic concept of a space that can be extended to the needs of the user. Ekholm (1996) is not connecting a function to a space but describes the size of a space according to the accommodation of equipment and users.

According to the four described concepts above, a space is a result of its architectural components which for the base of these definitions. IndoorGML has an opposite approach and (Lee et al., 2016) does not store the actual architectural elements themselves but they store spaces like entrances, corridors, rooms, doors, or stairs which can cross physical boundaries like doors. Zlatanova et al. (2013) is also describing a space like the IndoorGML standard. They subdivide a space in functional parts like sitting places, workplaces, navigation places and so on.

In summary, the definition of a space can be categorized into two groups. The first group describes a space as the result of architectural components. The second group describes a space as a specific entity with a distinct function. In most cases, the second concept category does not save the actual architectural components. Forming a general definition that describes all the different definitions of a space is difficult. This results in a general description such as Ekholm and Fridqvist (2000) introduced. This definition is in most cases not specific enough and rather vague. Therefore, the definition of a space will always be given if it is implemented in a research. To be able to work with the concept of a space, a clear definition is needed. This definition is introduced in § 3.1.

2.2 Related work

In this section, the related work is discussed. First, different approaches in building reconstruction techniques are investigated. Second, the detection of dynamic objects will be discussed. After this, navigation in robotics will be discussed. Subsequently, the voxelization and the identification of entryways will be introduced.

2.2.1 Building reconstruction

Indoor navigable spaces can be defined as the free surfaces that are used to navigate inside a building without bumping into any objects. To identify the indoor navigable space, building surfaces need to be reconstructed. The reconstruction of indoor environments is researched a lot and each research has its own approach. In this section, the different approaches of several researches in the field of 3D reconstruction are discussed.

A common approach for surface reconstruction is the calculation of the normal of each point in the point cloud. Rabbani, Van Den Heuvel, and Vosselmann (2006) designed a method that semantically enriches point clouds of industrial constructions through normal estimations, plane fitting and the analysis of the residuals of this plane fitting process. The method uses both the k-nearest neighbour and the fixed distance neighbour approach. By grouping points together which have the same normal, 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 the classification of doors, obstacle aware navigation is implemented by combining the planar regions together with the furniture points, see figure 2.2.

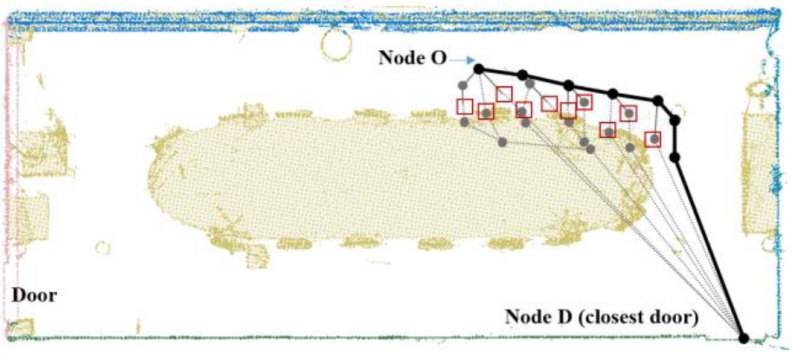


Figure 2.2: Shortest path planing based on furniture points and planar regions (Díaz Vilariño et al., 2016)

Budroni and Boehm (2010) developed a method to reconstruct buildings from point clouds using plane sweeps. First, horizontal plane sweeps are applied. If the amount of points in these plane sweeps reaches a threshold, floors or ceilings are detected. Second, a few vertical plane sweeps at random locations are applied. For each location a histograms is computed. The combined result of these histograms provides the main direction of all walls in the entire building. Third, the identified floors are decompositioned into cells. It is assumed that the floor is surrounded by walls. Therefore, the walls are extruded from the side of the floors into the final building model. This method only focusses on the reconstruction of flat surfaces, perpendicular and parallel walls (Budroni & Boehm, 2010).

Fichtner (2016) introduced a similar approach based on histograms instead of plane sweeps. This method uses an octree method proposed by Broersen et al. (2016) to structure the input point cloud. Because the octree method starts with a box shape, the method essentially creates voxel models with different voxel sizes.
Using a voxel model has several advantages. First, the number of voxels is much smaller than the number of points which makes storage, retrieval and processing far more efficient. Second, the voxel grid possesses a spatial structure, which makes searching for neighbour voxels quick and easy (Suzuki et al., 2010b; Vo et al., 2015). Since the voxelization is a generalization of the point cloud, the representativeness of the voxelized point cloud depends on the voxel size or in this case the amount of subdivisions of the octree structure.
The method of Fichtner (2016) first rotates the entire point cloud so that the direction of the walls and floors in the model are aligned with the x, y and z axis. After the voxelization, it continues by deriving histograms of the occupied leaves (or voxels) in the z direction. If a peak is detected, a possible floor is detected. The next step consists of wall identification by applying histograms to the x and y direction. Furthermore, stairs are detected using a filter based approach. This method only works on Manhattan worlds and it is difficult to distinct small floors in larger point clouds.

The histogram method is also implemented by Khoshelham and Díaz-Vilariño (2014) which produced a method for near Manhattan Worlds. The method exists of applying histograms to subdivide a building into floors. Secondly, walls are detected using vertical histograms just like the method of Fichtner (2016). Subsequently, cuboids are placed in the free spaces between floors, ceilings and walls. By merging these cuboids, the interior space is created and identified.

The proposed method by Vo et al. (2015) combines the octree approach of Broersen et al. (2016) with the smoothness approach of Rabbani et al. (2006). In this approach, the point cloud is structured in an octree/voxel model. The normal is calculated for the points in each voxel. These voxels are then grouped together according to this classification. This method improves the speed and accuracy of the classification of the final output model.

Turner et al. (2015) implements a approach which is based on a moving scanning platform and applies the raytracing method on the gathered data in the voxel space. The voxel with the gathered points are marked as boundary voxels. If a laser beam crosses through a voxel it is marked as an interior voxel. By partitioning the voxel faces into planar regions and representing these regions as a quad tree, a triangulation is performed. This way, a building representation which contains furniture is created. Besides this model, a second model without furniture is created based on the 2D floor plans. These floor plans are generated 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.

Koopman (2016) produced a method that creates an indoor navigation model for path planning based on a voxelized model of the indoor space. The method exists of the buffering of a space according to the dimensions of the actor in a so called dilated model. After this, the voxels are divided in a floor, stairs or obstacle class. The next step consists of cell generation using a distance transform and a watershed transformation, see figure 2.3. At the end of this step the voxels of each room are classified as one cell. The doors and stairs are then used for the creation of a network graph. The final step consists of the implementation of a path finding algorithm based on the derived graph and cells. Furniture and other objects in the indoor environment can generate problems during the distance transformation. Therefore, the cell generation can lead to over-segmentation.

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

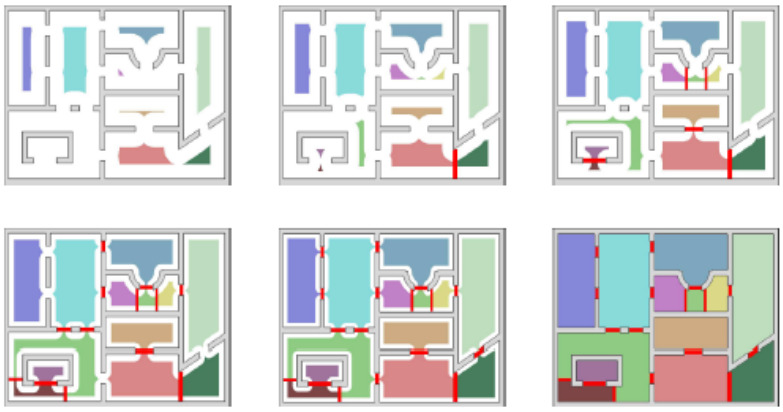


Figure 2.3: Watershed transformation (Koopman, 2016)

2.2.2 Detection of dynamic objects

Commonly public buildings like hospitals or airports cannot be closed when the data is captured. Therefore, the generated point cloud contains dynamic objects like people or small moving vehicles. These elements have an effect on the final output and should be filtered before further processing. Detecting moving objects is largely investigated in robotics and mostly done with specific technologies like range scans, stereo cameras, monocular cameras and recently with consumer RGB-D cameras. However, identifying dynamic objects from an already merged point clouds is not extensively explored (Litomisky & Bhanu, 2013). The detection of dynamic elements is even more difficult because they are only present on a specific location at a specific time, which makes these dynamic elements long-drawn shadows in the final registered point cloud (Józsa, 2012).

The approach for the detecting of dynamic objects described by Litomisky and Bhanu (2013) is based on splitting the point cloud into two data frames. These data frames should have a significant overlap and time difference. Large planes inside these data frames are subtracted, which is followed by a segmentation of the remaining points. The individual segmented clusters are then compared to the clusters of the other data frame so that the dynamic elements can be detected and removed.

A different approach is presented by Holenstein et al. (2011). They work with raytracing to divide the entire space into free and occupied voxels like Turner et al. (2015) does. By combining rays of multiple measurements, points from dynamic objects are identified and removed. Raytracing 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. Using raytracing methods are therefore not advised.

The detection of all dynamic elements in point clouds is difficult because the points form long-drawn shadows. Therefore, it is difficult to detect pedestrians based on their shape. To detect these elements, the above described methods can be used which assumes that a dynamic object was present on a specific location at a specific time. This key assumption will also be used to detect dynamic objects in this thesis.

2.2.3 Navigation in robotics

The detection of navigable space and obstacles are basic features in the research field of robotics where a lot of different approaches and methods exists. In this section, three voxel based approaches are discussed.
The method of Adán et al. (2015) automatically scans the interior of a building using a mobile laser

scanner which is attached to a moving platform. This process starts with the creation of a histogram of the first scan. Next, the roof and floor are identified and a flatness map is created based on the calculation of the normals of the points. Afterwards, a bounding box is created around the roof and floors. The points inside the bounding box are voxelized and classified as occupied, free or occluded voxels based on raytracing. Thereafter, the next best scan position is determined. This is the location where based on the flatness map the robot navigates to, which changes the most occluded voxels into occupied or free voxels. This process continues until the room is completely scanned Adán et al. (2015).

Maier, Hornung, and Bennewitz (2012) implemented a real time navigation application. Their method exists of two maps: a static voxel map, which is captured before the navigation, and a dynamic voxel map. The dynamic voxel map is used for path planning in a non-static environment and is continuously updated using depth cameras. Then, the path to the destination is planned. For the robustness and the performance of the path planning process, the path is based on a 2D map. This map is created by projecting the 3D dynamic voxel map on the floor.

Suzuki, Kitamura, Amano, and Hashizume (2010a) implemented an approach to test the positioning capabilities of a moving robot. They did this by first scanning the outdoor environment with a Mobile Mapping System (MMS) on a car. After this, the 3D model is voxelized. To accurately represent the road surfaces which can be used by a robot, a voxel size of 10x10x1 cm was chosen. A robot with various sensors was manually driven to test its capabilities to position itself inside the pre captured map.

The raytracing and the histogram methods are also applied in building reconstruction processes. To navigate a robot, real-time information is required to stay free of static and dynamic objects. For this purpose there are also different techniques like depth cameras, range scans and stereo cameras applied see § 2.2.2. Since this thesis uses data collected with a laser scanner, these other technologies cannot be used.

2.2.4 Voxelization

As can be read in 2.2.1, voxelizing a point cloud has it advantages. The voxelization process can be done in different ways. The voxelization method described by (Nourian, Gonçalves, Zlatanova, Arroyo, & Ohori, 2016) firstly detects the bounding box of a point cloud. Secondly, it subdivides this bounding box into voxels. Thirdly, each voxel is checked if it contains points. The voxel is saved when it contains a point. To reduce the amount of saved data, not all the vertices of a voxel are stored. In most cases the width of the voxel is saved together with a specific point which represents the centre or one of its eight corners. In this way, voxelizing gives the position of the voxels without a data structure. A data structure is required to process the data efficiently during further processing steps.
Another voxelization method is based on an octree structure as described by Broersen et al. (2016). By generating an octree, a cube or voxel is subdivided into equal smaller cubes. This process is done for each point in the point cloud until a specific number of subdivisions is reached see figure 2.4. By saving the cube numbers for each subdivision, the locationalcode, a data structure is created. This data structure allows to retrieve information about a parent cube at a later time. It also contains information about the specific neighbouring cubes on each level.

2.2.5 Entryway identification

The detection of entryways can be applied in different ways. The method of Koopman (2016) uses cell generation and a distance transformation as described earlier. The method of Fichtner (2016) uses a filter based approach see § 2.2.2. Entryways can also be detected based on the trajectory of a MLS. (Nikoohemat, 2016) detects entryways by voxelizing the point cloud and applying three basic rules, see figure 2.5:
1. An entryway center is an empty space
2. Above the entryway center points are present

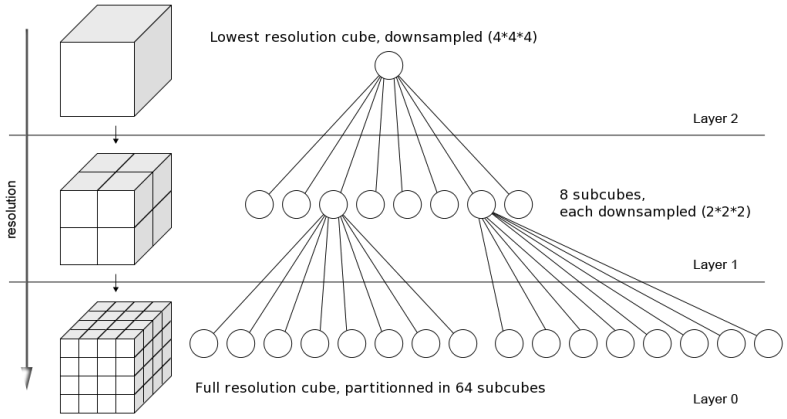


Figure 2.4: Octree structure (R. Dumusc et al., 2013)

3. There should be a trajectory close by

With these three rules entryways can be identified in an efficient way (Nikoohemat, 2016)

Figure 2.5: Door centre detection (Nikoohemat, 2016)

3 Methodology

In this chapter the used method is introduced. First, the definition of a space is discussed. Second, an overview of the proposed method is given. Third, the different steps which are discussed in the overview are explained in more detail.

In summary, the definition of a space can be categorized into two groups. The first group describes a space as the result of the architectural components. The second group describes a space as a specific entity with a distinct function. In most cases, the second concept category does not save the actual architectural components. Forming a general definition that describes all the different definitions of a space is difficult.

3.1 The definition of a space

As discussed in § 2.1.2, a space can be categorized into two groups. The first group describes a space as the result of architectural components. The second group describes a space as a specific entity with a distinct function. This last group describes the meaning of a space like an entrance, rooms and corridors as a semantic value which can cross architectural components. As described in § 1.1, the focus of this MSc thesis lies in the identification of features needed for path computation such as floors, stairs and furniture elements. Therefore, this MSc thesis is focussing not primary on the architectural components as identified in the first approach but is based on the second approach as discussed in § 2.1.2.
The indoor environment is used by actors. Actors are clients that engage in a certain indoor navigation activity. These clients exists of pedestrians, robots or remotely-controlled vehicles (Zlatanova et al., 2013). As discussed in 1.4, this thesis will only focus on pedestrians which from now on are called actors. An actor is described by its following specific dimensions: width, depth, height and the specific types of navigation surfaces it uses. For example, a pedestrian with a walker can only navigate on horizontal and sloped surfaces but cannot navigate on stair surfaces because it is based on wheels. Therefore, this navigation surface type is excluded for the navigation activities of this specific actor.
A space is described as:

- A section of the indoor environment which is enclosed by walls, entryways, windows, floors and ceilings that can be accessed by an actor through an entryway.

If this definition is applied to an indoor environment, a building is split into different spaces. These spaces contain different components like furniture elements, lightning features, plants, construction elements and are used for indoor navigation activities, see figure 3.1a. A single space can contain multiple navigation surfaces. As described earlier, an actor with a walker cannot navigate on stair surfaces but can navigate on horizontal or sloped surfaces. Therefore, the space needs to be subdivided into different subspaces based on the type of its navigation surface.
A subspace is described as:

- A section of a space which only contains one type of connected navigation surface.

As can be seen in 3.1b, this space exists of three subspaces: two horizontal surfaces and one sloped

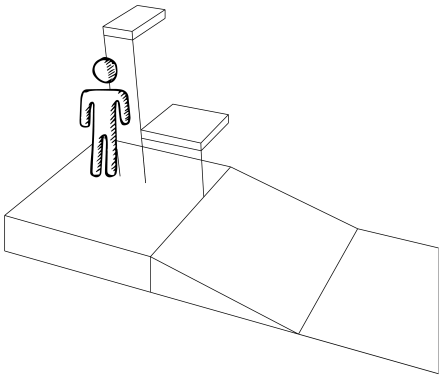
surface. These subspaces still contain building and furniture elements above the navigation surfaces which are not navigable for an actor. Therefore, these elements need to be removed from the navigation surface.
A navigable subspace per actor is described as:

- The subspace of a space which consists of one type of connected navigation surface where an actor, with a specific height, can navigate without colliding into any element.

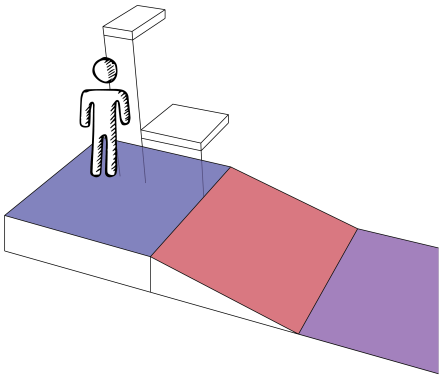
As can be seen in 3.1c, the height of the actor is lower than the air vent but higher than the furniture element. Therefore, the furniture element is removed from the subspace and the air vent is not eliminated. This results in the navigable subspace per actor.
A navigable space per actor is described as:

- A section of the indoor environment which is enclosed by walls, entryways, windows, floors and ceilings that can be accessed through an entryway (the collection of subspaces) where an actor, with a specific height, can navigate without colliding into any element.

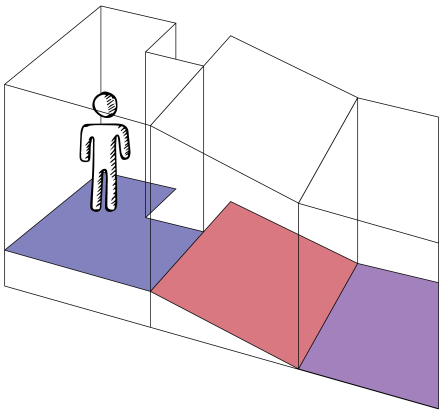
In this way, a 3D navigation environment emerges in which an actor can navigate without bumping into any obstacle, see figure 3.1d.



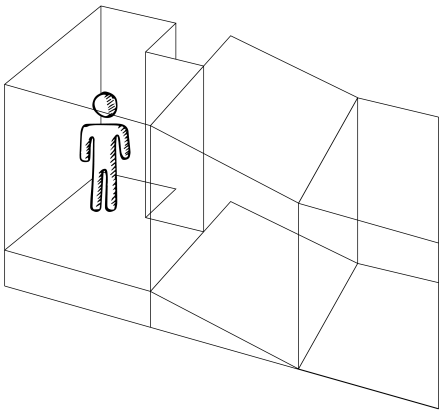
(a) An indoor space with an air vent and a furniture element



(b) Three subspaces with an air vent and a furniture element



(c) Three navigable subspaces per actor



(d) A navigable space per actor

Figure 3.1: Space, subspace, navigable subspace and navigable space

3.2 Research method

The proposed method is based on the two datasets of the MLS device that are simultaneously produced during the data capture: the point cloud and the trajectory of the MLS, as can be seen in figure figure 3.2. 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. First, it introduces a spatial structure and second, it decreases the amount of data points, which improves the processing speed of the proposed method. After the voxelization, dynamic objects that were present during the data capture are detected and removed. Following, small gaps in the voxel model are filled. The next step consists of the classification of the trajectory into three types: stairs, horizontal and sloped surfaces by using the stair and slope angle parameters between successive points. This classification is only a first indication of the type of floor element because the correctness of this classification depends on the changes in height of the MLS device during the data capture. The analysed trajectory is then voxelized in the same voxel space as the voxelized point cloud. The so called seed vooxels are identified by projecting these trajectory voxels on the voxelized point cloud.
During the next step entryways like doors, sliding doors and experientially entryways are identified. The identification of entryways is important because users will navigate to a specific space inside a building and not to multiple spaces at once. An entryway forms the connection between these different spaces and therefore needs to be detected and present in the final model. These entryways form one of the two stopping criteria in the region growing process. The seed voxels and identified entryways are further processed in a region growing process and implemented in the PostgreSQL database to form floor regions in each room.
The following step consists of checking the classification of the regions by analysing these seed voxels. This is based on the height differences of the seed voxels described by the slope and stair angle parameters used in the trajectory classification.
It is assumed that all the remaining voxels above the floor regions represent building or furniture elements. To derive the navigable voxel space inside a building, these furniture elements need to be marked as a non-navigable space. The voxels above the floor regions are considered furniture elements when they are lower or match the height characteristic of an actor. This results in the final navigable voxel space per actor, see figure 3.2 and figure 3.3. The point cloud and trajectory are captured using a Zeb Revo laser scanner provided by Geometius (2017).

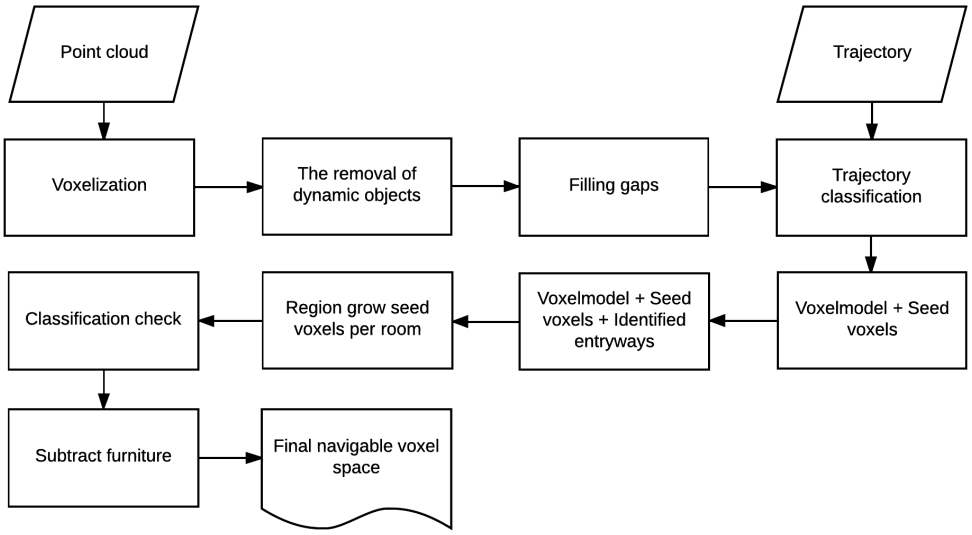


Figure 3.2: Proposed method to derive the navigable voxel space per actor

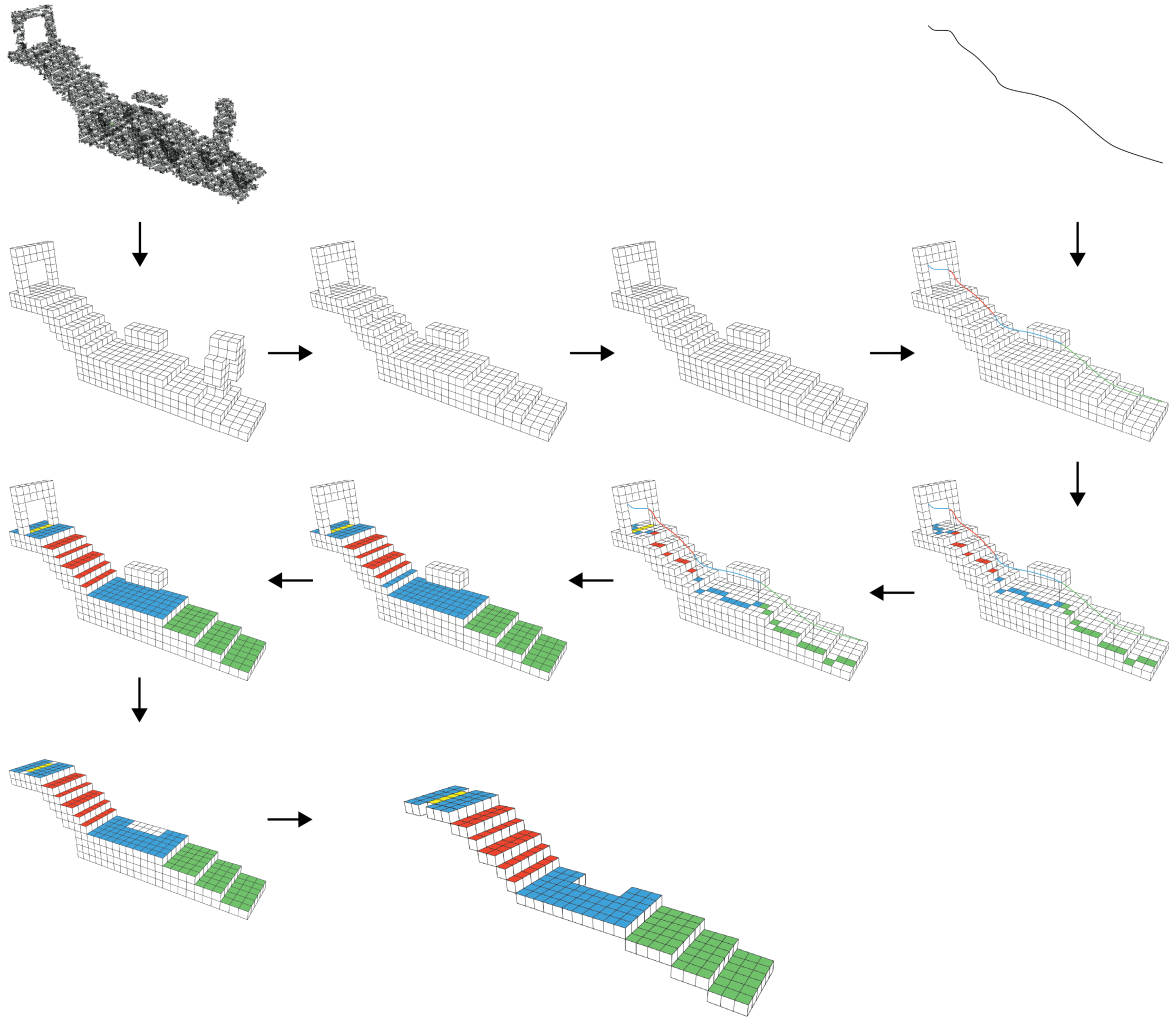


Figure 3.3: Proposed method explained in images

3.2.1 Point cloud and trajectory

The input data consists of the two datasets of the MLS: a point cloud figure 3.4 and the corresponding trajectory figure 3.4. The data has been captured in the Orange hall of the Faculty of Architecture and the Built Environment at the Delft University of Technology. In the centre of the room multiple groups of tables are located and the space can only be accessed by stair or sloped surfaces.

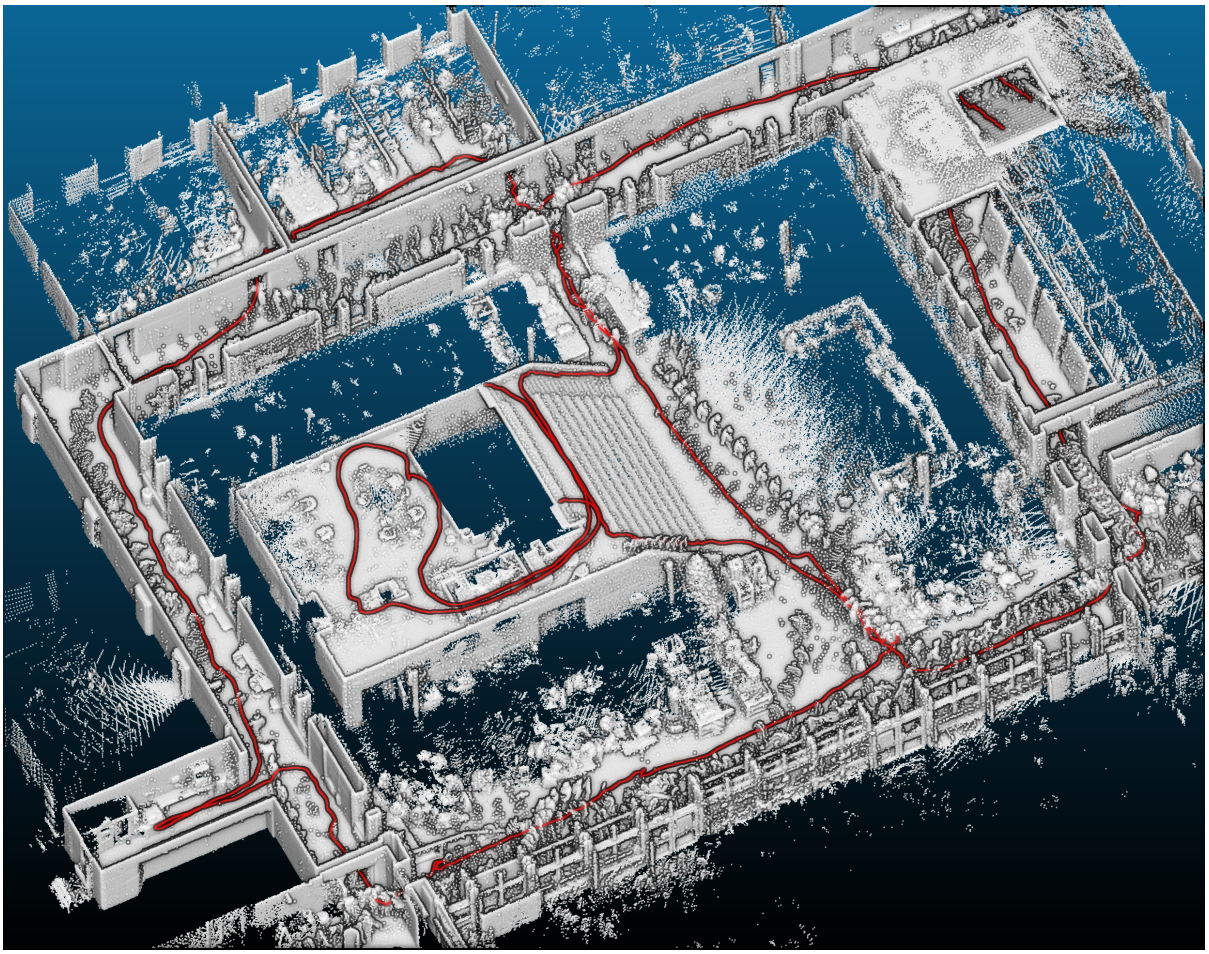
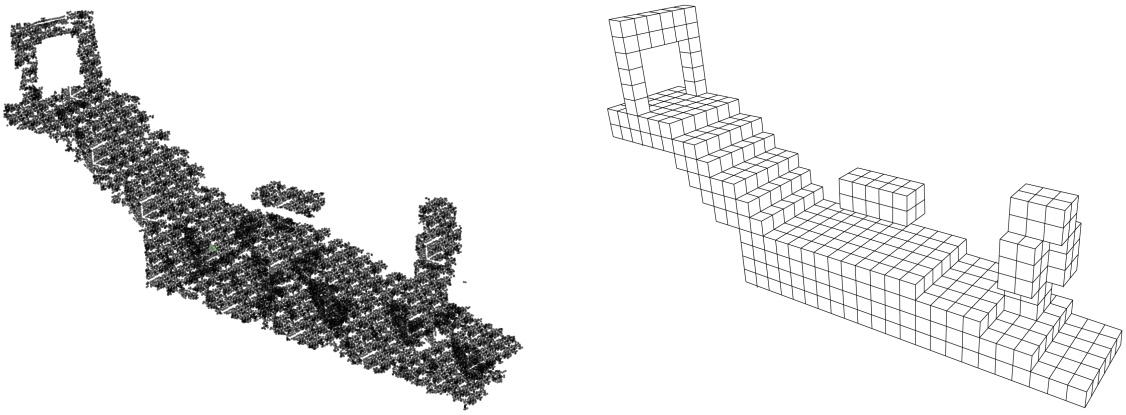


Figure 3.4: Point cloud and the corresponding trajectory

3.2.2 Voxelization

As discussed in § 2.2.1, the voxelization of a point cloud creates a spatial structure and reduces the amount of data points which increases the processing speed of the end model (Vo et al., 2015). After the voxelization process, the point cloud is transformed into a voxel space, see figure 3.5b

The voxelization is implemented by using the first part of the octree method introduced by Broersen et al. (2016). By generating an octree, a cube or voxel is subdivided into equal smaller cubes. For each subdivision level the point cloud is represented with voxels of different sizes. In this research, the smallest subdivision, also known as the octree leaves, are saved and from now on referred to as the voxel model. The introduced octree method automatically translates the point cloud to the quadrant where the x, y and z values are positive. The number of voxels along an axis is represented by integers between 0 and



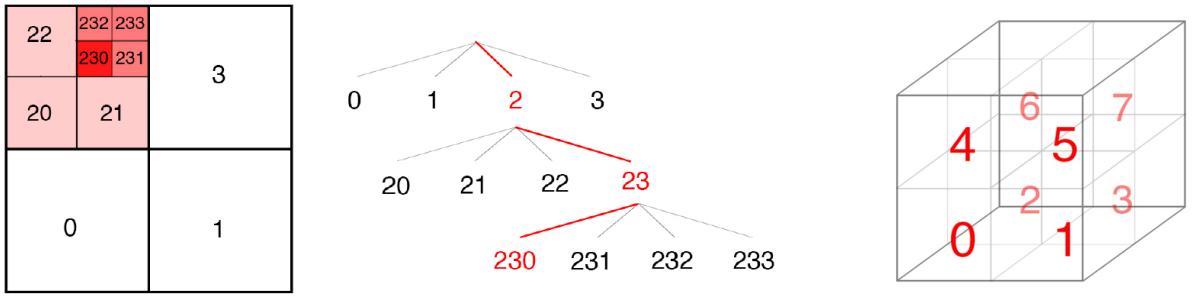
(a) A point cloud

(b) A voxelized point cloud

Figure 3.5: The voxelization of a point cloud

$2^n max$. The nmax parameter is represented by the octree depth which is a value describing the desired number of subdivisions. The point cloud is automatically scaled so that the highest axis of the point cloud bounding box is equal to $2^n max$. This method is scalable to any size of a point cloud with an almost linear increase of processing time (Broersen et al., 2016).

An octree structure is based on the equal subdivision of a space in eight quadrants. For explanation purposes, a quadtree is used which divides a square in four pieces. The same principle applies to an octree structure except that it is a 3D structure which divides the cube into 8 equal pieces. Therefore, the enumeration is runs from 0 to 7 instead of 0 to 3, see figure 3.6. For each point in the point cloud the corresponding leaf voxel is calculated. This is done for each level in the quadtree; from root node to leaf node. The corresponding number of the quadrant is added to the *locationalcode* for each level. When the leaf node is reached, depending on the amount of subdivisions, the final *locationalcode* is found. The point with a *locationalcode* of 230 follows the path from 2 to 3 and then to 0 as illustrated in figure 3.6a (Broersen et al., 2016).



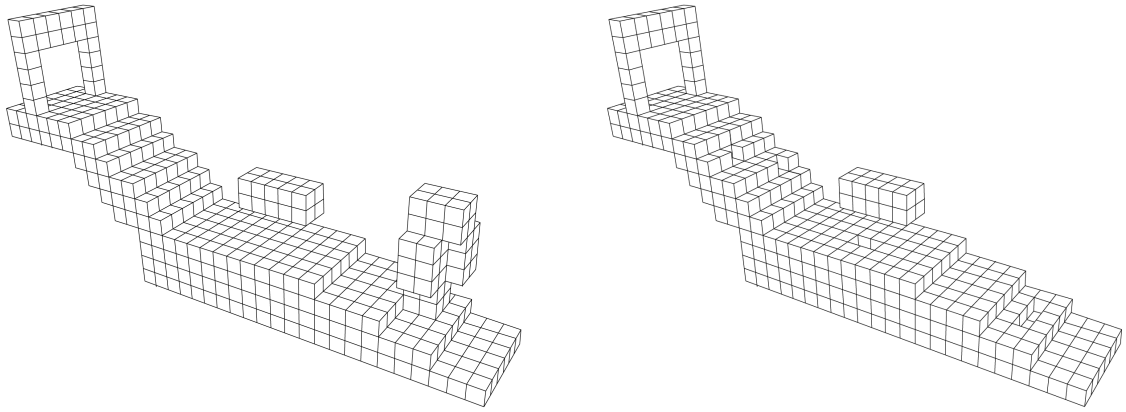
(a) Numbering of a quadtree structure

(b) Numbering of an octree structure

Figure 3.6: A quadtree and an octree structure (Broersen et al., 2016)

3.2.3 The removal of dynamic objects

If a building is scanned during opening 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 elements like furniture, walls or floors and therefore need to be identified and removed, see figure 3.7b.



(a) A voxelized point cloud (b) A cleaned voxel model by dynamic object detection

Figure 3.7: The detection and removal of dynamic objects in a voxel model

In this thesis, the detection of dynamic elements is tested in five different ways:

- Delete voxels containing n-amount of points:
  The detection of dynamic elements is based on the amount of points in a voxel. If a voxel contains less points than a predefined threshold, the voxel is removed as introduced by Fichtner (2016).

- Different time frames:
  The detection of dynamic elements is based on the comparison of two different scan frames which are compared to each other. If a voxel only exists in one scan frame, the voxel is removed.

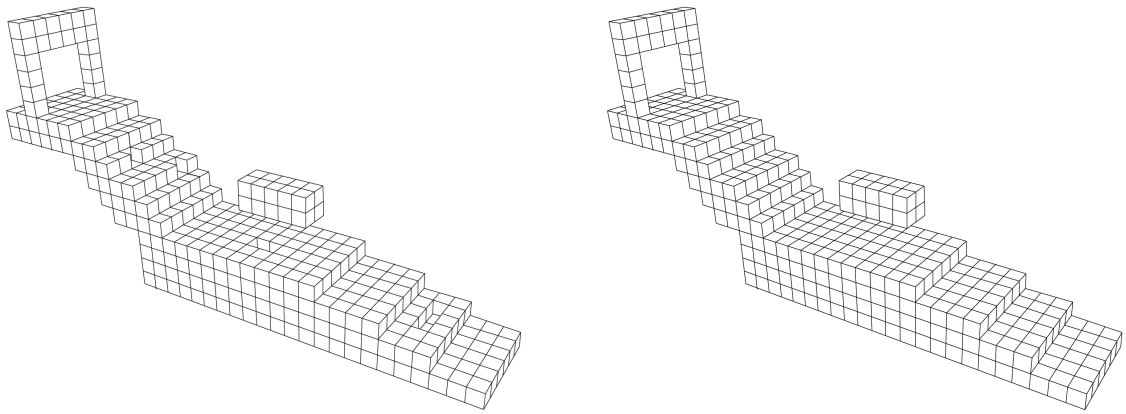
- Unique time stamps:
  The detection of dynamic elements is based on the number of unique scanning seconds per voxel. If there are less unique scanning seconds than a specific treshold, the voxel is removed.

- Floor voxels with voxels above:
  The detection of dynamic elements is based on the number of scanning seconds of the floor voxel compared to the voxels above. If the number of scanning seconds is different, the voxels above the floor voxel are removed.

- Count the voxels above a floor voxel:
  The detection of dynamic elements is based on the number of voxels above a floor voxel. By counting the amount of voxels above and assigning a weight to the floor voxels, a map which consists of mainly one colour is created. By analysing this map pattern, dynamic objects can be detected and removed.

3.2.4 Filling gaps

scan lijnen: iemand voor, occluded by objects, model,cleaning, Due to the occlusion caused by dynamic or static objects during the data capture, the final voxel model, the input point cloud or the cleaning described in § 3.2.3, some voxels are removed. This results in small gaps in the voxel model. To close these gaps, the distance between the voxels in the horizontal plane for the x, y direction is calculated. If the distance between two voxels is smaller than a certain threshold, the gaps are filled in with new voxels, see figure 3.8.

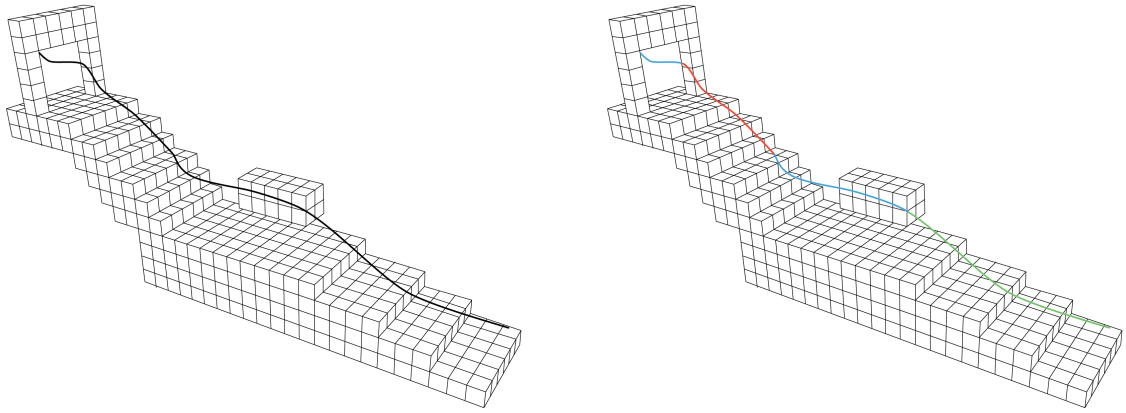


(a) A voxel model with its corresponding trajectory (b) The classification of the trajectory points

Figure 3.8: Trajectory classification

3.2.5 Trajectory classification

The trajectory classification forms the basis for the detection of distinct surfaces. The trajectory is classified into three types: stairs, flat and sloped surfaces, as introduced in 1.5. In order to identify these different types, the angles between successive trajectory points are analysed. Six parameters are needed to identify these three types: a *minimal raise*, a *maximal raise* and a *number of connected elements* for both the stair and the sloped surface.



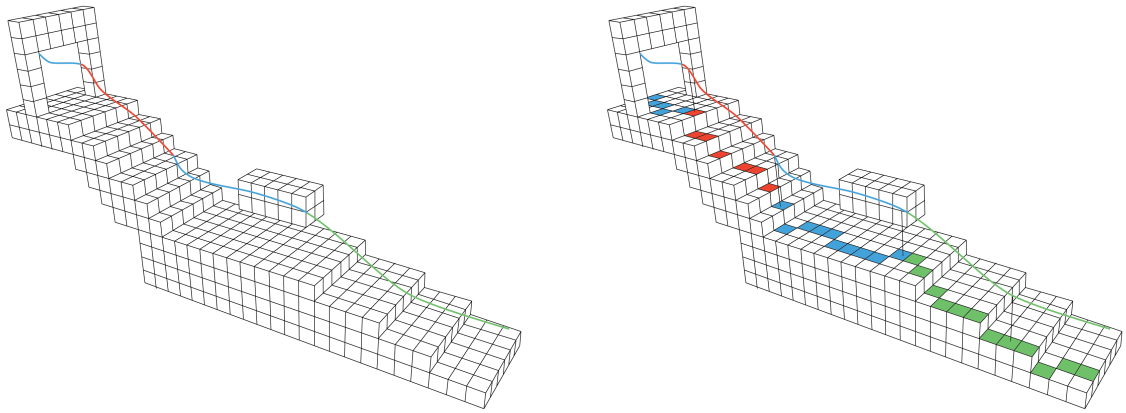
(a) Voxelization of the point cloud

(b) Trajectory classification

Figure 3.9: Quadtree and octree structure (Broersen et al., 2016)

3.2.6 Voxelmodel + seed voxels

The identification of seed voxels from the classified trajectory to the voxel model is important because these seed voxels form the basis for the region growing process. In order to identify these seed voxels, the trajectory is voxelized in the same spatial structure as the voxelized point cloud using the process described in 3.2.2. The type of trajectory voxel depends on the classified points inside the voxel, see § 3.2.5. If two or more types of points are present in one voxel, the voxel type is set to a combination class. The seed voxels are identified by finding the corresponding voxels for each trajectory voxel which have the same x, y position, see figure 3.10b.



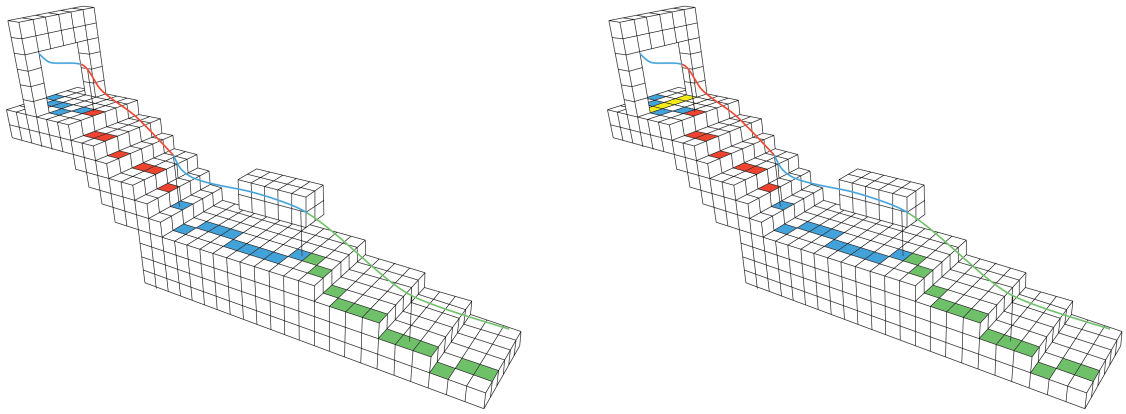
(a) A voxel model with a classified trajectory

(b) Identified seed voxels from a classified trajectory

Figure 3.10: The identification of seed voxels from a classified trajectory

3.2.7 Voxelmodel + seed voxels + identified entryways

The destination of an indoor navigation activity is most of the time located inside a specific space inside the building. Therefore, it is important to distinguish the different spaces in the final output data model. Entryways form the connecting elements between different spaces. When entryways are identified, these spaces can be split during the region growing process and form separate spaces, see figure 3.11.



(a) Identified seed voxels from a classified trajectory

(b) Identified door voxels

Figure 3.11: Identification of door voxels

Detecting doors can be implemented in two different methods. The first method detects entryways in the z-axis as described by Nikoohemat (2016) and the second method detects entryways by their x, y plane. Both methods are based on the trajectory of the MLS. If a space is not accessed through all of its entryways, they cannot be detected. If both of these classification methods are positive, an entryway is found. If only one of these methods is positive, a possible entryway is identified.

The detection of entryways is based on the height difference along the z-axis as introduces in the method by Nikoohemat (2016). In this method an entryway is detected if the following rules are met:
1. An entryway centre is an empty space
2. Above the entryway centre there are points present
3. There should be a trajectory close by

If these rules are applied along the trajectory of the MLS, a possible implementation of this method is based on the continuing calculation of the distance between the seed voxels and the ceiling voxels, see figure 3.12. Before an entryway is passed, the distance has a value of A. During the passing of the entryway it has a value of B and after the passing a value C. If this distinct pattern is recognized, an entryway is detected.

The method only works if the height differences between the seed voxels and the ceiling are big enough. Furthermore, if a lamp is mounted on the ceiling, the pattern, which is used to detect entryways, can look the same which results in the detection of a non-existing entryway. Therefore, the second door identification method is implemented. The second approach is based on the x, y plane. As can be seen in figure 3.13, the x, y plane of an entryway has a very characteristic shape. The distance to the left and right is first big, then small and then big again, see situation one. This characteristic pattern can also be recognized. There are four cases that can occur in the identification of entryways in the x, y plane:
1. there is no wall or furniture perpendicular to the entryway on the other side, see case 1
2. there is a wall or furniture on the other side perpendicular to the entryway on the right side, see case

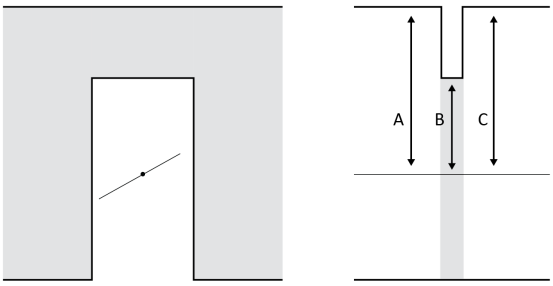


Figure 3.12: Identification of an entryway in the z-axis

2
3. there is a wall or furniture on the other side perpendicular to the entryway on the left side, see case 3
4. there is a wall or furniture on the other side perpendicular to the entryway on both sides, see case 4 After the implementation of both methods, the entryways are identified and possible entryways are found.

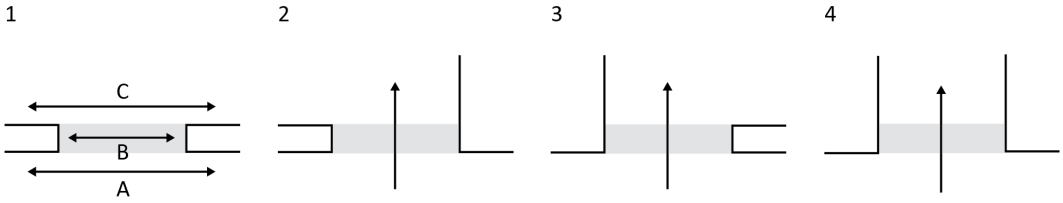


Figure 3.13: Identification of an entryway in the x, y plane

After the detection of the location of the entryways, the floor voxels that form the entryway need to be detected, see 3.14. This is necessary because otherwise the region growing process that is implemented in the following step does not stop when an entryway is reached which makes it impossible to split multiple spaces into single spaces.

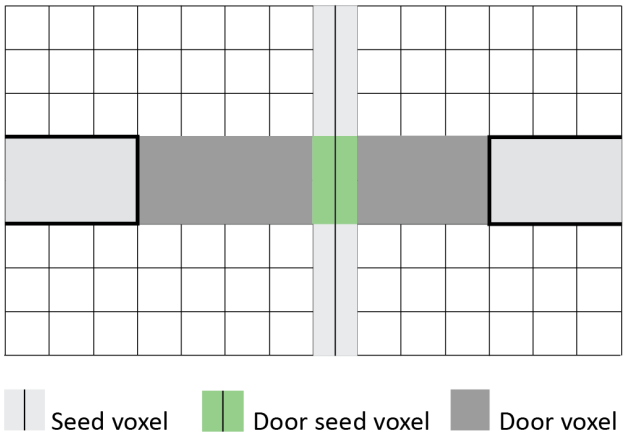


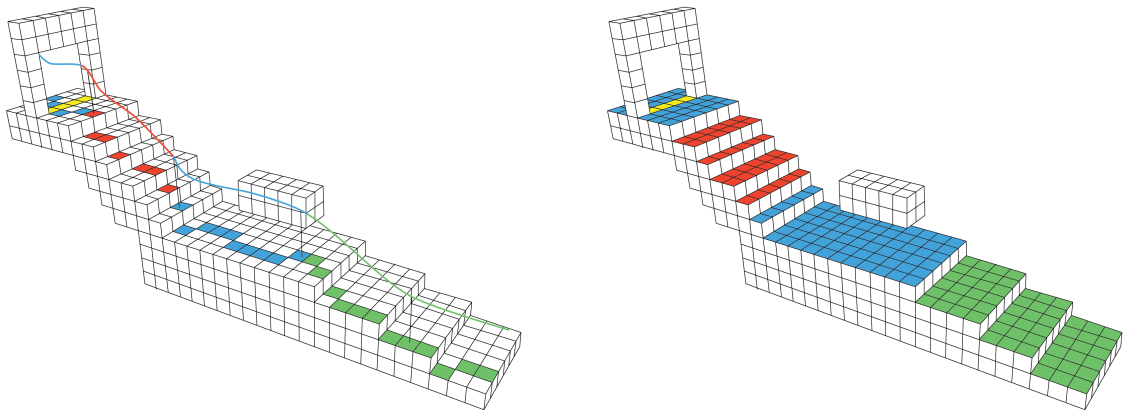
Figure 3.14: Identification of door voxels in an entryway

3.2.8 Region growing seed voxels per room

The region growing process is implemented to detect subspaces inside spaces, see figure 3.15. There are two types of region growing processes: top-down and bottom-up. The bottom-up method uses seed points to form regions with the same attributes. The top-down methods tries to find large groups of points with the same type of attributes (Rabbani et al., 2006). In this thesis, the bottom-up method is used based on the classified seed voxels identified in section 3.2.6.
There are two stopping criteria for the region growing process. The first consists of passing a door, see figure 3.14, and the second consists of a specific number of existing voxels above the current checked voxel. This last criteria shows the existence of something on the floor (which is above the seed voxel) prohibiting the continuation of the growing process of the region in that direction.
The region growing process is implemented in two different ways:

1. The first approach, the *ordered checking*, is based on the region growing process described in Rovers et al. (2015). This approach is implemented by identifying the four neighbour voxels of a seed voxel. If these voxels pass the two stopping criteria, they are added to the region.

2. The second approach is implemented in the PostgresSQL database using the ST_ClusterDBSCAN algorithm (PostgresSQL, 2017).



(a) Identified seed and door voxels

(b) Region growing based on seed voxels

Figure 3.15: Region growing from the identified seed voxels

3.2.9 Classification check

Sometimes furniture objects are marked as floors. This is caused by passing the MLS above furniture objects during the data capture. If the trajectory voxels are projected down on the voxelized point cloud, these furniture voxels are marked as seed voxels and region grown. To find out if these elements identified as regions are actually furniture objects, the change in height of the classified seed voxels is analysed. If the MLS is held above furniture objects, the height of the seed voxels changes rapidly which appears as large peaks in figure 3.16. These changes in height are bigger than the ramp and stair parameters introduced in § 3.2.5. Based on these parameters, the peaks are detected and identified as furniture and the regions are removed from the final model.

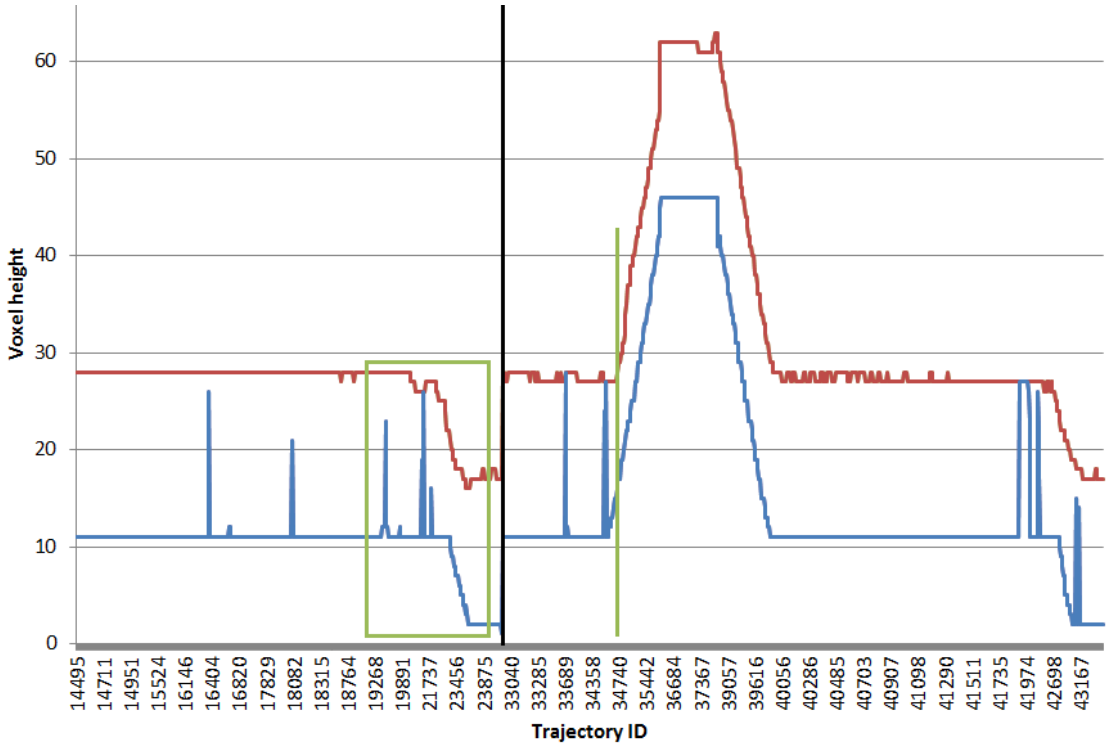
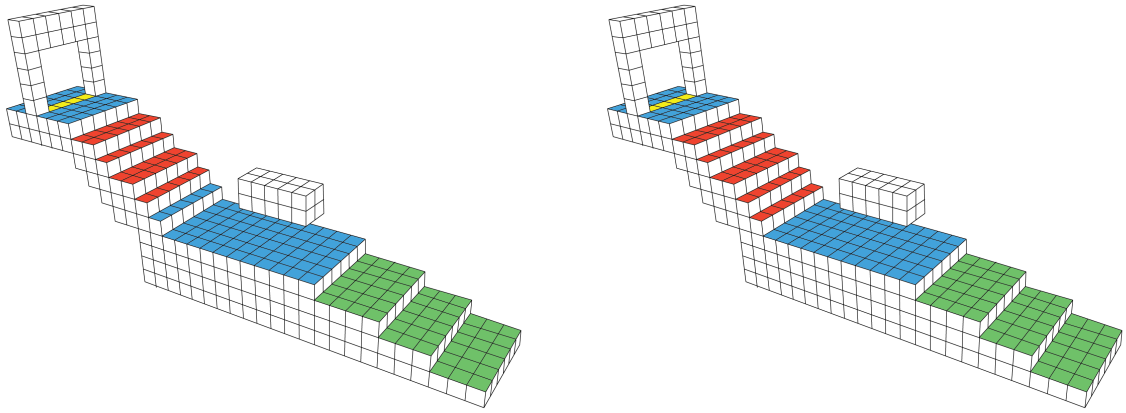


Figure 3.16: Difference between the trajectory and the seedvoxels in the z axis.
Trajectory (red), seedvoxels (blue) and marked areas (green)

It is not possible to define the exact start and end point of a stair based on the trajectory itself, as can be seen in the green box in figure 3.16. The change at the beginning and the end is caused by the MLS scanner which steadiness depends on the data capturer and at which position the MLS is held. Furthermore, when the data capturer is going up the stairs, the trajectory of the MLS is not rising at the beginning of the staircase. In most cases, the scanner is held in front of the data capturer. When the data capturer is going up the stairs, the MLS is already above the second riser before the height of the MLS trajectory changes.
Therefore, the bottom risers can never be found based on merely the trajectory but are present in the voxel model which makes them detectable. When a seed voxel compared to the previous seed voxel along the trajectory has a different height and the following seed voxels are marked as stairs (or slopes), the seed voxel could represent a riser (or part of a slope). By checking the height changes this way, wrongly classified regions can be identified correctly, see figure 3.17.

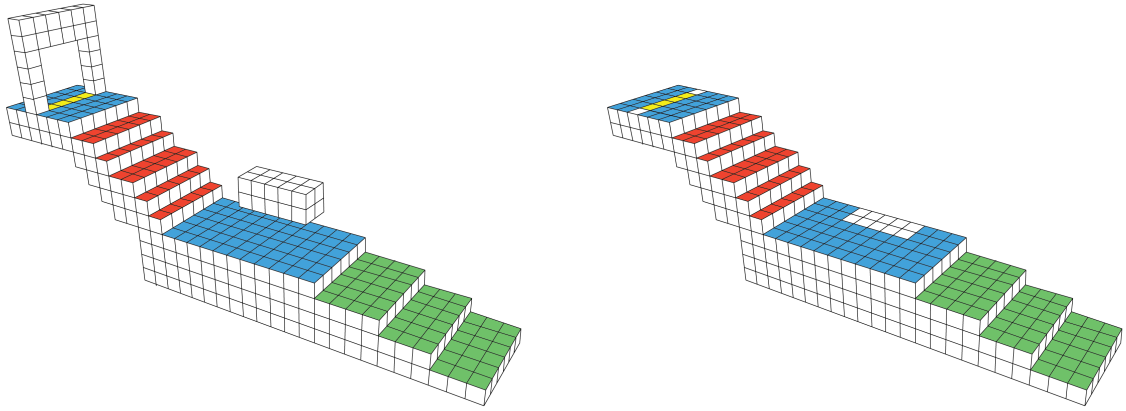


(a) Wrongly classified regions (b) Classification check of the region types

Figure 3.17: Classification check

3.2.10 Subtraction of furniture

The MLS scans the whole interior of a space. Therefore, there are also points present under furniture objects, construction elements and other indoor objects which are added to the floor regions. To derive the navigable space per actor, the space below these objects needs to be marked as a non-navigable voxel space, see figure 3.18. After the detection and removal of the dynamic voxels described in 3.2.3, the resulting voxels above the floor, stair and slope regions are assumed to be furniture objects and are therefore removed. The subspaces are checked for furniture objects up to and including the height of the actor, as described in § 3.1.

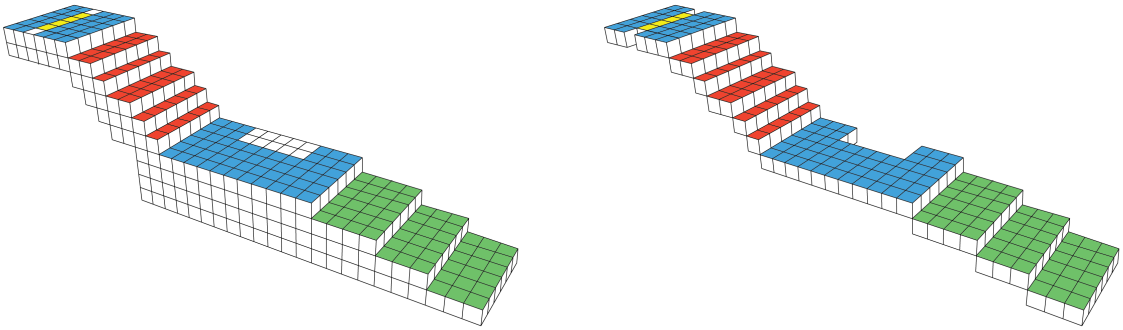


(a) Different classified subspaces (b) Navigable subspaces per actor

Figure 3.18: The identification of navigable subspace per actor by furniture subtraction

3.2.11 Final navigation voxel space

After the subtraction of the furniture, the final navigable voxel space per actor is derived.

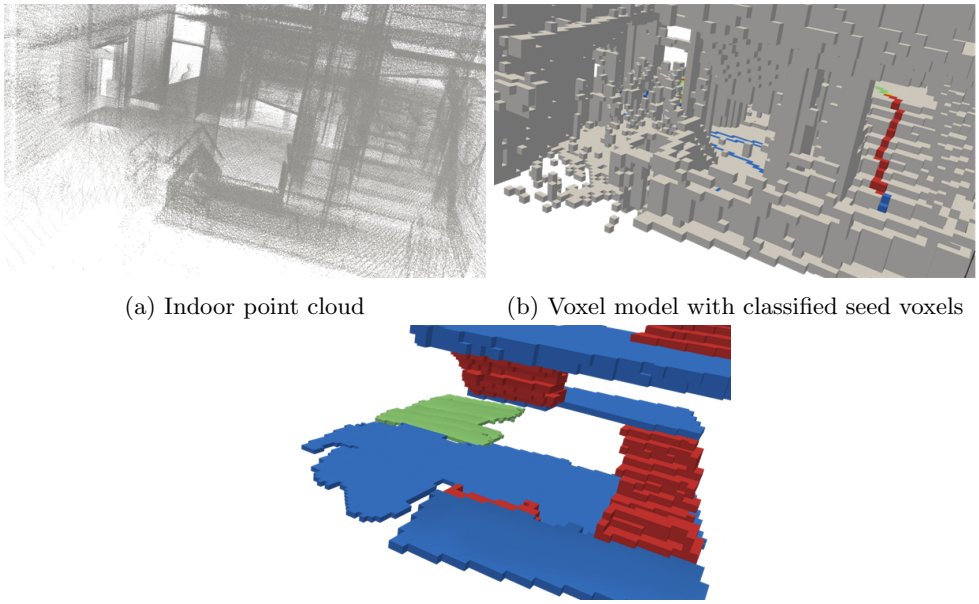


(a) Navigable subspace per actor

(b) Navigable voxel subspace per actor

Figure 3.19: Quadtree and octree structure (Broersen et al., 2016)

The entire proposed method can be distinguished into three separate phases. The first phase exists of the data capture of the indoor environment into a point cloud and a trajectory figure 3.20a. The second phase exists of several data processing steps described in this methodology. The third and last phase consist of the generation of the final product which is the total navigable voxel space per actor. These different steps are illustrated in figure 3.20c.



(a) Indoor point cloud

(b) Voxel model with classified seed voxels

(c) Final navigable voxel space

Figure 3.20: Three stages of the proposed method

4 Implementation and results

In this chapter, the implementation and the results of the implemented method are discussed. This is done by describing the results and implementation step by step, as introduced in § 3.2. After the description of the individual results, the representativeness and the processing time of the method are discussed.

4.1 General input

The method is implemented in Python 2.7 and PostgreSQL and is coded in different modules, see figure 4.1. The modules are executed from a main file which is implemented in Python. However, first, five parameters need to be set in the main file:
1. The database name
2. The database user
3. The database password
4. The name and the location of the .las file
5. The name and the location of the trajectory.txt file

To be able to connect to the local database three parameters are needed. Besides these three parameters the right point cloud .las and the trajectory .txt file need to be selected. The different modules are based on the proposed method discussed in chapter 3. If no comment is made about the origin of the used programming code, the code is fully implemented by the author. This method is 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).

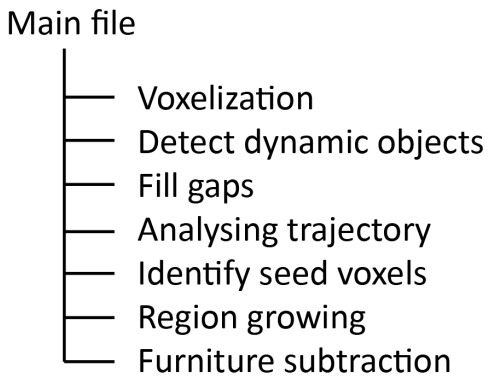


Figure 4.1: Main file structure

4.2 Voxelization

The voxelization process is implemented in Python and the PostgreSQL database. The code of Broersen et al. (2016) is used to voxelize the input point cloud which automatically saves it in the database. The calculation of the smallest leaf size and the scale factor introduced by Fichtner (2016) are also used. These scripts were adjusted and extended to be usable in this research. First, the scale factor, smallest voxel size, point cloud file name, number of voxels along the axis and the three translation values are saved in the *projectdata* table. Second, the corresponding left bottom x,y,z coordinate of each voxel is saved in the *smallestpointleaves* table together with the locational code of the voxel. For each point, a voxel record is created which results in a lot of duplicate records. The amount of duplicate records per voxel are counted and added, together with the data of the *smallestpointleaves* to the *smallestleaves* table. In this table only one record per voxel is present.

As discussed in § 3.2.2, the method automatically translates and scales the point cloud. If the octree depth increases by one, the octree structure splits one more time. This results in twice the amount of voxels along a axis, a scaling factor which is twice as large and a voxel size which is twice as small. If the same point cloud is voxelized with different octree depths, the different output models are scaled which result in figure 4.2.

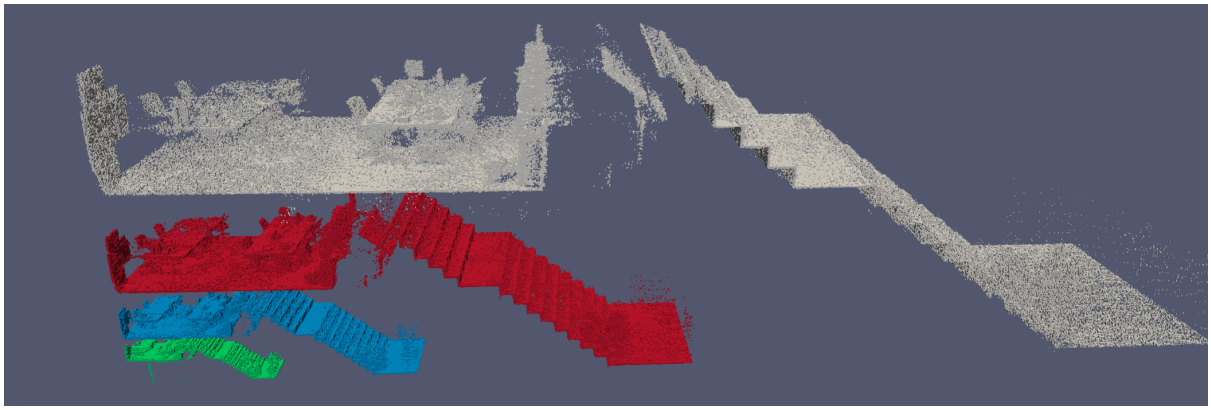
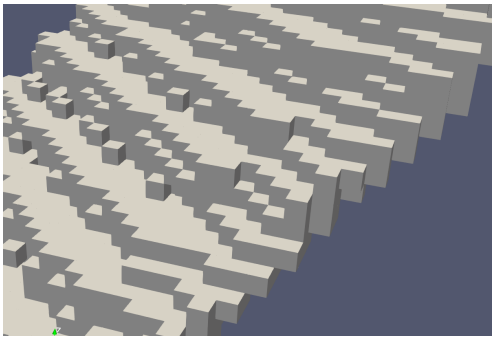


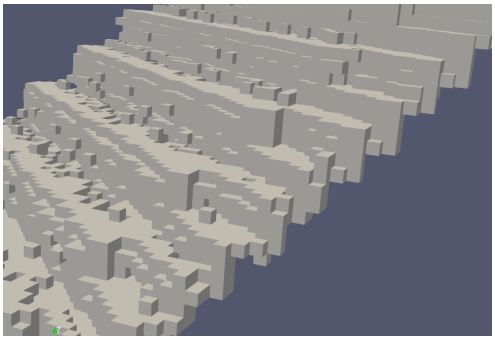
Figure 4.2: Scaling of the voxel model with different depth parameters. A depth and voxelsize of: 10 - 1.8 cm (white), 9 - 3.65 cm (red), 8 - 7.3 cm (blue) and 7 - 14.6 cm (green)

The voxel size has a large influence on the representation of the original point cloud. If a threshold below an entryway needs to be distinguished from the rest of the floor, the voxel size should be very small. In this thesis, an indoor space contains different kinds of navigable surfaces like stairs, flat and sloped surfaces as described in § 3.1. A riser of a stair forms the smallest connected surface that needs to be detected. Therefore, the detection of this element determines the voxel size. According to the ISO 21542 (2011) standard, the riser of a stair should not be higher than 15 cm. Since the risers should always be represented, a voxel size around 7 cm should probably be used. This assumption is investigated in the next part of this section.
figure 4.3a until figure 4.3d, illustrates the voxelization process of the point cloud with different octree depth values. An octree depth of 7, which is a voxel size of 14.6 cm, represents the height of a riser as one voxel. If there is noise present before a riser, a voxel is added and the riser gets deformed directly. This impacts the representation of this riser, see figure 4.3a. Therefore, the octree depth of 7 is not suitable to detect risers. An octree depth of 8, which has a voxel size of 7.3 cm, represents the height of a riser by three voxels. These three voxels are enough to detect a riser even if some noise voxels are present. The voxel models with an octree depth of 9 and 10, which have a voxel size of 3.65 and 1.83 cm, represent the point cloud of the interior of a building more accurate but also increases the amount of voxels. This

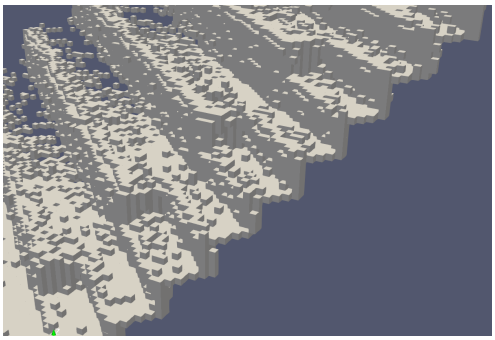
larger amount of voxels increases the processing time. As also can be seen in the images, a smaller voxel size creates gaps at the parts which are not scanned thorough enough. As earlier assumed, a voxel size of around 7 cm is sufficient to detect individual risers of a stair and is not introducing large gaps in the voxel representation.



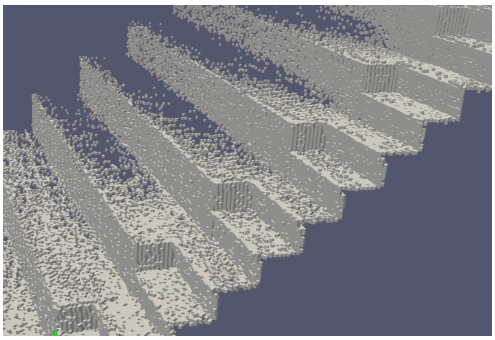
(a) Voxelmodel: depth 7, voxel size 14.60 cm and 1 voxels/riser



(b) Voxelmodel: depth 8, voxel size 7.30 cm and 3 voxels/riser



(c) Voxelmodel: depth 9, voxel size 3.65 cm and 6 voxels/riser



(d) Voxelmodel: depth 10, voxel size 1.83 cm and 10 voxels/riser

Figure 4.3: Different voxel depths and voxel sizes

The representativeness of a voxel model compared to the original point cloud depends on the voxel size. As can be seen in figure 4.4 the green voxels are representing the red points of the point cloud more accurate than the gray voxels. However, increasing the amount of voxels also increase the processing time. Therefore, a balance between the voxel size and the amount of voxels is important.

It should be noted that in the current implementation, the voxel size depends on the number of subdivisions of the octree and the bounding box of the point cloud as described in section 3.2.2. This voxel size is automatically calculated and results in a different voxel size dependant on the extensions of the input point cloud. This means that a fixed octree depth of 8 will not always result in a voxel size of 7.3 cm. Before different point clouds of the same environment can be compared, they need to be rescaled to their original sizes. It would be less time consuming to ask the operator about the preferred voxel size before processing the data.

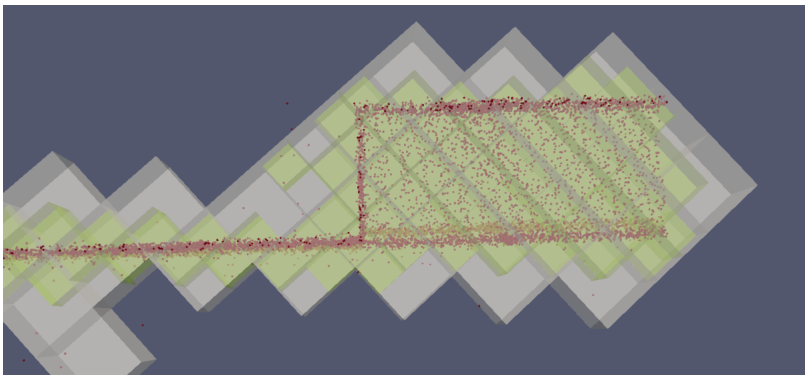


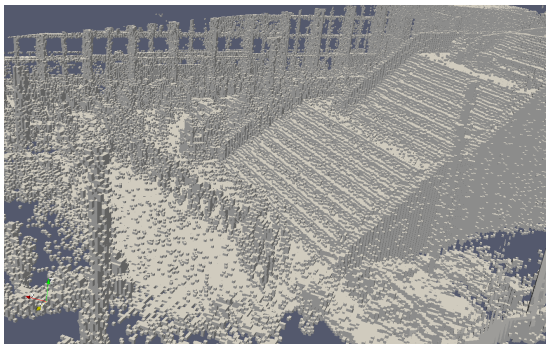
Figure 4.4: Voxel representation of the point cloud. Voxelsize of 7.3 cm (white), voxelsize of 3.65 cm (green) and the original point cloud (red)

4.3 Remove dynamic objects

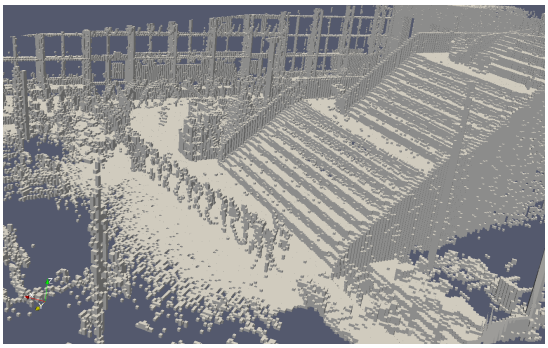
As described in § 3.2.3, many different cleaning techniques are investigated. These techniques will be discussed in this section. All cleaning methods are implemented in the PostgreSQL database. The most applicable cleaning method will be chosen and further introduced after the discussion.

4.3.1 Delete voxels containing n-amount of points

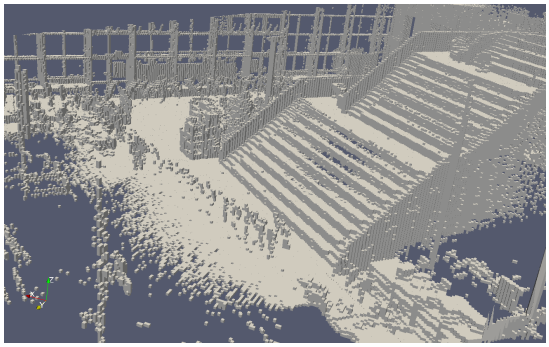
As described in 4.2, the amount of points per voxel are saved in the *smallestleaves* table. If a voxel of 5 by 5 cm contains two points, the voxel does not represent an important part of an object in the indoor environment. A possible way to clean this voxel model is to remove the voxels which contain an n-amount of points. As illustrated in figure 4.5a, most dynamic elements are partly removed when voxels with less than eight or ten points are detected.



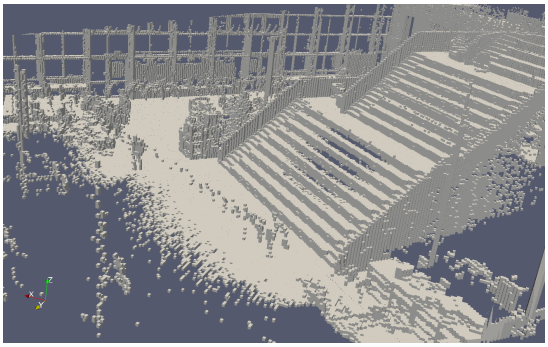
(a) No voxels removed



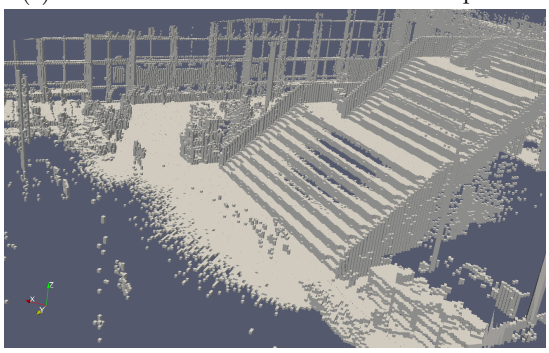
(b) Voxels which contain more than two points



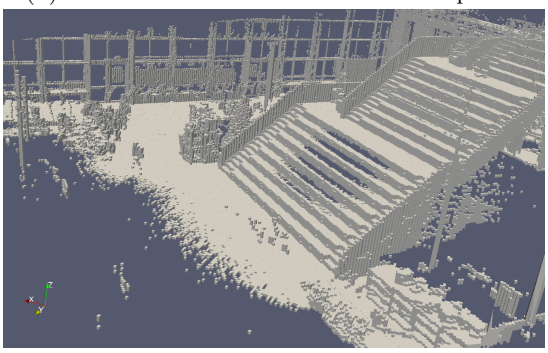
(c) Voxels which contain more than four points



(d) Voxels which contain more than six points



(e) Voxels which contain more than eight points



(f) Voxels which contain more than ten points

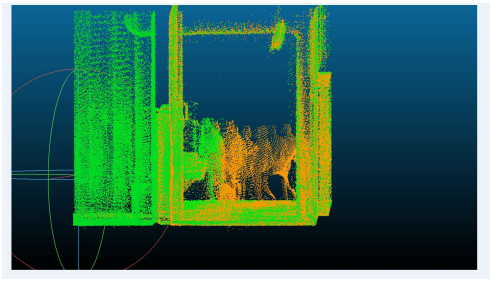
Figure 4.5: Different removements of voxels with a n-amount of points with a voxel size of 7.3 cm

Cleaning this way does not focus on the dynamic elements themselves but it becomes a general cleaning method which removes a lot of other data as well. Furthermore, it is still possible that a dynamic object which was close to the scanner during the data capture contains more than twenty scanning points. This means that a lot of data must be removed before these dynamic voxels are detected.

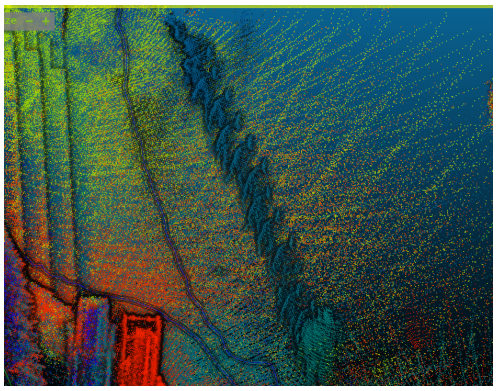
4.3.2 Different time frames

The SLAM algorithm which is used to form the final point cloud from the MLS device gives a more reliable result when the data is captured in closed loops (GeoSLAM, 2016). Because of the loop, parts of the specific rooms or hallways are scanned multiple times. As can be seen in figure 4.6a, this part of the room exists of green and orange points which means that the space is scanned twice. A possible way to identify dynamic objects is by comparing these two time frames. A dynamic object is only present at a

specific position during a specific time (Józsa, 2012). Therefore, dynamic objects can be present in one time frame but are not present in the next time frame and are therefore only represented by points of one colour. Static objects are present in both time frames and consist of both green and orange points. This is also illustrated in figure 4.6b where dynamic objects are present in the red and blue time frame whereas the static elements are present in all five time frames.



(a) Point cloud coloured by time. Scan time one (green) and scan time two (orange)



(b) Point cloud coloured by time. Multiple scanning times

Figure 4.6: Point cloud collored by time

These different time frames are detected by selecting all the points inside the voxels and sort them on their capturing time. If there is a gap of more than thirty seconds, it indicates a different time frame. By counting the different time frames per voxel, the amount of time frames can be derived. By removing all voxels that contain only one time frame, the dynamic objects are removed as illustrated in figure 4.7.

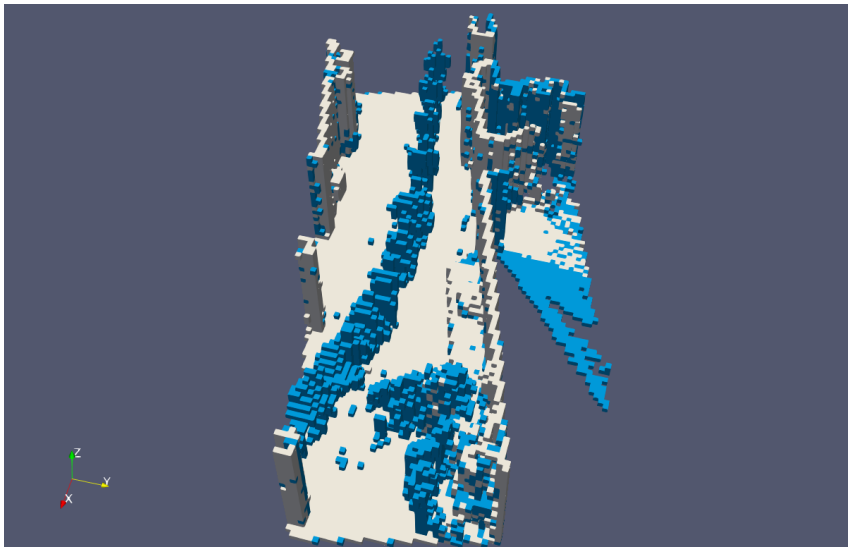


Figure 4.7: Identified dynamic objects in different time frames

To apply this method to the whole point cloud, a room needs to be scanned multiple times if dynamic objects are present during the data capture. Using different time frames to detect dynamic elements

requires that first a different space is scanned before the same space can be scanned again. Scanning the same space multiple times results in a longer scanning time which in the end leads to higher costs for the entire project.
As can be seen in the right bottom corner of figure 4.7, static elements are also noted as dynamic objects. This is caused by the dynamic objects which occludes the static objects. Therefore, the second scan needs to capture the same elements inside a space as the first scan. If a space behind a closet is scanned for the first time, this space also needs to be scanned during the second scan or the space behind the closet will be removed. Applying this method in environments with a lot of dynamic objects is difficult since both the scanned spaces and the order in which they are scanned need to be memorized. Moreover, environments with a lot of dynamic objects block many static objects which are therefore removed. This results in a model with many missing static objects.

4.3.3 Unique time stamps

A different method to detect dynamic objects is to calculate the amount of unique scanning seconds per voxel. If a dynamic object is only present at a specific location during a specific time, the voxel only contains points that are scanned during a very short period of time. The dynamic objects are identified by getting the scanning times of the points inside a voxel. By rounding the time records on seconds and sort them on their data capture time, the different scanning times can be computed. As can be seen in the figure 4.8, the static voxel A contains four scanning seconds whereas the dynamic voxel B only contains two scanning seconds and is therefore marked as a dynamic object. This value is described by the *numoftimestamps* parameter. Detecting dynamic objects this way does not require to scan every

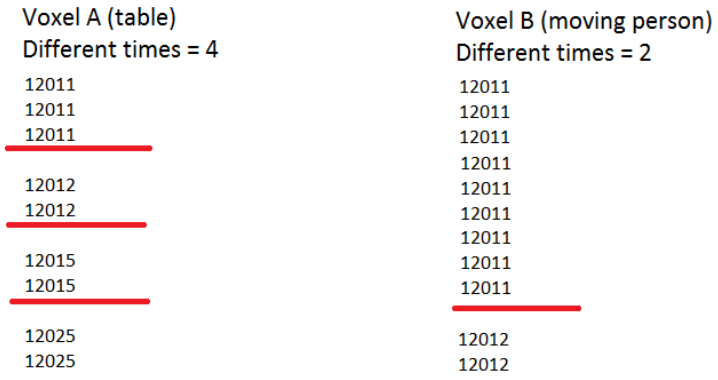


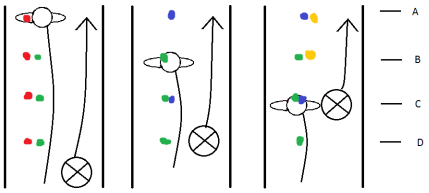
Figure 4.8: Unique time stamps of a static (A) and dynamic (B) voxel

space twice. Dynamic objects that were at a specific place during the data capture, like actors in line for a coffee machine, cannot be detected.

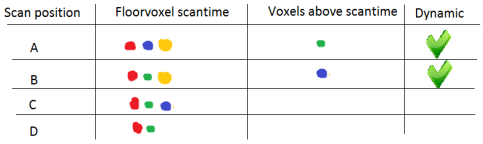
4.3.4 Floor voxels with voxels above

A different approach compares the unique scanning seconds of the floor voxels with the time of its voxels above. As can be seen in figure 4.9a, A,B,C and D are all scanned during the red data capture period. B, C, and D are all scanned during the green data capture period and A is occluded by a dynamic object. C and A are scanned during the blue data capture period, B is occluded by the dynamic object and D is behind the data capture device and is because of this not scanned. A and B are scanned during the yellow data capture time. This time, the points C and D are behind the capture device and therefore

not present during the yellow data capture time period. The dynamic object is scanned above the floor voxels at position A and B. If the data capture time of the floor voxels A,B,C and D are compared with the data capture time of its voxels above dynamic objects can be detected, see figure 4.9b.



(a) Scanning of points A,B,C and D with a MLS while a dynamic object is passing (the human shape)



(b) Comparing the scanning times of floor voxels with the voxels above

Figure 4.9: Dynamic objects

Floor voxels need to be identified by the region growing process before this approach can be implemented. After selecting the floor voxels and its voxels above, the individual time frames of the voxels can be compared. If a floor voxel contains different scanning time frames than the voxels above, a dynamic object is detected. As mentioned earlier, only dynamic objects that were moving during the data capture can be detected.
As can be seen in figure 4.10a and figure 4.10b, the floor voxels have a value of 0 and the voxels above have a value of 1 in the *basepoint* column. The voxels above this floor voxel have different z values. As can be seen in figure 4.10a, the bottom voxel contains four different scanning seconds whereas the voxels above contains only one scanning second. Therefore, the voxels above contain an object that was not scanned for a longer period of time and would probably be a dynamic object that was moving during the data capture. figure 4.10b shows the different timestamps of a dynamic element that was not moving during the data capture. As can be seen, the floor voxel contains four unique scanning seconds and the voxels above contain one or multiple scan seconds. In this case it is more difficult to decide which voxel is dynamic and therefore needs to be removed.

basepoint double precision	baseid double precision	unitime double precision	z text
0	3763	1481810022	1
0	3763	1481810023	1
0	3763	1481810024	1
0	3763	1481810025	1
0	3763	1481810026	1
1	3763	1481810023	11
1	3763	1481810023	12
1	3763	1481810023	13
1	3763	1481810023	14
1	3763	1481810023	15
1	3763	1481810023	16

(a) A dynamic object: unique rounded seconds of the floor voxels with the voxels above

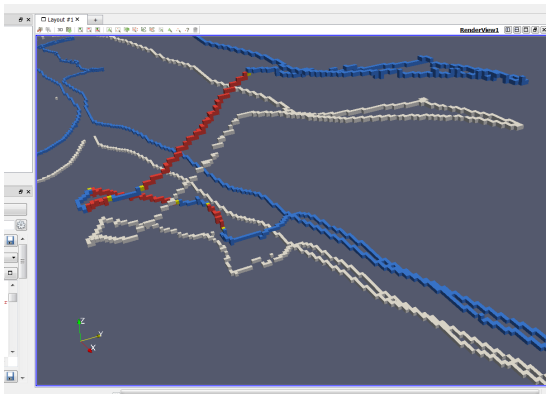
basepoint double precision	baseid double precision	unitime double precision	z integer
0	4747	1481810020	1
0	4747	1481810021	1
0	4747	1481810022	1
0	4747	1481810024	1
1	4747	1481810022	2
1	4747	1481810023	2
1	4747	1481810022	3
1	4747	1481810023	3
1	4747	1481810022	7
1	4747	1481810022	8
1	4747	1481810022	9
1	4747	1481810022	10
1	4747	1481810022	11
1	4747	1481810022	12
1	4747	1481810022	18

(b) A dynamic object that was not moving: unique rounded seconds of the floor voxels with the voxels above

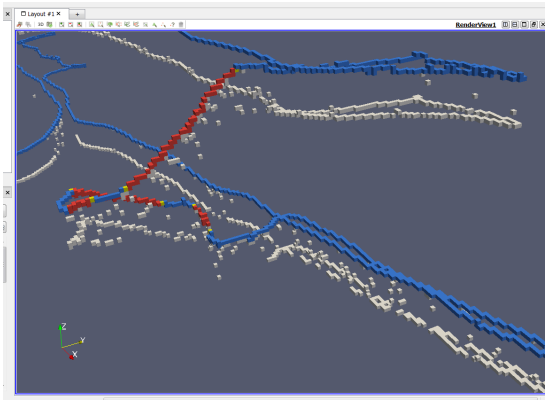
Figure 4.10: A dynamic object on the move and a dynamic object at the same position

Cleaning the point cloud from dynamic objects after the region growing process requires that all the steps before this process are applied to the uncleaned voxel model. Applying a cleaning method after the seed voxel identification and the region growing process results in a wrong identification of these seed voxels. The coloured voxels represent the trajectory voxels and the white voxels represent the seed

voxels, see figure 4.11a. Applying the cleaning method after the seed voxel identification results in wrong identifications, as can be seen in figure 4.11b. The number of wrong identifications increases when an environments has a lot of dynamic objects. Therefore, the cleaning method should take place at the beginning of the proposed MSc classification process. Moreover, it is not possible to detect dynamic elements that were not moving during the data capture.



(a) Identified seed voxels after the detection of dynamic objects

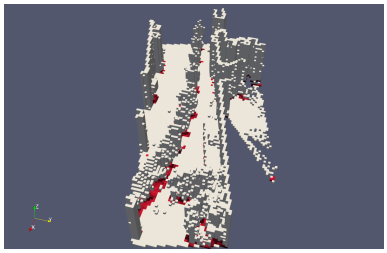


(b) Identified seed voxels before the detection of dynamic objects

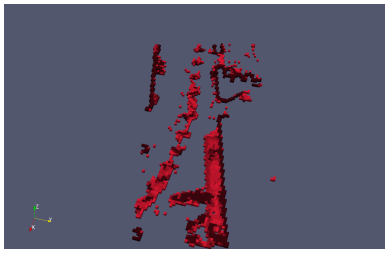
Figure 4.11: Identification of seed voxels in an cleaned and uncleaned environment

4.3.5 Count the voxels above a floor voxel

Instead of comparing the unique scanning seconds between the floor voxels and its voxels above, another way to detect dynamic objects is by counting its voxels above. This should be counted until the height of the agent. Afterwards, the floor voxels get the counted weight which is visualized in a map as illustrated in figure 4.12a and figure 4.12b. In this image, when there is no colour there are no voxels above the floor voxel and the darker the colour, the more voxels are present above the floor voxels. This map can



(a) 3D view of the voxel model voxel model



(b) Counted voxels above Floor voxels

Figure 4.12: Counting voxels above floor voxels

be used to identify a dynamic object based on training samples as illustrated in figure 4.13. Pedestrians for example have a head and legs, which results in more voxels in the middle and fewer voxels around the centre of their representation in the voxel model. As introduced by Józsa (2012), pedestrians sometimes turn into long-drawn shadows in point clouds, as can be seen in the left bottom part of figure 4.12b. In this case, detecting pedestrians by training samples is a lot harder. Furthermore, the cleaning process can only take place after the region growing process which results in seed voxel identification errors as discussed earlier.

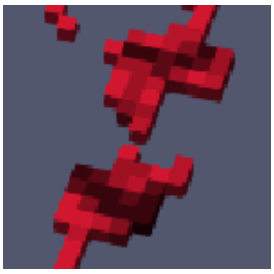


Figure 4.13: A close up of figure 4.12b where a single pedestrian can be seen

4.3.6 Implementation of the detection of dynamic objects

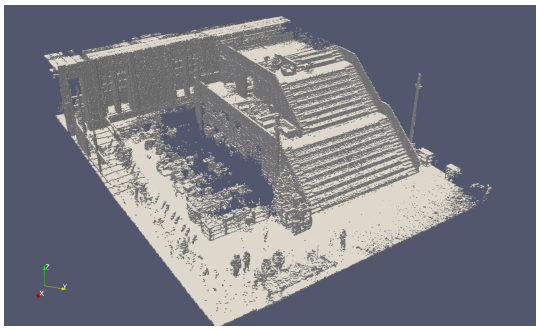
The described cleaning methods above have some limitations that can be seen in Table 4.1. Some methods are not focused on the identification of dynamic objects and some approaches can only be implemented after the region growing process. Implementing the cleaning method in a later stage leads to other complications as described above. A cleaning method that is not focussed on dynamic objects is also not useful. The *different time frames* method results in three times a 'NO' in the table. Scanning a space twice and a cleaning method which heavenly depends on occlusion is also not practical. Therefore, the *unique time stamps* method is chosen as the cleaning method for the detection of dynamic objects in the voxel model.

	Focus on dynamic objects	Scanning once	Occlusion no large impact	Detect dynamic objects standing still during data capture	Beginning of process	No problem with long drawn shadows
n-amount of voxels	NO	YES	YES	YES	YES	YES
Different time frames	YES	NO	NO	NO	YES	YES
Unique time stamps	YES	YES	YES	NO	YES	YES
Floor and voxels above	YES	YES	YES	NO	NO	YES
Count voxels above	YES	YES	YES	YES	NO	NO

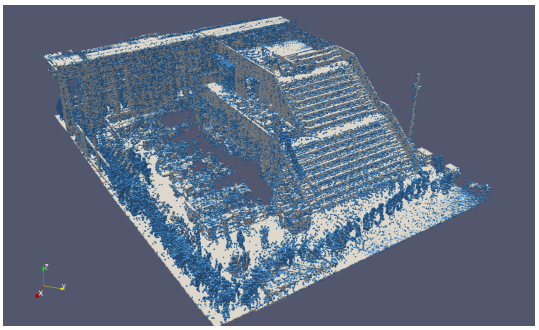
Table 4.1: Positive (YES) and negative (NO) factors of the proposed dynamic detection methods

4.3.7 Detection of dynamic objects: unique time stamps

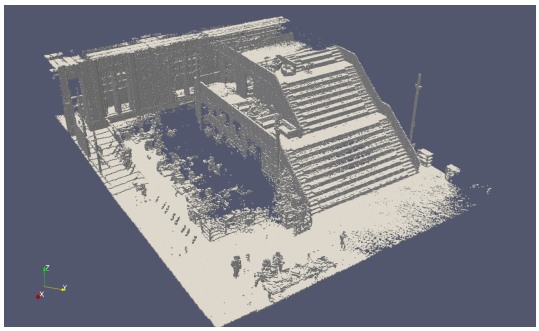
The chosen cleaning method contains only one parameter, which is called the *numoftimestamps* parameter. Because the unique scanning times of the points inside a voxel are counted, the voxel size has a large influence on the result. If the voxel size is large, the voxel contains presumably more points than if the voxel size is small, which results most likely in the voxel containing less points. Therefore, the *unique time stamps* method is tested on a voxel model with a voxel size of around 7 cm as described in § 4.2. As can be seen in figure 4.14a, removing less than two scanning seconds results in the loss of a lot of voxels. These voxels exists of unidentified voxels and dynamic objects. These unidentified voxels are most likely noise or poorly scanned objects which voxels only contain one or two points each.
If the *numoftimestamps* has a value of 3, whole dynamic objects are detected and removed. There are also voxels removed from the sides of the voxelized model. Increasing the *numoftimestamps* parameter



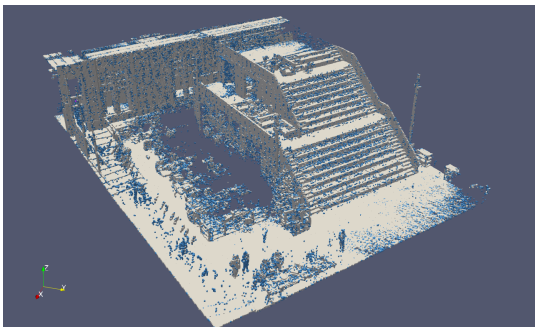
(a) Voxels witch contain two or more scanning seconds: *numoftimestamps* = 2



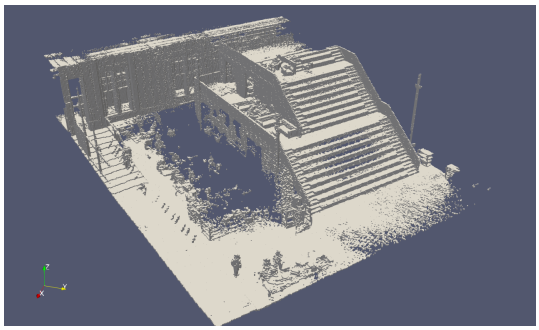
(b) Removed voxels from figure 4.14a (blue): *numoftimestamps* = 1



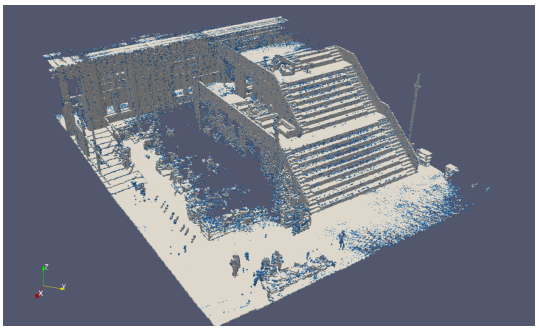
(c) Voxels witch contain three or more scanning seconds: *numoftimestamps* = 3



(d) Removed voxels from figure 4.14c (blue): *numoftimestamps* = 2



(e) Voxels witch contain four or more scanning seconds: *numoftimestamps* = 4

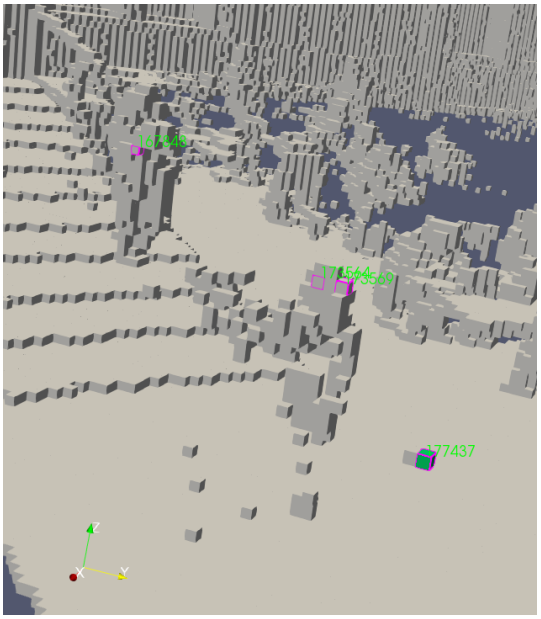


(f) Removed voxels from figure 4.14e (blue): *numoftimestamps* = 3

Figure 4.14: Removed voxels containing less than n-unique scanning seconds per voxel

results in the increase of the removal of these voxels. Moreover, it only partly detects and removes dynamic objects. Therefore, the cleaning parameter is set to a value of less than 3.
As can be seen in figure 4.14c, some dynamic elements are still present after the data cleaning process. When the unique time stamps of these elements are analysed, a lot of different scanning seconds are present. This indicates that these voxels were on the same position for a longer period of time during the data capture.
If there was a large time gap between successive seconds, dynamic objects of different scanning frames could be present at the same position as discussed in § 4.3.2. This is not the case for the investigated voxels. The numbers in figure 4.15a are represented by the *locationalcodes* in figure 4.15b. The difference between the two numbers is described in Table 4.2.
It should be noted that the detection of the dynamic objects that have remained at the same position

during the scanning cannot be distinguished at this moment. An example of such objects, as discussed before, are pedestrians who are waiting in a queue near the coffee machine. Such objects are therefore still present in the voxel model and will influence the results of the other processing steps.



(a) Voxels containing less than three unique seconds removed

	locationalcode character varying(100)	unitime bigint
1	120152550	1481806796
2	120152550	1481806797
3	120152550	1481806798
4	120152550	1481806799
5	121364341	1481806773
6	121364341	1481806786
7	121364341	1481806789
8	121364341	1481806790
9	121364341	1481806791
10	121364361	1481806786
11	121364361	1481806787
12	121364361	1481806788
13	121364361	1481806789
14	123150065	1481806772
15	123150065	1481806782
16	123150065	1481806783

(b) Timestamps of four voxels figure 4.14a

Figure 4.15: Unique scanning seconds per voxel

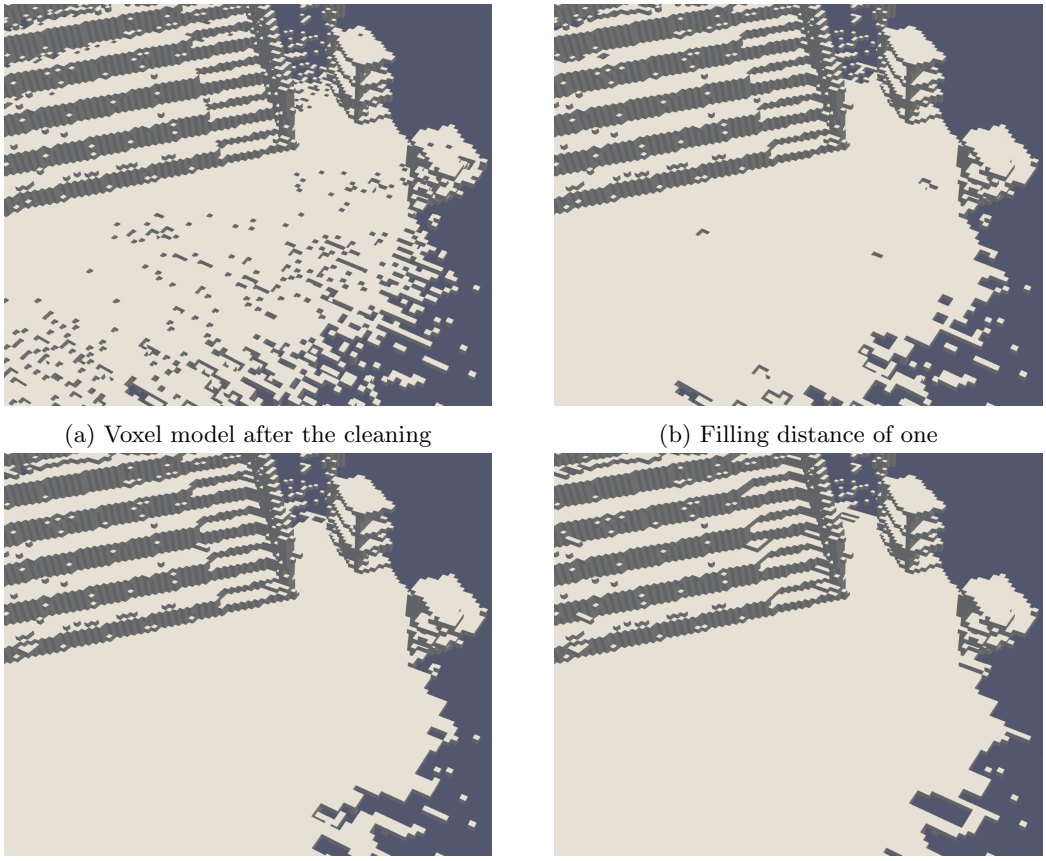
Number in figure 4.15a	locationalcode in figure 4.15b
173569	121364361
173564	121364341
167848	120152550
177437	123150065

Table 4.2: Number translation between figure 4.15a and figure 4.15b

4.4 Filling gaps

The detection of the gaps in the voxelized model is implemented in the PostgreSQL database. First, the SQL query groups all the voxels based on their height and orders these voxels based on the x- and y-axis. Second, the voxels are grouped by the x-axis. This way, the distance between the voxels on the y-axis can be checked. If the distance between two successive voxels is less than the *filgaps* parameter, the distance represents a gap and is filled with new voxels. After the identification of the position of the new voxels, the *locationalcode* is calculated in the Python environment and the data is added to the *smallestleaves* table. This process is repeated for the x-axis.
By using a *fillgaps* parameter value of 1, most gaps are filled as can be seen in figure 4.16b. If the sides of the surface are very close to each other, like risers of a stair to the boarding of a stair, the gaps between these elements are also filled with voxels. This effect increases if the *fillgaps* parameter increases which results in a strange representation of elements in the voxel model, see figure 4.16a until figure 4.16d.

Therefore, the *fillgaps* parameter will get a value of 1 which results in the addition of 4.5% of the total voxels.
A possible solution to this problem is to cross-check the added voxels with the original voxelized point cloud. If a match is found, the voxel is restored. If there is no match found during the cross-check, the voxel represents new data and can cause problems as described above.



(a) Voxel model after the cleaning

(b) Filling distance of one

(c) Filling distance of two

(d) Filling distance of three

Figure 4.16: Different *filling gaps* parameters

4.5 Trajectory classification

The detection of the different types of trajectory points is implemented in Python and described by the pseudo code in B. The process starts by reading the points from the .txt file. A possible stair or slope point is saved as a possible surface point if there is a change in height of 10 cm, see figure 4.17.

The following step consists of testing the possible trajectory points regarding the stair thresholds. For each possible trajectory point and the next point, the distance is calculated. The angle is derived based on the calculated distance and the height of 10 cm and is compared with the stair thresholds, see Table 4.3 testing set A. If the point falls within the thresholds, it is added to the *possible stair list*. A point is only classified as a stair if there are 4 successive stair points in a row. More stair points are added until the angle thresholds are exceeded. The points between the stair points are classified as stairs and removed from the *possible surface points* and the *total points* lists. The process is repeated to classify slope points

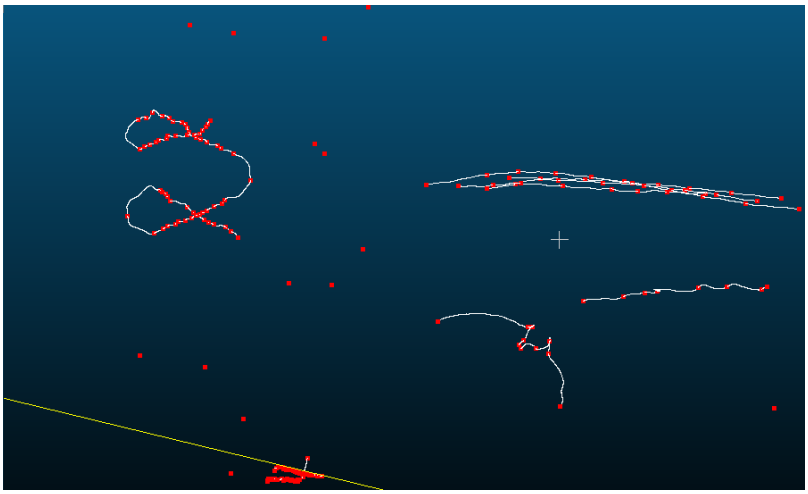


Figure 4.17: Possible stair/slope points: detected if the trajectory changes more than 10 cm

with the slope thresholds. The points remaining after the two classification rounds are classified as flat surfaces. A classification result is illustrated in figure 4.18.

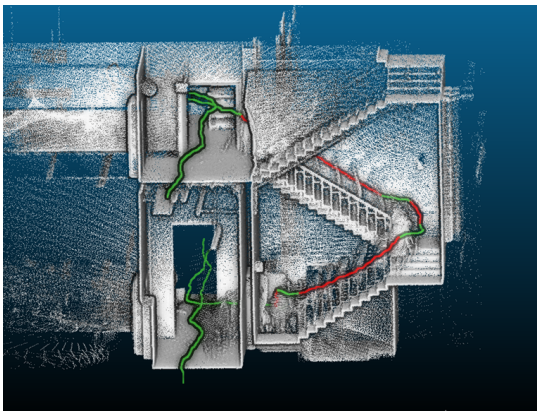


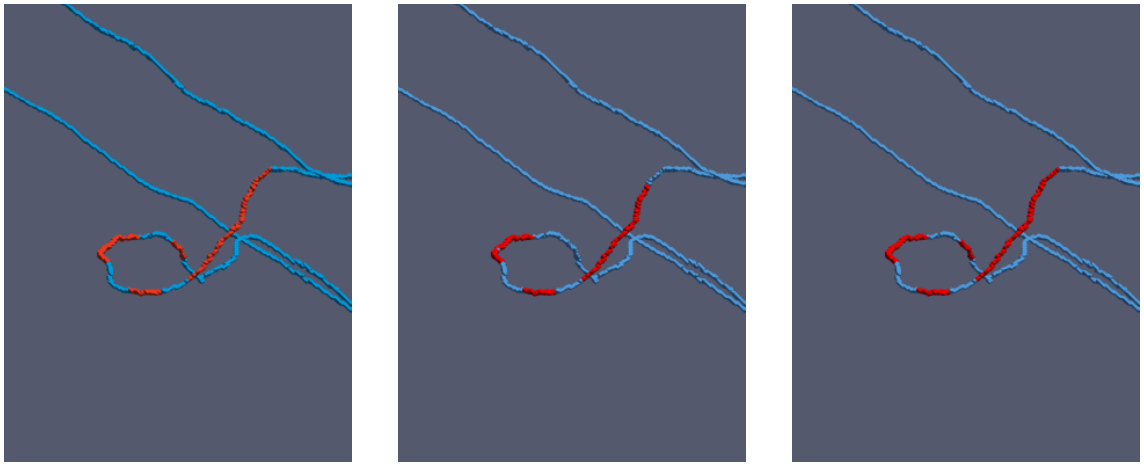
Figure 4.18: Classified trajectory: horizontal trajectory (green) and stair trajectory (red)

Defining the different threshold values are based on test cases. For explanation purposes three different thresholds sets are visualised and analysed. The first set consists of the thresholds which are used during the process. The second set exists of a lower maximum angle value and the third set exists of a higher minimum angle value, see Table 4.3. The whole classified trajectory for each set is visible in D.

Testing Set	Name	Slope Minimum angle in degree	Maximum angle in degree	connected elements	Stair Minimum angle in degree	Maximum angle in degree	connected elements
a	Used parameters	2.3	18.4	2	7.1a	90	4
b	Lower maximum	2.3	11.3	2	7.1a	45	4
c	Higher minimum	3.8	18.4	2	14	90	4

Table 4.3: Different sets of trajectory analysis parameters

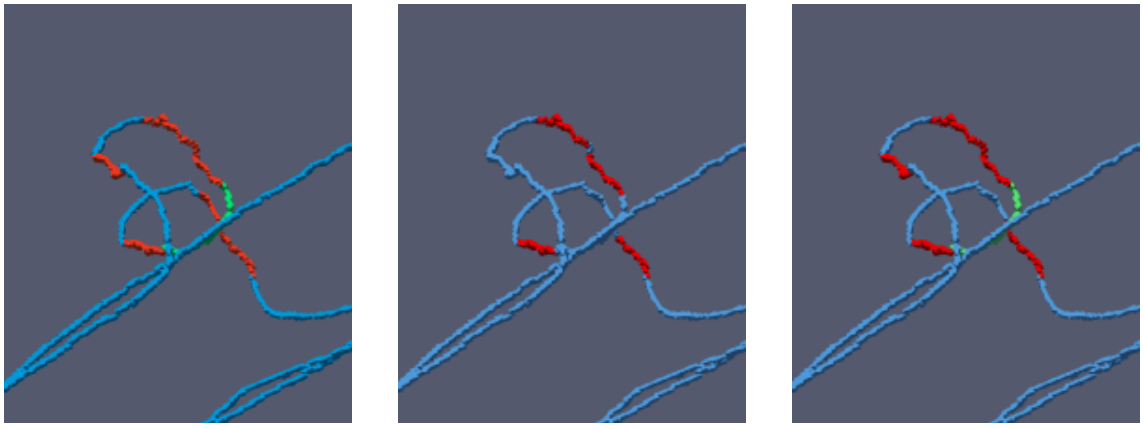
The environment where the data is captured contains stairs with a small number of risers. These risers are detected with set a and c, see figure 4.19. These small stairs cannot be detected with set b. This is because the change in height is calculated based on the trajectory which is based on the movements of the MLS device during the data capture. Therefore, part of the trajectory and each riser appears differently in the trajectory and large thresholds are needed. figure 4.20 shows a different staircase. This



(a) (b) (c)

Figure 4.19: Trajectory classification staircase 1, on testing set a, b and c

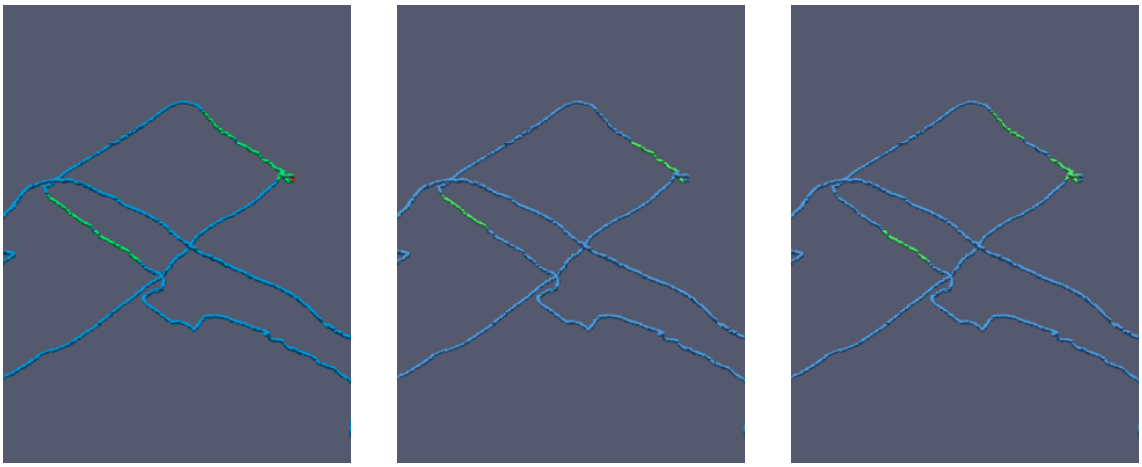
staircase has four of the same stairs which all have the same amount of risers. As can be seen in set a, all four stairs are detected and a small part is classified as a slope. In the second image, one of these stairs has disappeared and part of the bottom stair is removed. This part removed part is still missing in image c. It contains sudden height changes and therefore, the difference between the minimum and maximum thresholds should be quite large. figure 4.21 shows the classification of a slope. As can be seen



(a) (b) (c)

Figure 4.20: Trajectory classification staircase 2, on testing set a, b and c

in a, four slopes are detected. In the second and third image, the changes in the thresholds results in the removal of two slopes. To make sure that all slopes get classified, the difference between the minimum and maximum angle threshold need to be quite large as well. Especially if the angles of Table 4.3 are compared with the angles of figure 4.22.



(a) (b) (c)

Figure 4.21: Trajectory classification slope 1, on testing set a, b and c

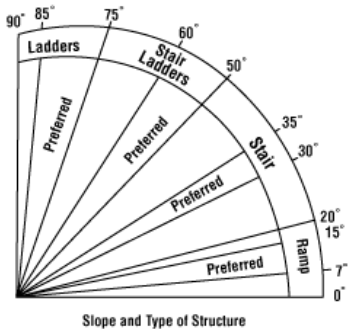


Figure 4.22: Angles for stairs and slopes Image from: http://inspectapedia.com/Stairs

Stairs which have a larger tread depth can have the same angle as sloped floors. Therefore, these trajectory points are classified as sloped floors which is not the case in reality, as can be seen in figure 4.23. To restore these errors, the classification check described in § 3.2.9 is implemented.

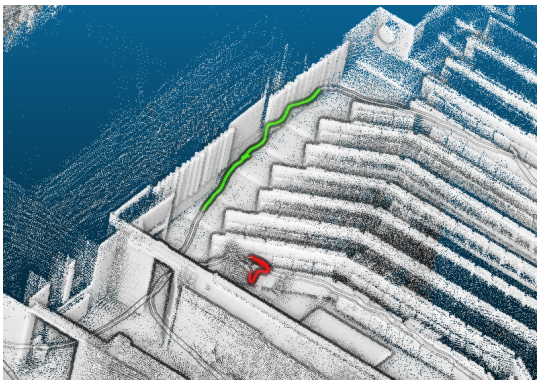


Figure 4.23: Slow stairs: classified as slope

4.6 Voxel model + seed voxels

The identification of seed voxels has no parameter and is implemented in the PostgreSQL database. The algorithm works in the following way:

1. The trajectory points are voxelized in the same spatial structure as the voxelized point cloud.
2. The height difference between the trajectory voxels and the corresponding voxels on the same x, y location are calculated. The seed voxels are the voxels below the trajectory voxels which have the smallest positive distance regarding the trajectory voxels.
3. The records of the seed voxels are changed in the *smallestleaves* table. The value of the *seedpoint* column is changed from 0 to 1, the *locationalcode* of the trajectory voxels are added to the *trlocational* column and the voxel types are added to the *voxeltype* column. There are seven different classes which exists of a stair class, a slope class and a flat class and four combination classes. The combination classes are based on the sum of the first three class ID's, see Table 4.4.

Database id	Surface type
1	slope
2	stair
3	slope + stair
4	flat
5	flat + slope
6	flat + stair
7	flat + slope + stair

Table 4.4: Different classes of seed voxel types

After this process, the seed voxels are identified and classified, see figure 4.24

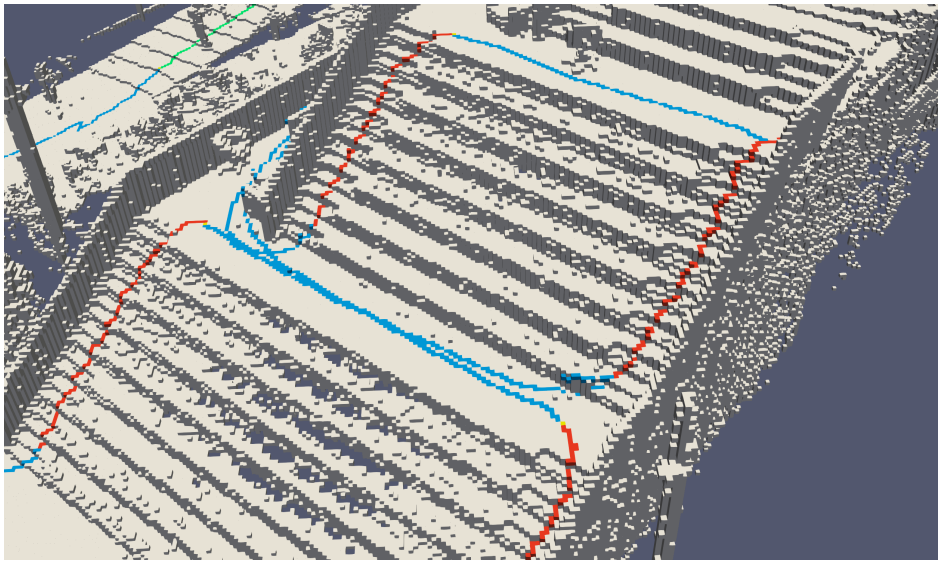


Figure 4.24: Identified seed voxels. Stair (red), flat (blue), slope (green) and multiple types (yellow)

The voxel directly below the voxelized trajectory is identified as a seed voxel. If the MLS device is held above furniture or other objects, seed voxels are classified on top of these objects, see figure 4.25. The

identification of these seed voxels results in the region growing of surfaces which should not be identified as such. These features do not represent a stair, slope or flat surface and thus need to be removed. This cleaning step is implemented in the classification check.

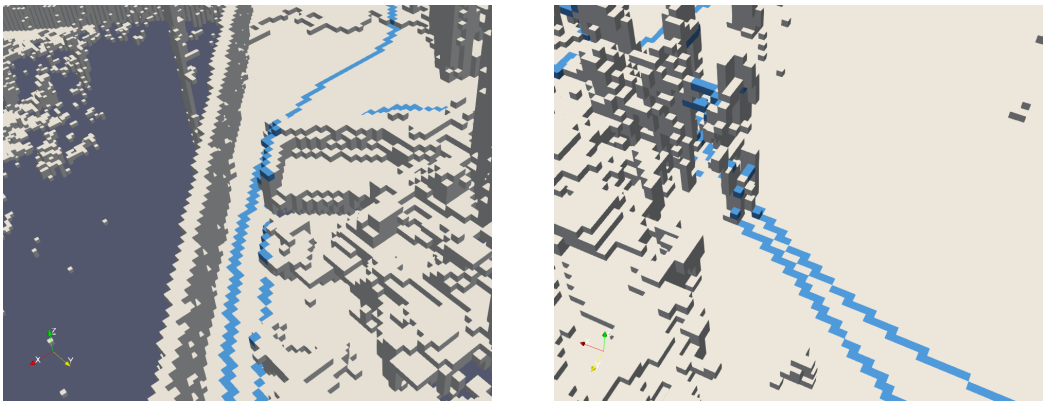


Figure 4.25: Voxels of non-floor elements that have a floor seed voxel on them

4.7 Voxel model + seed voxels + identified entryways

The detection of entryways is implemented in the PostgreSQL database and is based on the seed voxels which are identified by the trajectory. Due to time constraints of this MSc thesis, entryways are only detected using the height changes in the z-axis method and are not fully implemented in the classification process. Therefore, entryways are not identified in the final output of this MSc thesis.
The identification of entryways is implemented in the following way:

1. The *smallestleaves* table is joined with the *trajectory* table to get the sequence number of the seed voxels in the originally ordered voxelized trajectory.

2. For each x, y location all corresponding voxels are found and the z distances are calculated. The negative numbers are deleted and the result is saved in a materialized view.

3. The voxels with the smallest distance to the seed voxels are assumed to be the ceiling voxels.

By ordering the seed voxels on their capturing time and adding the corresponding height value to the ceiling voxels, a 2D image is created, see figure 4.26. The height of the ceiling in the Faculty of Architecture is around 5.9 meters. Entryways are far lower height value and can be detected using a threshold of 2.6 meters. As can be seen in figure 4.26, four entryways are identified. If these entryways are checked with the captured voxel model, the entryways are classified correctly, see figure 4.27. As mentioned earlier, entryways can only be detected if they are passed by the MLS during the data capture. Therefore, not all the entryways in figure 4.27 are detected.

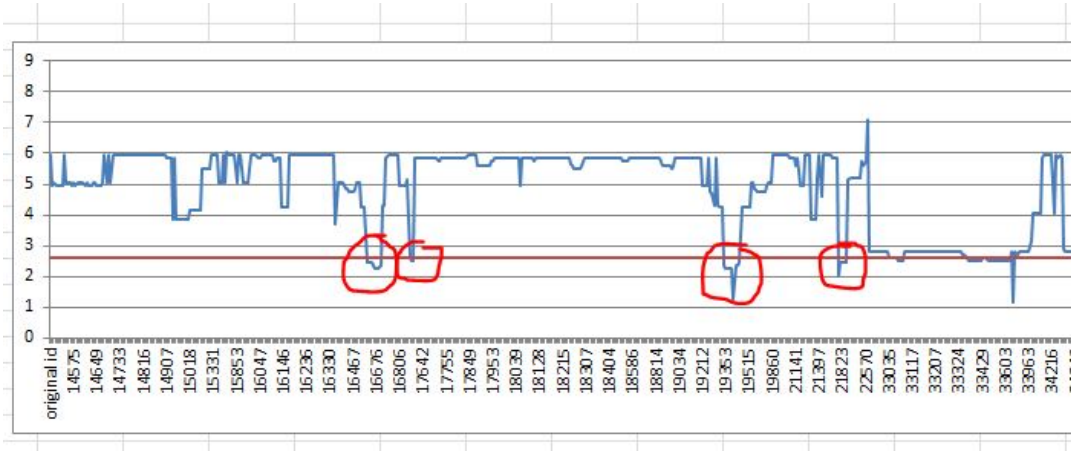


Figure 4.26: Distance from seed voxels to the ceiling. The entryways are detected with an height of 2.6 meters

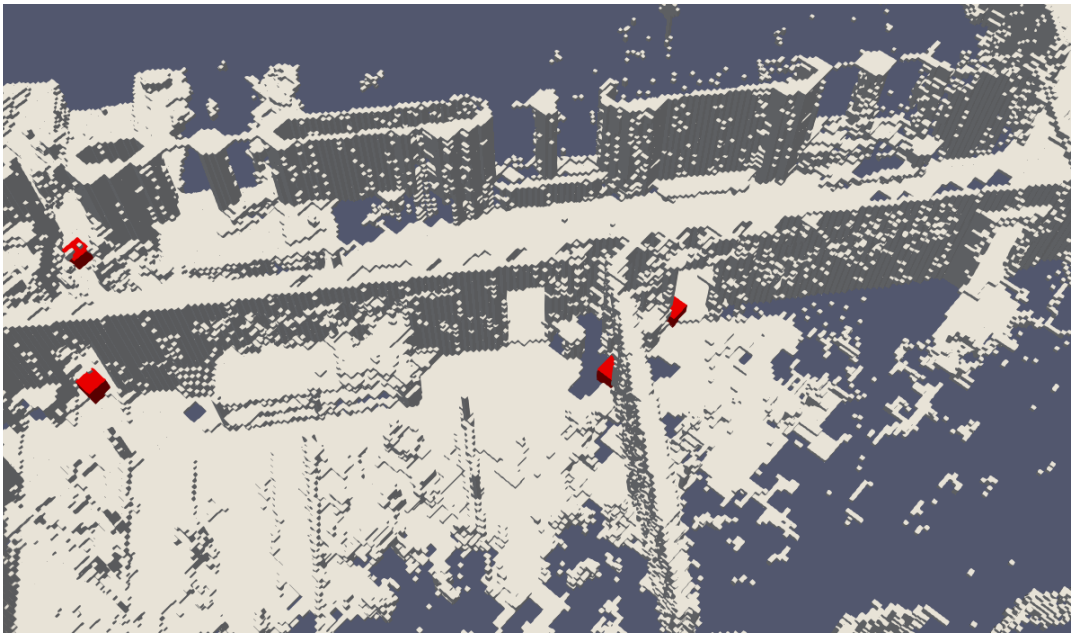


Figure 4.27: Enlarged seed voxels of the identified peaks in figure 4.26

Most of the buildings have a ceiling lower than 5.9 meters. If there is less distance between the seed voxels and the ceiling, the identifying an entryway is more difficult. All infrastructure objects like water pipes, construction beams, lightning objects and electrical wiring are visible in the Faculty of Architecture which makes the identification of entryways even more difficult, see figure 4.28. As can be seen in figure 4.26, the detection of entryways is impossible since all these infrastructure objects produce entryway patterns. This way it seems like all these infrastructure objects are entryways, which is not the case. If the distance between the seed voxels and the ceiling is larger, these infrastructure does not influence the detection of entryways, see .

Currently the detection of entryways is only possible in spaces with a high ceiling. Further research is needed to identify all the entryways of a building based on the trajectory of a MLS.

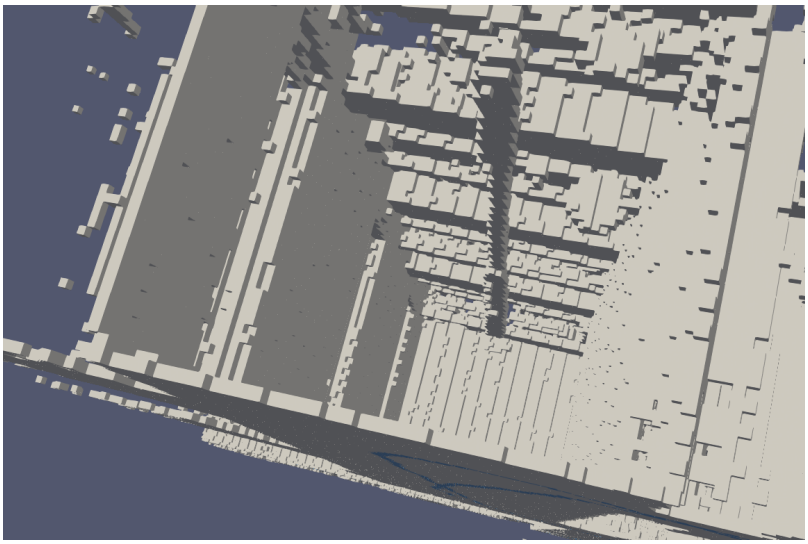


Figure 4.28: Ceiling with visible infrastructure objects

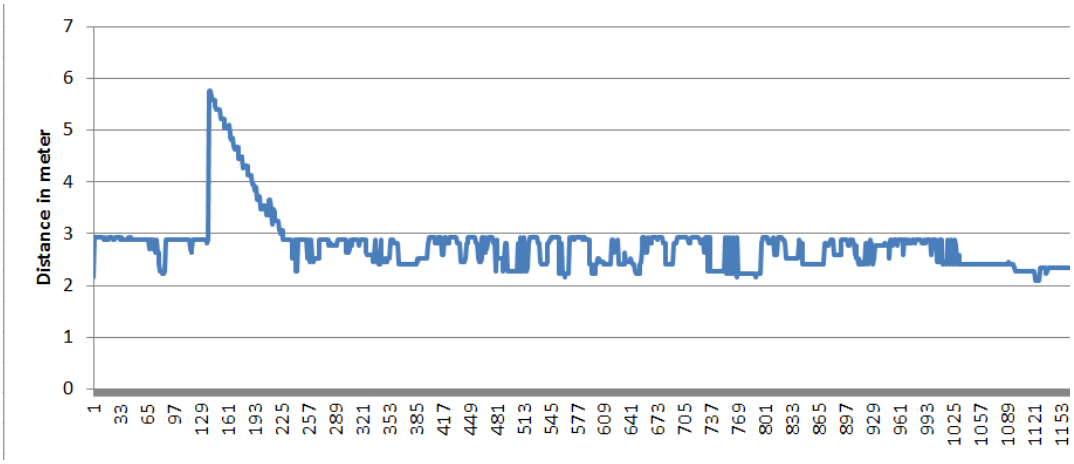


Figure 4.29: Distance from seed voxels to the ceiling

4.8 Regiongrow seed voxels per room

As discussed in § 3.2.8, two region growing methods are implemented. The ordered checking algorithm does not require any parameters. The ClusterDBSCAN algorithm requires two parameters: the distance (EPS) and the density (minpoints). The density parameter influences the number of clusters. A cluster must have more than n-minpoints of data elements to be saved as a cluster. Because the clusters are only saved if they contain seed points, the amount of points or the minimal size of the cluster does not matter. Therefore, the parameter is set to 5 which has no influence on the result. The other parameter is the maximum distance between different region points. Because each voxel is represented by an integer between 0 and $2^{octreedepth}$, all the neighbouring voxels have a distance of 1, see § 4.2 and § 3.2.2: the four neighbour adjacency. The distance to a neighbour considering an eight neighbour adjacency, in the horizontal plane, is 1 to the voxels who share a face. The distance to voxels who share a vertex are described by the Pythagoras theory: $a^2 + b^2 = c^2$ which results in a distance of $\sqrt{2}$ which is around 1.41, see figure 4.30. Therefore, it is useful to test the The desired distance between two points in the

ST_ClusertDBSCAN algorithm (EPS) parameter for a value below and above 1.41 which results in a testing value of 1.3 and 1.5. This way the four neighbour and eight neighbour adjacency is tested.

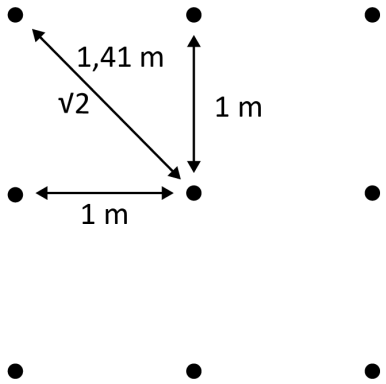


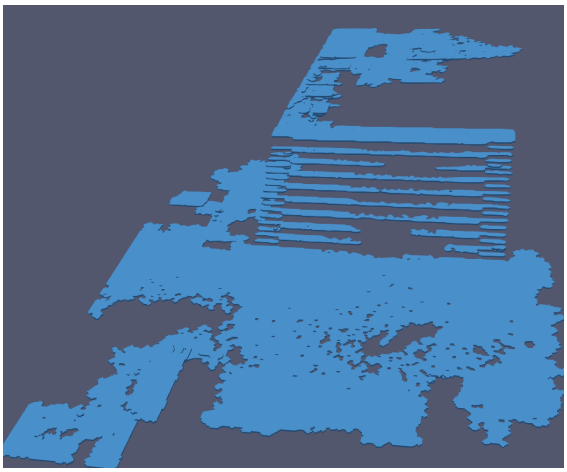
Figure 4.30: Distance to the voxel neighbours

The processing time of both dbscans is ten times as fast as the ordered checking, see Table 4.5. Moreover, the ordered checking contains more voxels than any of the dbscans. Furthermore, the difference of the largest region, considering the amount of voxels, between the dbscan with an EPS of 1.5 and the ordered checking is 0.5%. The difference between the largest region of the ordered checking and the dbscan with an eps of 1.3 is 5.2% which is ten times larger compared to the dbscan of 1.5.

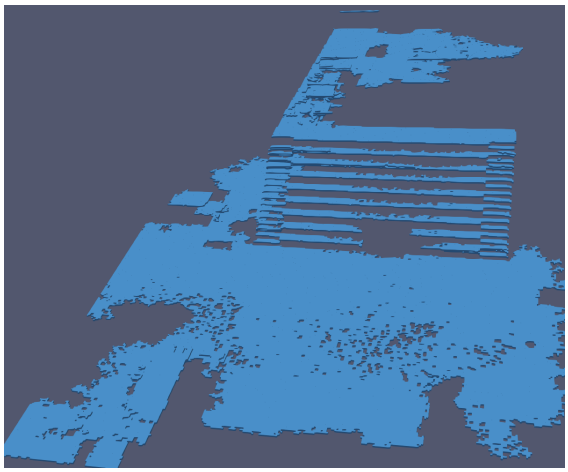
Region growing type	Processing time in minutes	Number of voxels	Largest region in voxels
ClusterDBSCAN EPS = 1.3	17	79144	46063
ClusterDBSCAN EPS = 1.5	17	84540	48320
Ordered checking	169	90292	48568

Table 4.5: Results of the region growing process for two the ordered checking, the SBDCAN with an eps of 1.3 and an eps of 1.5

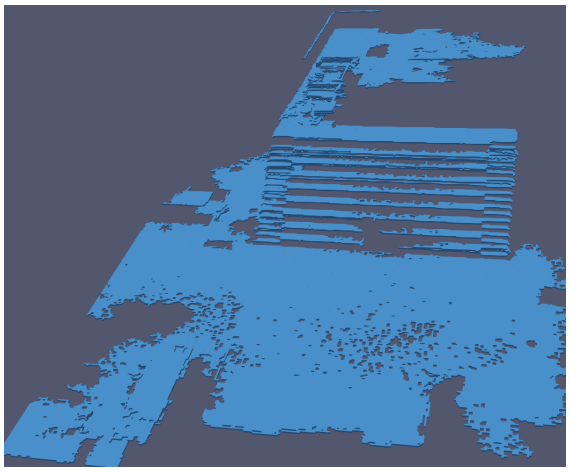
The visual differences between the three regions are not big, as can be seen in figure 4.31a until figure 4.31c. As earlier discussed, the ordered checking adds more regions to the region model compared to the dbscan with an EPS of 1.5. This can be noticed on the right top of the stair. The dbscan with an EPS of 1.5 is compared to the ordered checking much quicker processed, although it visually almost gives the same result. This is strange because the ordered checking works with a four neighbour adjacency, just like the dbscan with an eps of 1.3. The dbscan with an eps of 1.5 which has an eight neighbour adjacency, looks visually and numerically more like the ordered checking. Because the dbscan with an EPS of 1.3 also misses some voxels in the middle of the space and has a 10 times faster processing time than the ordered checking, the dbscan with an eps of 1.5 is used in the main process.
This means that not a four neighbour adjacency described in § 3.2.8 is used but an eight neighbour adjacency. As can be seen in the images, the furniture elements that contained a seed point introduced in § 4.6 are also region grown. These wrongly classified regions need to be removed by the classification check.



(a) Region growing: ClusterDBSCAN with an EPS of 1.3



(b) Region growing: ClusterDBSCAN with an EPS of 1.5



(c) Region growing: Ordered checking

Figure 4.31: Region growing ordered checking and ClusterDBSCAN

4.9 Classification check

The classification check is implemented in Python. The following steps are implemented in order to detect the bottom risers of a stair or the first part of a slope:

1. Identify the rise of a slope in an amount of voxels
2. Visit all successive seed voxels along the capture order of the trajectory
3. Compare the height between the previous visited voxel and the current voxel. If the height difference is negative the voxel is higher than the current voxel. If the height difference is positive the voxel is lower than the current voxel, see figure 4.32
4. If the height difference (negative or positive) is lower or the same as the slope parameter and if the *frontcheck* voxel is classified as a slope voxel or if the *backcheck* voxel is classified as a slope voxel, the first part of a slope is detected

5. This information is updated into the current regions
6. These steps are repeated for the stair parameters. The height difference described in step 4 needs to be changed. In this case, the height difference needs to be the same as or more than the minimal stair rise and needs to be the same as or less than the maximal stair rise

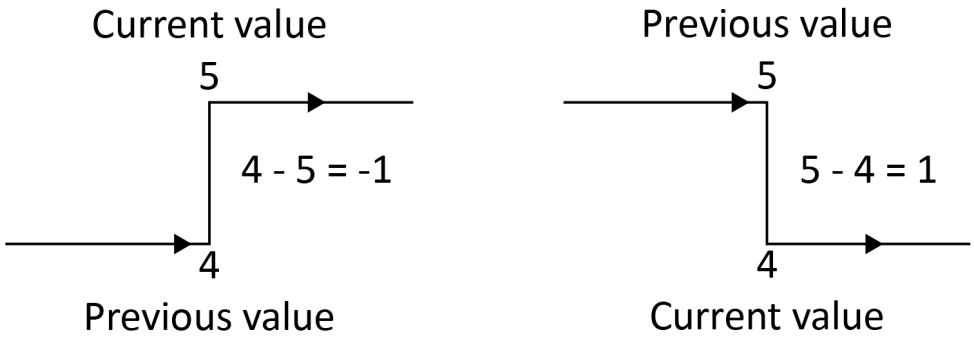


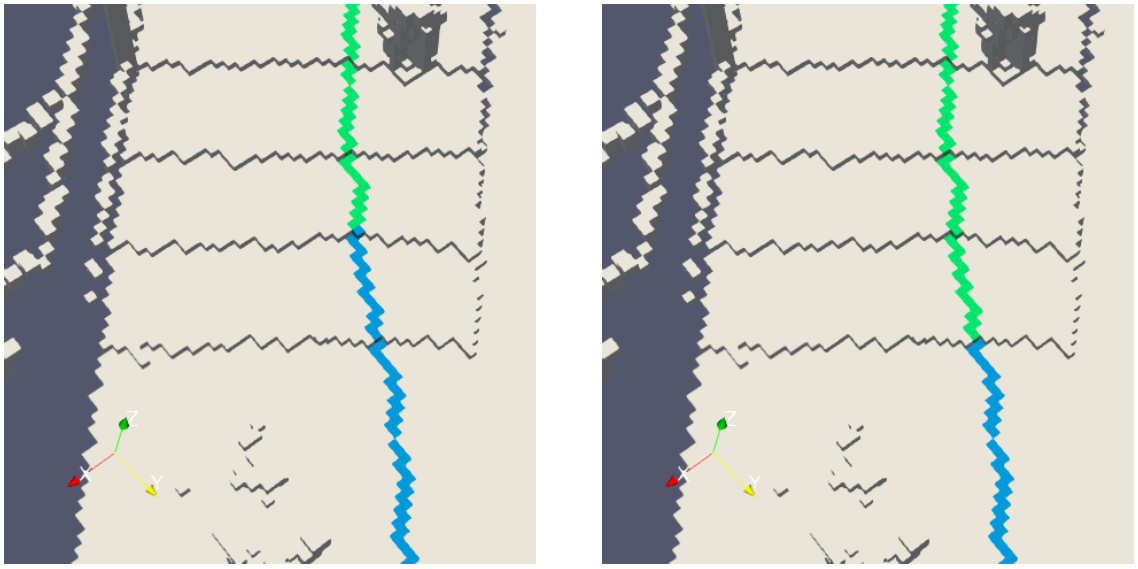
Figure 4.32: Up and down a stair or slope results in a negative and a positive height change

The *frontcheck* and *backcheck* parameters are based on the length of the stairs and slopes. If a height of 1.5 meters needs to be covered by a stair or slope, the length of the slope is longer than the length of the stair. Therefore, the *frontcheck* and *backcheck* of the slope is larger than the parameters of the stair, see Table 4.6. The rise of a slope in the point cloud is gradual which results in a voxel representation which increases gradually with 1 voxel per distance. The bottom stair parameter and top stair parameter are based on the voxel size of around 7 cm and the risers in the indoor environment that were around 15 cm. Therefore, a riser is always represented by 2 or more voxels or 4 or less voxels, see Table 4.6. These parameters are checked in multiple test cases.

Parameter	slope	stair
frontcheck	20	10
backcheck	20	5
Sloperise	1	-
Minimal stair rise	-	2
Maximal stair rise	-	4

Table 4.6: Classification check slope and stair parameters

The implemented code results in figure 4.33. As can be seen, the seed voxels of the bottom part of the slope are classified correctly. If the regions are corrected, the result looks like figure 4.34.



(a) The bottom part of the slope seed voxels (green) are classified as flat seed voxels (blue)

(b) Improved classification: slope seed voxels (green) and flat seed voxels (blue)

Figure 4.33: Seed voxels before and after the classification check

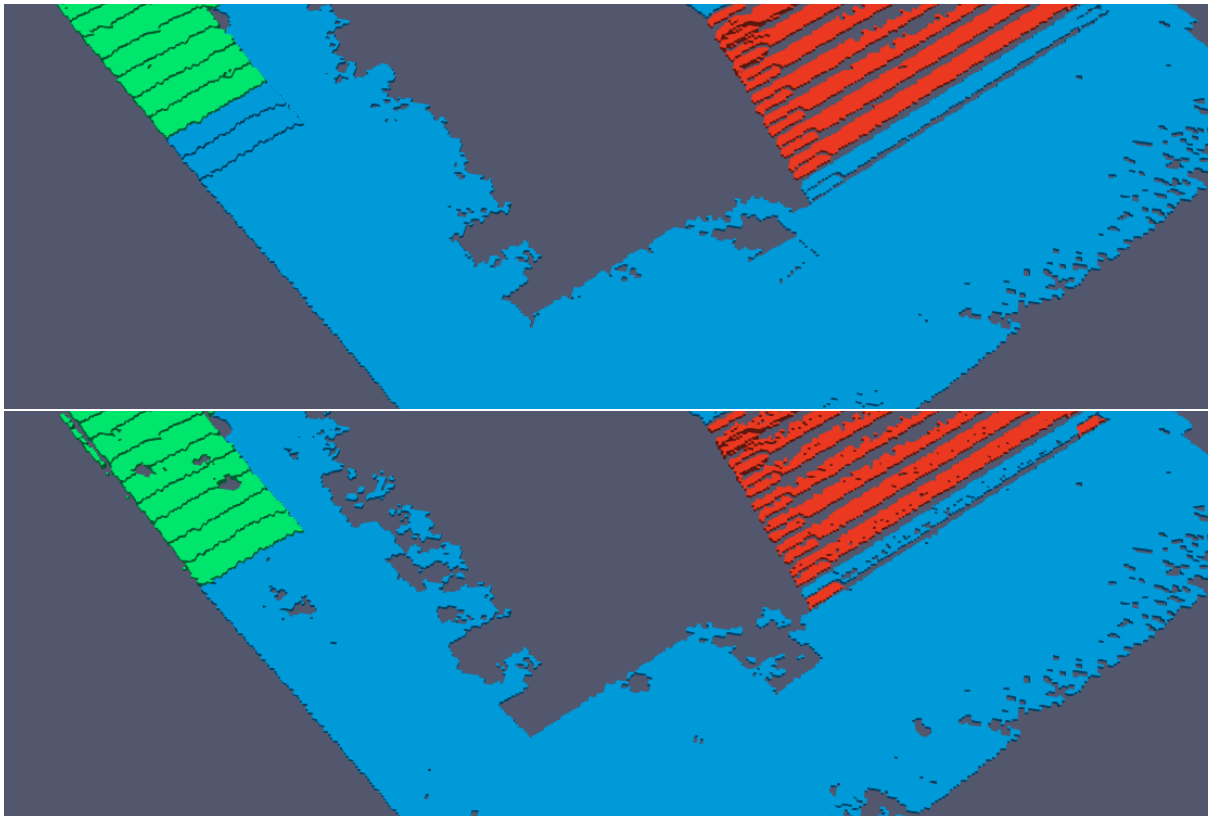
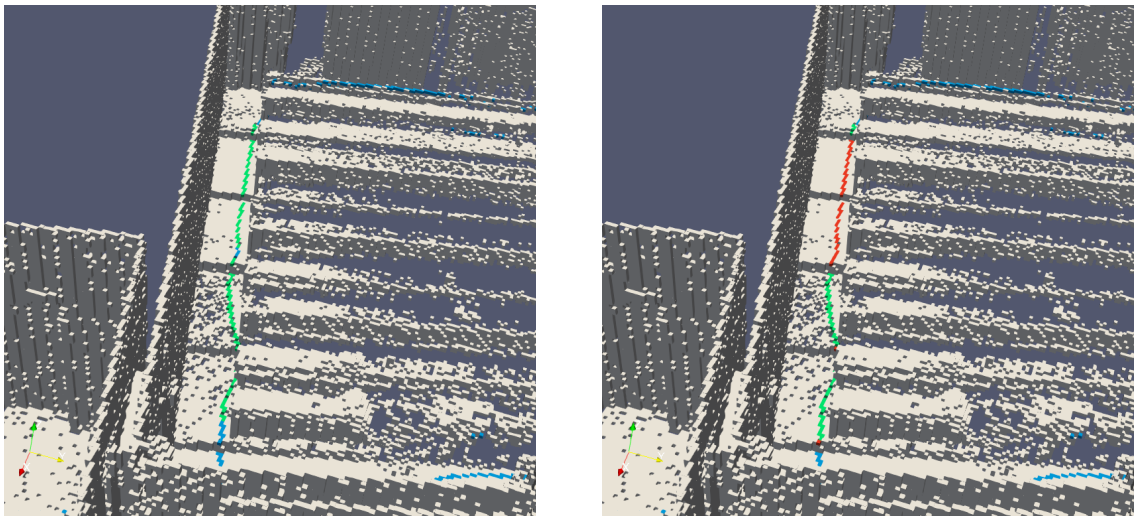


Figure 4.34: Corresponding regions to the trajectory voxels before (above) and after (below) the classification check: flat regions (blue), slope regions (green) and stair regions (red)

This approach can also be used to detect slow stairs identified as slopes, see figure 4.23. The current results can be seen in figure 4.35b.



(a) Stair voxels classified as slope (green) and flat surfaces (blue)

(b) Partial detection of stair seed voxels (red), slope seed voxels (green) and flat surfaces (blue)

Figure 4.35: Detection of stairs wrongly classified as slope

As can be seen in figure 4.35b and the bottom part of figure 4.34, the currently implemented code cannot detect all the stair risers and not all the wrongly identified sloped floors. Also the furniture objects marked as floors cannot be detected currently and are therefore not removed from the navigable voxel space in the following processing step. The proposed classification check can be improved by combining the information of the ordered seed voxels with the floor space belonging to those specific seed voxels and the amount of other seed voxels that belong to the same region. This information can for example be used to get the average stair riser acreage which can be applied to identify flat seed voxels after a stair.

4.10 Subtract furniture

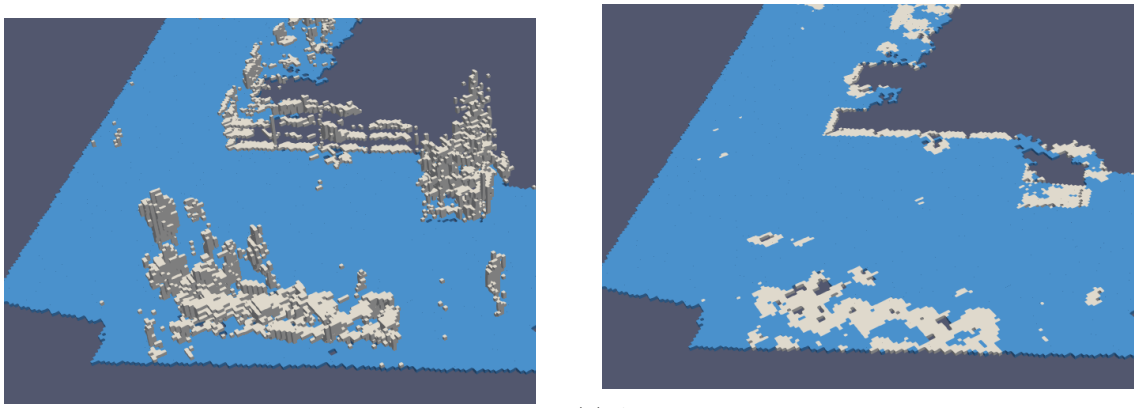
This process is implemented in the PostgreSQL database. The furniture subtraction uses only one parameter named the *maximum checking value*. This value depends on the type of actor which for this thesis is a pedestrian with a preferred height of 2.30 meters. For explanation purposes only the subtraction of furniture objects of one subspace is shown. The described process can be applied to all subspaces at once.
The process exists of the following steps:

1. Identify the amount of voxels that need to be free above each floor region. This is done by dividing the actor height by the retrieved voxel height from the *project* table. This results in a n-amount of voxels above the subspaces, the *actorheight* parameter, which should be kept empty.

2. The voxels of the voxelized point cloud are retrieved from the database and the voxels of the floor regions are removed from these database voxels. The identified floor regions do not represent furniture objects themselves.

3. The floor regions are them joined on the x, y position to the remaining voxels of the voxelized point cloud.
4. The height of the voxel, the z value, is reduced by the height of the corresponding floor region. The distance is now described from 0 to the height of the existing voxels. By removing all the negative heights and the voxels above the *actorheight* parameter, furniture voxels can be identified.
5. These furniture voxels are removed from the floor regions based on the *locationalcode* of the corresponding voxel in the floor region.

By removing the voxels which are below the *actorheight* parameter, the navigable subspaces per actor are identified. In the current implementation, all voxels above the navigation surface are removed. A voxel without any neighbours is probably not a real object but represents noise and is subtracted. In further researches, this voxel should be detected instead of being removed from the navigable subspaces per actor.
There are also some navigable subspaces per actor which cannot be reached by the actor; this is the case in cluttered environments. This can be seen close to the table in the left bottom corner of figure 4.36a. These unreachable voxel subspaces can therefore be removed from the navigable subspace.
A graphical representation of the furniture objects which are subtracted from a subspace is illustrated in figure 4.37. Dynamic objects like pedestrians that were not moving during the data capture are not detected and therefore also removed from the navigable subspaces.



(a) A subspace with furniture and building objects

(b) A subspace with furniture and building objects merged to one height

Figure 4.36: A subspace with furniture elements

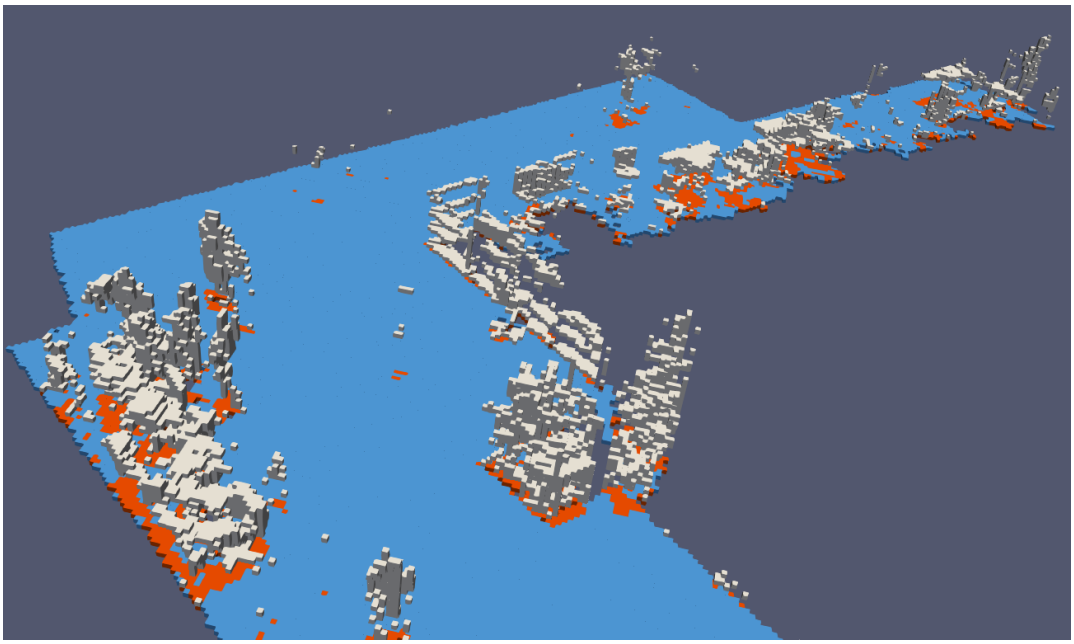


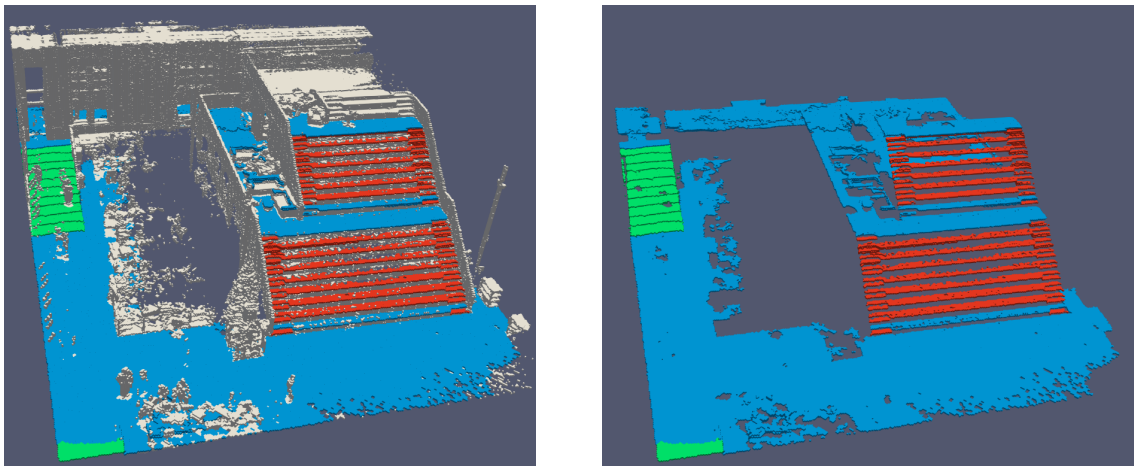
Figure 4.37: Navigable subspace (blue), removed furniture subspace (red), furniture and building elements above the subspace (white)

4.11 Final navigation voxel space

The output of the proposed method is described as the navigable voxel space per actor. In this section, the final output model is compared to the situation in reality and the processing time of the proposed method will be discussed.

4.11.1 Navigation voxel space

After the different processing steps, the indoor navigable voxel space is identified, see .



(a) Navigable voxel space with surrounding voxels: sloped surfaces (green), flat surfaces (blue) and stair surfaces (red)

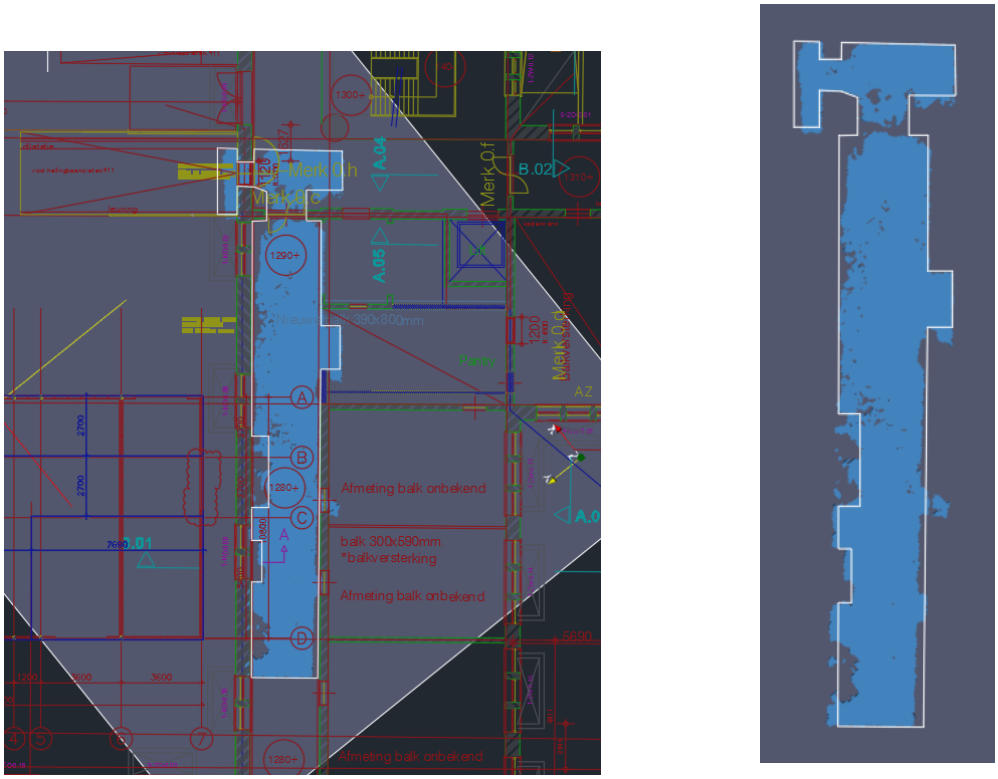
(b) Navigable voxel space without surrounding voxels: sloped surfaces (green), flat surfaces (blue) and stair surfaces (red)

Figure 4.38: Navigable voxel space

4.11.2 Representation

The final output of the described method is the navigable space per actor. If the m^2 of the navigable space of the model is close to the m^2 of environment in the real world, the method produces an accurate m^2 model. The accuracy check is implemented on a Computer-aided Design (CAD) model of the Faculty of Architecture, as can be seen in figure 4.39a. figure 4.39b shows the extends of the navigable space from the proposed method and the navigable space of the CAD model.

As can be seen in figure 4.39b, some areas are navigable spaces in the CAD model but are not marked as such in the voxel model. These areas could be removed described by the process in § 4.10 or are not scanned thorough enough and thus should be scanned more detailed.
Besides the missing data in the corner, an entryway (on the top of the image) is removed from the navigable space. This is because the entryway falls within the height of the actor. If the entryway detection was implemented, these entryway voxels would not have been removed in the final output. There is also a gap visible close to the entryway. This gap is created by a pedestrian that was not on moving during the data capture which therefore is not removed in the detection of dynamic objects. These problems influence the total m^2 of the final model.
The m^2 comparison is applied to two different navigable subspaces:
1. The hallway subspace represents a space with a few objects along a wall. 2. The first floor subspace of the orange rock represents an environment with a lot of chairs, desks and scale models which is also



(a) Area calculation in a **cad!** model

(b) Comparison between identified navigable space and the real world navigable space from a **cad!**

Figure 4.39: Representation of the area

known as a cluttered environment.
Besides the two test cases, the m^2 is calculated for the chosen voxel size of 7.3 cm and a smaller voxel size of 3.65 cm. As can be seen in Table 4.7, there are some differences between the m^2 with the 7.3 cm voxel size and the m^2 of the CAD model. It can also be noted that the difference in m^2 between the voxel size of 7.3 and 3.65 is not that big. Based on these results it can be assumed that an increase in voxel size does not influence the representation in m^2 a lot. Furthermore, the difference in m^2 between the CAD model and the final navigable space is around 10%. These conclusions are only based on two test cases. More researches are needed to compare these results with the results of other indoor spaces.

Checking type	Halway 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
3.65 cm voxel model	67.5	61.9
Difference between CAD and 7.3 cm	-8.5 %	-9.9 %
Difference between CAD and 3.7 cm	-8.8 %	-9.0 %

Table 4.7: Area calculation in a **cad!** model compared to the area of the output model with an voxel size of 7.3 and 3.7 cm

4.11.3 Performance

The processing time of the method is illustrated in figure 4.40. The method has been tested for different point clouds of four, eight and sixteen million points. By doubling the amount of points, the processing time almost doubled as well. As can be seen in the figure, the voxelization process of 4 million points requires half of the total processing time. However, with 18 million points the voxelization process requires more than half of the total processing time. Furthermore, it can be noted that the processing time is decreased significantly by the implementation of the DBSCAN algorithm, see Table 4.5.

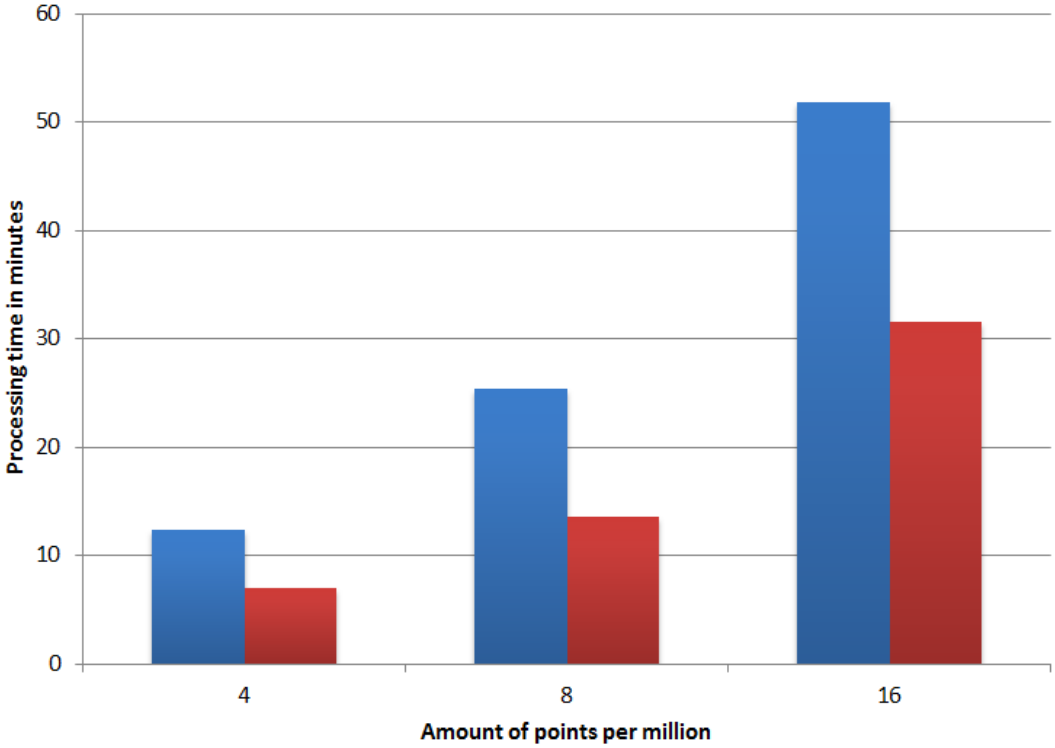


Figure 4.40: Processing time of point clouds from 4, 8 and 16 million points. Total time (blue), voxelization time (red)

4.11.4 Tresholds: fixed or changeable

There are a lot of parameters used in the development of this method. Some parameters can and other parameters cannot be changed without large consequences within the final model. This section discusses what would happen when the parameters which are used would be changed and which effect this would have on the final model.

1. Octree depth: this parameter is important because it defines the voxel size. The scaling factor of each point cloud is different which results in different voxel sizes with the same octree depth. The octree depth parameter needs to be adapted until the preferred voxel size is reached. Therefore, the effect of a changing voxel size and not the octree depth parameter is further discussed.
   In the current implementation, a voxel size of around 7.3 is used. If the voxel size increases, the voxels will get bigger and there will be problems with the identification of objects like the risers of stairs. Furthermore, the *Numoftimestamps* parameter needs to be larger which results in the

removal of a lot of voxels before the dynamic objects are detected.
If the voxel size is two times as smaller, a voxel size around 3.65, the final result is not influenced that much as can be seen in § 4.11.2. There will be created more floor regions with the smaller voxel size because gaps in the voxel surfaces can occur and the six eight neighbour relations during the region growing process cannot be applied as to a bigger voxel sizes. Even smaller voxel sizes will probably result in even more gaps in the model which will finally result in only the identification of navigable space around the seed voxels identified by the trajectory. The detection of the entire navigable space is then not possible any more.

2. *Numoftimestamps*: increasing this parameter results in the removal of voxels in the model which are no dynamic objects. A increase of this parameter results in gaps in the voxel model and gaps in the final navigable space.

3. *Filgaps*: when the value of this parameter is changed, it will not have a large impact on the final output. By increasing the value of this parameter, extra voxels will be added between for example a stair and a riser border. This results in different representations of the objects in the voxel model.

4. Trajectory classification parameters: as the experiments in this thesis showed, changing the values of these parameters results directly in stairs and slopes which are not classified correctly. Therefore, these parameter should not be changed.

5. Entryway height: this parameter is not implemented and therefore not tested. However, this parameter will probably be heavily influenced when the height difference between the ceiling and the top of an entryway gets smaller.

6. Region growing, *minpoints*: this parameter should be kept at a small value because all regions should be created. The regions containing a seed voxel will be saved.

7. Region growing, EPS: this parameter should also not be changed. Increasing this parameter results in the detection of neighbours that are not connected to the voxel and are further away which results in the wrong classification of regions across small walls or furniture objects.

8. *Frontcheck* and *backcheck*: the value of these parameters are based on two times the riser depth and two times the voxel slope length. Therefore, this parameter depends heavily on the voxel size. There will not be a large impact on the final result if this parameter increases.

9. *Sloperise*: a slope rises gradually in the point cloud and increases with 1 voxel in the voxel model. Therefore, this parameter is set to 1 and cannot be changed.

10. Minimal and maximal stair rise: these parameters depend on the amount of voxels which can represent a stair. In the current model, the risers of a stair are around 15 cm. Therefore, this parameter cannot be changed. This parameter depends on the voxel size and the smallest and largest risers of a stair.

11. Actor height: this parameter is necessary and depends on the height of the actor. This value can be higher than the actor, but should never be smaller than the height of the actor. When the value is increased a lot, especially in spaces with low ceilings, a lot of navigable space will be removed.

5 Discussion and conclusion

This thesis investigates the possibilities for the use of the trajectory of a mobile laser scanner to identify navigable spaces. Before the conclusion can be drawn, the research question and sub questions, as defined in § 1.2.1, will be answered. After this section, the method will be discussed and final conclusions will be drawn. Afterwards, recommendation for future researches will be given.

5.1 Research question

In this section, the research question and subquestions introduced in § 1.2 are answered.

1. What are the characteristics of a walkable space?
   Forming a definition of a space is difficult based on the amount of literature, as discussed in § 3.1. A general definition of a space can always be used but is unfortunately not specific enough. Therefore, the definition of a space is most of the time extended for the current model that is used. This results in a lot of different definitions of a space. Broadly, the concept of a space can be categorized in two categories. The first describes a space as the result of the architectural components like walls, floors, electric cabling and other building components. The second describes a space based on the function it has like entrances, waiting areas and so on. In this thesis, the definition of a space a section of the indoor environment which is enclosed by walls, entryways, windows, floors and ceilings that can be accessed by an actor through an entryway. A spaces contain different components like furniture objects, lightning features, plants, construction elements and are used for indoor navigation activities.
   Actors that use a space can be categorized in the type of navigation surfaces they can use for their navigation activities. Some actors can use a flat and sloped floor but cannot use stairs. Therefore, it is important to further subdivide a space in subspaces. A subspace is a section of a space which only contains one type of connected navigation surface. A navigable subspace per actor is an environment in which an actor can navigate without bumping into any obstacles on the ground or in the air. The navigable space is cleared from obstacles (furniture and building components) until the height characteristics of the actor are reached.

2. In what way can the trajectory of a mobile laser scanner be used to identify a walkable space?
   If the MLS is operated by a human, the trajectory contains information that can be used to identify walkable spaces. Therefore, points directly below the trajectory represent human navigable points. By structuring the point clouds into a voxel model, the number of the data records is reduced and is structured in space. If the trajectory is voxelized in the same spatial structure and projected on the voxelized model, seed voxels can be identified. These seed voxels are the representation of the trajectory into the voxelized point cloud. By region growing these seeds in a horizontal plane, pre-walkable voxel surfaces can be defined. Because the MLS gathers data from the entire indoor space, surfaces below furniture and building objects are also identified. By removing these elements from the pre-walkable voxel space, the navigable voxel space can be defined. In this way, the trajectory forms the basis for the final output and cannot be replaced by other data.

3. How can walkable areas, identified by a mobile laser scanner trajectory, be subdivided into different spaces?
   The seed voxels described above can be used to detect entryways. By checking the height differences between these seed voxels and the ceiling, a sudden height decrease and increase can be detected. This characteristic pattern indicates the position of an entryway. This way, the scanned indoor environment can be subdivided into different spaces. These spaces contain different kinds of navigation surfaces like stairs, slopes and flat surfaces. Not all actors can use the same navigation surface. Therefore, the identification of these surfaces is crucial. Distinction between these navigation surfaces can be made by further subdividing spaces into subspaces. This is done based on the change in height of the trajectory of the MLS.

4. In what way does the voxel size influence the accuracy of the generated walkable space?
   The voxel size influences the representativeness of the voxel model compared to the input point cloud. If the voxel size is too large, different components of the indoor space cannot be detected. This results in a wrong classification of objects which has a negative effect on the final navigable voxel space. In this thesis, a riser of a stair needs to be identified. For this purpose a voxel size of 7.3 cm has proven to be sufficient. As described in Table 4.7, the actual difference in m^2 between the implemented method and the real world is around 10 % with a voxel size of 7.3 cm. Using a smaller voxel size does not improve the identified m^2. More tests need to be performed to validate if this value is indeed correct.

After discussing the different sub questions, the main research question can be answered:
Which walkable space can be identified from a voxelized point cloud using the trajectory of a mobile laser scanner?
Point clouds are unstructured points containing only a x, y and z coordinate. To structure a point cloud in space, an octree approach is used. The size of the voxels depends on the number of subdivisions. The point cloud is automatically scaled so that it fits the octree structure. This results in different voxel sizes based on the extends of the point cloud.
The voxel size is important to detect different objects in the indoor environment. A large voxel size over simplifies the indoor environment and makes the detection of objects impossible. A small voxel size represents the indoor environment more accurate but also increases the amount of data elements and probably the processing time. In this thesis, a voxel size of 7.3 cm is used.
Because the building is not closed during the data capture, dynamic objects that were present during the data capture are also present in the voxelized model. These dynamic elements are detected and removed because they do not represent building elements. Dynamic objects that were static during the data capture have not been removed from the voxel model and are therefore still present in final model.
The trajectory plays an important role in the implemented method. Different navigation surfaces can be defined by analysing the height differences of the trajectory. These navigation surfaces exists of stairs, flat floors and sloped floors. This is achieved through the voxelization of the trajectory in the same spatial structure as the voxelized point cloud. By projecting these voxels on the voxel model, seed voxels can be identified. Assuming that entryways can be identified using the height between the seed voxels and the ceiling, different spaces can be split into single spaces. Floor spaces are found by region growing the seed points into floor regions on the same horizontal plane. These floor regions contain furniture objects, building objects and dynamic objects that were not moving during the data capture. By removing these objects from the floor regions, the navigable voxel space can be defined.
The accuracy of this process compared to a CAD model is around 10 %. Further testing is needed to check the correctness of this value. An increase in the voxel size does not improve the results of the represented m^2. Concluding can be said that the proposed method identifies three different kinds of navigable voxel subspaces based on the trajectory and the point cloud of the MLS device. This method makes it possible to create a continuous navigable space inside buildings which may consist of floors, stairs and elevations. Data can be captured during business hours because the method detects and removes dynamic objects in the final result. The proposed method can be used for any type of building without any constraints because the complexity of the building is already present in the trajectory of the MLS.

5.2 Discussion

The method which is used in this thesis clearly illustrates that by combining the trajectory with a voxel model of a point cloud, navigation surfaces can be identified without constraints like for example a Manhattan World or a flat surface constraint. This way, the method is applicable in many more buildings. Horizontal, sloped and stair surfaces are identifiable when using this method. However, rounded surfaces cannot be detected in the current implementation. Furthermore, stairs are detected when their risers are circa 15 cm. Yet, when a riser is lower than this value it could not be identified with the current voxel size. However, a temporarily smaller voxel size can be applied to detect these risers which are lower than 15 cm. If even smaller voxels sizes are needed, the original point cloud could also be temporarily used to detect stairs which have an unique riser pattern.
In this research a trajectory of MLS is used. This method can also be applied regarding other trajectories. If a 3D model is created based on photogrammetry which images are collected by a human operator, the trajectory which is based on the position of the images could also be used for this method. It is also possible to draw trajectories in already captured point clouds which are for example captured by a TLS device. It is not expected that the trajectory of drones can be used which can be applied for this method since they fly above unnavigable and navigable surfaces. Therefore, this trajectory does not contain the required walkability aspect that is present when a human operator captures the data.
At this moment, the classification of the different types of sub spaces is based on the trajectory of the MLS. It would be better when the classification of the different surface types is based on the seed voxels themselves since the quality of the trajectory of the MLS depends on the movements of the device during the data capture.
Although this method is only tested for an indoor environment, the method could also be applied in an outdoor environment. However, a couple of changes to the implemented method which is used in this thesis should be made. First, the definition of a space needs to be changed since it does not comply with an outdoor space. Second, an outdoor space has curved roads for the drain of water. The implemented method will probably work on these roads, a possible smaller voxel size is required. Furthermore, outdoor spaces contain smaller height differences between surfaces which also require a smaller voxel size. Outdoor surfaces are also more likely not to be not horizontal. If it is not possible to use smaller voxels, because they do not touch any more, the original point cloud can be used which resumes other analysing processes. Third, it would be difficult to distinguish between surfaces which are on the same horizontal level, for example bicycle paths and pedestrian paths, since they are detected as one surface even though they are used separated. Colour information could help by identifying these different surfaces in an outdoor environment. This feature is currently developed by the supplier of the MLS, which is used in this research. Colour information can also help by the identification of different objects in the indoor environments like pedestrians and furniture objects.

5.3 Conclusion

The proposed method, which was formed during this thesis, was based on several experiments. As described in this thesis, the trajectory of a MLS which is operated by a human operator contains valuable information. Since a human operated the MLS, he has been at exactly the same position as the mobile laser scanner and thus the surfaces below this trajectory indicates human walkable areas. Combining this knowledge to reconstruct the indoor space results in a new way of identifying navigable space and building reconstruction. Since there are no constrains applicable, in contrast to other approaches which use for example a Manhattan World constraint, there are far more building types where this method can be applied. Furthermore, this method is also suitable to create up-to-date floor plans in a short amount of time. This creates even more possibilities for the usage of this method and is especially useful for companies which want to make quick models of their buildings. Also, when this method is used for the creation of navigable maps, the method creates one continuous navigate flow throughout the entire

building. Moreover, this method can identify horizontal spaces, stairs, sloped floors and even small split level floors. Furthermore, static and dynamic object can be detected, identified and removed which is the reason why the data can be captured during business hours. Unfortunately, a curved floor is not detectable at this moment which could be the subject of future research. Furthermore, a new definition of a space, a subspace and a navigable sub- space per actor is introduced in this thesis;
- A space is a section of the indoor environment which is enclosed by walls, entryways, windows, floors and ceilings that can be accessed by an actor through an entryway.
- A subspace is a a section of a space which only contains one type of connected navigation surface.
- A navigable subspace per actor is a subspace of a space which consists of one type of connected navigation surface where an actor, with a specific height, can navigate without colliding into any element.
- A navigable space per actor is a section of the indoor environment which is enclosed by walls, entryways, windows, floors and ceilings that can be accessed through an entryway (the collection of subspaces) where an actor, with a specific height, can navigate without colliding into any element.

Concluding, the implemented 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 and the identification of the navigable spaces only have a 10% difference in the m^2 compared to the environment where the data was captured.

5.4 Future work: recomended research

This paper proposed a methodology 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 used to stop adding voxels when the there exist two voxels directly above the floor voxel. This gives an indication for the location of possible walls. If the total number of voxels above the floor voxel are summed until a specific height, it results in a heat map of the indoor space as discussed in § 4.3.5. By detecting linear elements, walls can be identified.
   Walls can also be detected 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 currently marked as furniture objects. These voxels are subtracted from the floor regions. It would be better to group the furniture features and identify the type of furniture. 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. Therefore, a chair in the middle of a free space does not have to be removed from the final navigation voxel space. This way, there can be more details concerning the furniture types.

3. Identification of dynamic objects which do not move during the scanning
   Dynamic objects which were not moving during the data capture are now marked as furniture. Future research should focus on identifying these objects and should remove them from the voxelized point cloud. At this moment they can block a path which is normally available and therefore should be included in the final result. This can be done by using the described process in § 4.3.5. Because pedestrians were standing at one position they are better scanned which makes their human shape more recognizable. When objects like these are projected down onto the floor, as discussed in § 4.3.5, they could be detected and classified as dynamic objects.

4. Generation of a node network (network graph)
   Navigation applications are mostly based on node networks. With the identification of entryways,

stairs, flat and sloped surfaces, such a network can be generated in for example the IndoorGML standard. This network is necessary when the identified navigation space investigated in this research is implemented in mobile applications which have limited memory capabilities. 4.3.3

Bibliography

Adán, A., Quintana, B., Vázquez, A. S., Olivares, A., Parra, E., & Prieto, S. (2015). Towards the automatic scanning of indoors with robots. *Sensors*, *15*(5), 11551–11574. Retrieved from http://www.mdpi.com/1424-8220/15/5/11551 doi: 10.3390/s150511551

Anagnostopoulos, I., Pătrăucean, V., Brilakis, I., & Vela, P. (2016). Detection of walls, floors, and ceilings in point cloud data.. doi: 10.1061/9780784479827.229

Bailey, T., & Durrant-Whyte, H. (2006, Sept). Simultaneous localization and mapping (slam): part ii. *IEEE Robotics Automation Magazine*, *13*(3), 108-117. doi: 10.1109/MRA.2006.1678144

Boguslawski, P., Mahdjoubi, L., Zverovich, V., & Fadli, F. (2016). Automated construction of variable density navigable networks in a 3d indoor environment for emergency response. *Automation in Construction*, *72*(Part 2), 115-128. doi: 10.1016/j.autcon.2016.08.041

Broersen, T., Fichtner, F. W., Heeres, E. J., Liefde, I. d., Rodenberg, O. B. P. M., Verbree, E., & 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., & Kolbe, T. H. (2013). Modelling 3d topographic space against indoor navigation requirements. In J. Pouliot, S. Daniel, F. Hubert, & A. Zamyadi (Eds.), *Progress and new trends in 3d geoinformation sciences* (pp. 1–22). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-29793-9_1 doi: 10.1007/978-3-642-29793-9_1

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

building SMART. (n.d.). *5.4.3.45 ifcspace.* Retrieved from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/schema/ifcproductextension/lexical/ifcspace.htm (Accessed: 2017-03-16)

Díaz Vilariño, L., Boguslawski, P., Khoshelham, K., Lorenzo, H., & Mahdjoubi, L. (2016). Indoor navigation from point clouds: 3d modelling and obstacle detection. In (Vol. XLI-B4).

Durrant-Whyte, H., & Bailey, T. (2006, June). Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, *13*(2), 99-110. doi: 10.1109/MRA.2006.1638022

Eastman, C. M., & Siabiris, A. (1995). A generic building product model incorporating building type information. *Automation in construction*, *3*(4), 283–304.

Ekholm, A. (1996). A conceptual framework for classification of construction works. *Electronic Journal of Information Tech-nology in Construction ITcon*, *1*, 25–50.

Ekholm, A., & Fridqvist, S. (2000). A concept of space for building classification, product modelling, and design. *Automation in Construction*, *9*, 315–328.

Fichtner, F. W. (2016). *Semantic enrichment of a point cloud based on an octree for multi-storey pathfinding* (Unpublished master's thesis). Delft University of Technology.

Fuentes-Pacheco, J., Ruiz-Ascencio, J., & Rendón-Mancha, J. M. (2015, January). Visual simultaneous localization and mapping: A survey. *Artif. Intell. Rev.*, *43*(1), 55–81. Retrieved from http://dx.doi.org/10.1007/s10462-012-9365-8 doi: 10.1007/s10462-012-9365-8

Geometius. (2017). *Main website.* Webpage. Retrieved from http://www.geometius.nl/ (Visited: 21 june 2017)

GeoSLAM. (2016). *Zeb-revo user's manual v1.0.2.*

Geospatial slam. (n.d.). http://www.geoslam.com/company/about-slam/. (Accessed: 2017-03-10)

Gilliéron, P.-Y., Büchel, D., Spassov, I., & Merminod, B. (2004). Indoor navigation performance analysis. In *Enc gnss* (p. 30).

Gunduz, M., Isikdag, U., & Basaranera, M. (2016). a review of recent research in indoor modelling & mapping. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 289–294.

Holenstein, C., Zlot, R., & Bosse, M. (2011, Sept). Watertight surface reconstruction of caves from 3d laser data. In *2011 ieee/rsj international conference on intelligent robots and systems* (p. 3830-3837). doi: 10.1109/IROS.2011.6095145

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* (Unpublished master's thesis). Budapest University of Technology and Economics.

Karas, I. R., Batuk, F., Akay, A. E., & Baz, I. (2006). Automatically extracting 3d models and network analysis for indoors. In A. Abdul-Rahman, S. Zlatanova, & V. Coors (Eds.), *Innovations in 3d geo information systems* (pp. 395–404). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-36998-1_31 doi: 10.1007/978-3-540-36998-1_31

Khoshelham, K., & 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*, *XL-5*.

Koopman, M. (2016). *3d path-finding in a voxelized model of an indoor environment* (Unpublished master's thesis). Delft University of Technology.

Lee, J., Ki-Joune Li, K.-J., Zlatanova, S., Kolbe, T., Nagel, C., & Becker, T. (2016). *Ogc® indoorgml.* Retrieved from http://docs.opengeospatial.org/is/14-005r4/14-005r4.html (Accessed: 2017-03-15)

Li, G. (2014). *Automatic detection of temporary objects in mobile lidar point clouds* (Unpublished master's thesis). University of Twente.

Litomisky, K., & Bhanu, B. (2013). Removing moving objects from point cloud scenes. In X. Jiang, O. R. P. Bellon, D. Goldgof, & T. Oishi (Eds.), *Advances in depth image analysis and applications: International workshop, wdia 2012, tsukuba, japan, november 11, 2012, revised selected and invited papers* (pp. 50–58). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-40303-3_6 doi: 10.1007/978-3-642-40303-3_6

Macher, H., Landes, T., & 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*, *Volume XLI-B5*.

Maier, D., Hornung, A., & Bennewitz, M. (2012, Nov). Real-time navigation in 3d environments based on depth camera data. In *2012 12th ieee-ras international conference on humanoid robots (humanoids 2012)* (p. 692-697). doi: 10.1109/HUMANOIDS.2012.6651595

Nikoohemat, S. (2016, October). Permanent structure detection in cluttered point clouds from indoor mobile laser scanners (imls)..

Nourian, P., Gonçalves, R., Zlatanova, S., Arroyo, K., & Ohori, A., K. A. Anh Vu Vo. (2016). Voxelization algorithms for geospatial applications: Computational methods for voxelating spatial datasets of 3d city models containing 3d surface, curve and point data models. *MethodsX*, *13;3*, 69-86. doi: 10.1016/j.mex.2016.01.001

PostgresSQL. (2017). *8.9. spatial relationships and measurements.* Retrieved from http://postgis.net/docs/manual-dev/ST_ClusterDBSCAN.html (*http* : *//postgis.net/docs/manual* − *dev/ST*$_C$*lusterDBSCAN.html* (7 April 2017))

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

R. Dumusc, R., Gonzalez, G., Lucchi, A., & Fua, P. (2013). *Multi-scale rendering of huge 3d volumes.* Webpage. (Visited: 4 may 2017)

Rovers, A., De Vreede, I., Rook, M., Psomadaki, S., Nagelkerke, T., Quak, W., ... Verbree, E. (2015). *Semantically enriching point clouds: The case of street levels* (Master's thesis, Delft University of Technology). Retrieved from http://repository.tudelft.nl/islandora/object/uuid:fbbc321b-12c5-46df-9bf8-e3a491bb32ea?collection=education

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

Suzuki, T., Kitamura, M., Amano, Y., & Hashizume, T. (2010b, Oct). 6-dof localization for a mobile robot using outdoor 3d voxel maps. In *2010 ieee/rsj international conference on intelligent robots and systems* (p. 5737-5743). doi: 10.1109/IROS.2010.5652983

Thill, J.-C., Dao, T. H. D., & 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), 405-421. doi: 10.1016/j.jtrangeo.2010.11.007

Turner, E., Cheng, P., & Zakhor, A. (2015, April). Fast, automated, scalable generation of textured 3d models of indoor environments. *IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING*, *9*(3), 409-421. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6985553

Verbree, E., & van Oosterom, P. (2001). Scanline forced delaunay tens for surface representation. *INTERNATIONAL ARCHIVES OF PHOTOGRAMMETRY REMOTE SENSING AND SPATIAL INFORMATION SCIENCES 34.3/W4*, 45-52. Retrieved from http://www.isprs.org/proceedings/XXXIV/3-W4/pdf/Verbree.pdf

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

Yan, L., Liu, H., Tan, J., Li, Z., Xie, H., & Chen, C. (2016). Scan line based road marking extraction from mobile lidar point clouds. *Sensors*, *16*(6), 903. Retrieved from http://www.mdpi.com/1424-8220/16/6/903 doi: 10.3390/s16060903

Zlatanova, S., Liu, L., & Sithole, G. (2013). A conceptual framework of space subdivision for indoor navigation. *Publication of 3D GIS*. Retrieved from https://3d.bk.tudelft.nl/szlatanova/thesis/html/refer/ps/ISA_2013_SZ_LL_GS_final.pdf

Zlatanova, S., Liu, L., Sithole, G., Zhao, J., & Mortari, F. (2014). Space subdivision for indoor applications. *Delft University of Technology, OTB Research Institute for the Built Environment*. Retrieved from http://repository.tudelft.nl/islandora/object/uuid:c3ef4c87-9c35-4d05-8877-a074c3f7fdbf?collection=research

Appendices

A Reflection MSc thesis

Bart Staats 4162552 Navigating 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 inside the building of your destination. Outdoor navigation is well implemented and used a lot in daily life. Inside a building, it is impossible for GNSS signals to be received, which is the reason why a navigation aid is not widely available. This aid is not necessary in small, simple buildings but is useful in more complex buildings like hospitals, airports, conference venues and large shopping malls. 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 inside a large building. By introducing the method proposed in this MSc thesis, navigable maps of buildings which form one of the required features for indoor navigation, can be automatically created in a quick way with less to no constraints. This is important because the creation of a navigable maps based on 2D floor plans which are out of date is time consuming and costly. Installing an indoor navigation system with an up-to-date map can now be done more quickly. This lowers the costs of installing these systems in larger buildings like hospitals, airports and venue centres. Therefore, these systems will be more quickly available to be used by the public who visit these buildings.
The implementation can also be used to generate floor plans of buildings without a with a out of date map. Generating these maps of recently completed skyscrapers can help fire-fighters enormously during an emergency situation. It can also help real estate companies by generating accurately measurements about their different assets.

The developed method fits very well with the concept of the Master Geomatics which is based on collecting, processing, quality analysing and visualizing the data. All these elements are present in this MSc thesis. The data is collected in the form of a point cloud from at an indoor environment with a mobile laser scanner. The captured data is processed to detect dynamic elements, to classify the trajectory of the mobile laser scanning, to identify different navigable surfaces like slopes, stairs and horizontal surfaces, to correct wrongly classified surfaces and to subtract furniture objects which in the end results in the final navigable space. The different results are also visualized in ParaView. The quality analysis of the processed data is compared to a CAD model of the navigable space which results in a 90% accurate m^2 representation.

This MSc thesis covered a lot of different research fields. The identification of dynamic objects, stairs, slopes, flat surfaces, entryways, furniture objects and navigable spaces could all be the subject of a separate MSc thesis. Therefore, a lot of different research fields needed to be covered. It was possible to research all these fields because the trajectory of the mobile laser scanner contains a lot of information which was not yet used in other researches. Besides all this research, a paper was written which is accepted for the Indoor3D 2017 conference in Wuhan, China.
All these things costed more time than was planned in the original GANTT chart. Looking back at the planning of the MSc thesis, the report which was presented during the P2 should have been finished, but this was not the case. Since this was not finished in time, the work which needed to be done on the report was getting behind schedule and the GANTT needed to be changed all the time. Also the period between the P2 until the P3 and the P4 went by far more quickly than expected. This resulted in a lot of changes of the planning for the completion of this MSc thesis.

The final product has almost fulfilled all goals that were set at the beginning of the graduation period. Only the identification of the entryways and the classification check was unfortunately not finished in time. The proposed method was tested during the Geomatics Day 2017 by scanning the indoor environment before the venue started and processing the gathered point cloud which was showed in the CGI booth in the afternoon. This confirmed that the proposed method worked efficiently and well enough for the generation of a navigable voxel space.
Concluding, I think that there were too much things to investigate in this MSc thesis. If I could change things now, I would have chosen for less subject which needed to be investigated and would have worked more and for a lnger period of time on the specific elements of interest than I did in the end.

B Pseudo code trajectory classification

Algorithm 1: Stair and slope detection

```
for all points in trajectory
  if firsttime
    startNumber = xyz
  if next z higher than heightdifference startNumber
    save startNumber to possible-list
    startNumber = xyz
for all points in possible-list
  if firsttime
    prevPosition = xyz
  else
    get xyz as curPosition
  calc xy distance curPosition and prevPositionn
  if max distance > calc distance > min distance
    save to posibstair
    count track += 1
  else
    if count track > number of connecting elements
      save possiblestair to finalstair
      count track = 0
      posibstair = empty
    else:
      count track = 0
      posibstair = empty
write finalstair to database
repeat for slope
rest trajectory = flat
```

C Pseudo code region growing process

This is the method of the self implemented regiongrowing process.

Pseudo code region growing own implementation

```
seedid = 0
regionlist – list with final region
checkedvoxels – list with checked seedvoxels
needtobechecked – list with voxels that need to be checked

querydata – all voxels
querry seed – all seedvoxels

for seedvoxel in querry seed
  if seedvoxel not in checkedlist
    neighbours – get neighbours
    for each neighbor in neighbours
      if len neighbour >1
        if present in data
          add to needtobechecked

    if seeditem not in regionlist
      add to regionlist
    while exiting = true
      if needtobechecked is not empty
        checkingvalue = 1st item of needtobechecked
        if checkingvalue in datalist
          get upperneighbour
          if upperneighbour in datalist
            remove checkingvalue from datalist
            remove checkingvalue from needtobechecked
          else
            get neighbours
            for each neighbor in neighbours (same as earlier on)
              if len neighbour > 1
                if present in data
                  add to needtobechecked
            add checkingvalue to regionlist
            remove checkingvalue from datalist
            remove checkingvalue from needtobechecked
```

D Different trajectory classification sets

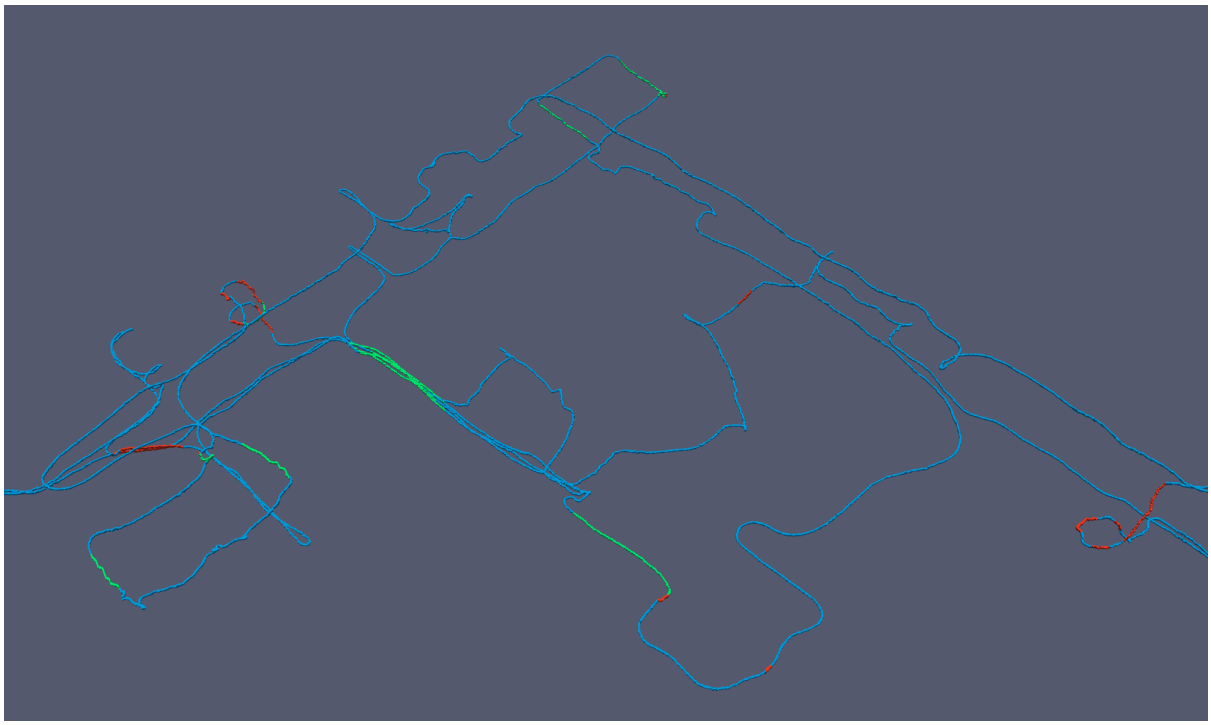


Figure D.1: Classification set 1

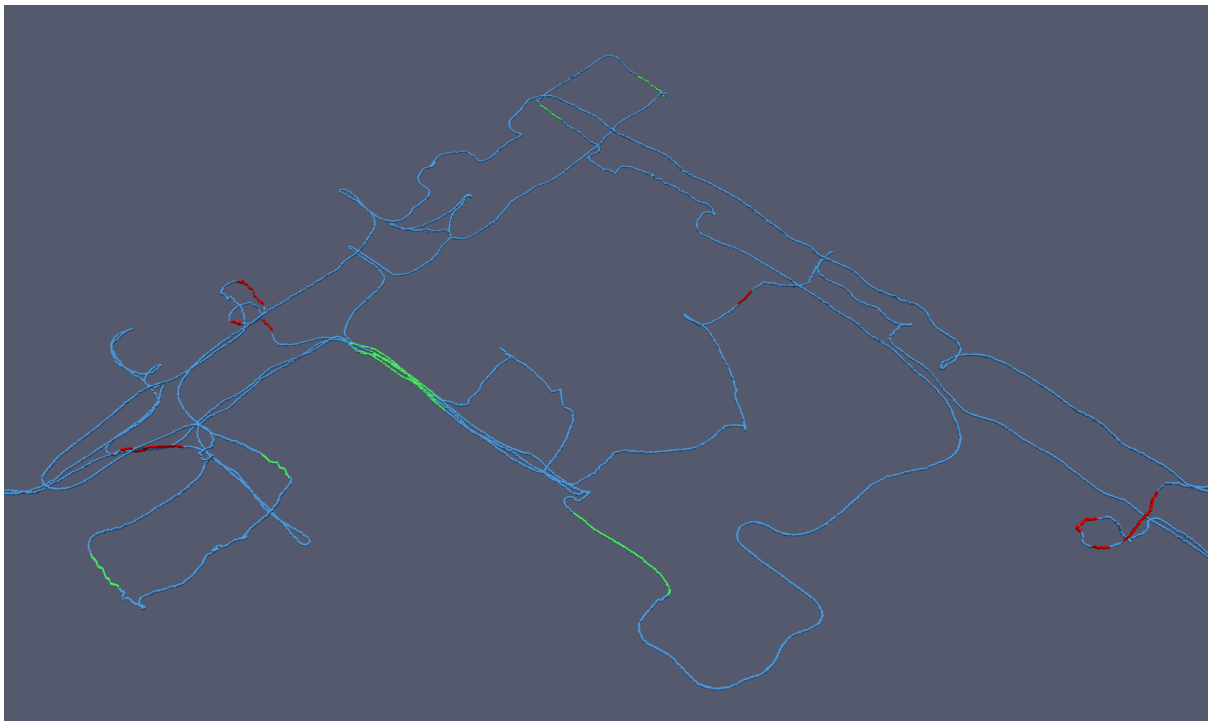


Figure D.2: Classification set 2

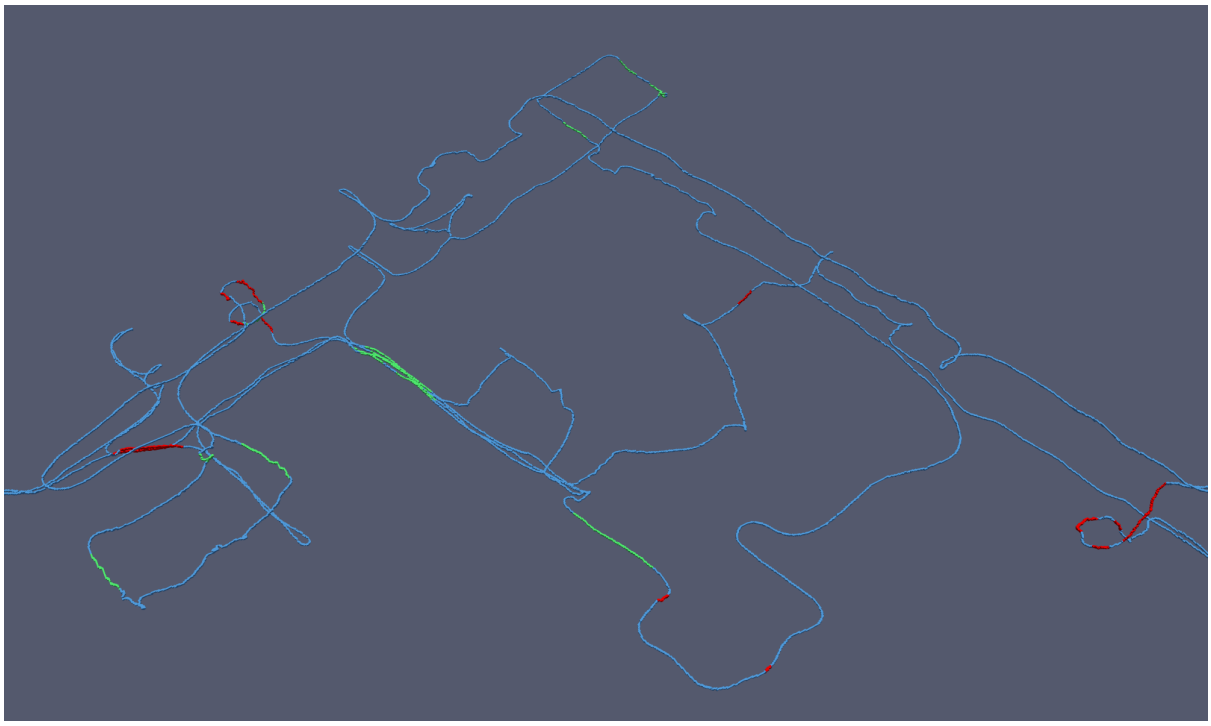


Figure D.3: Classification set 3