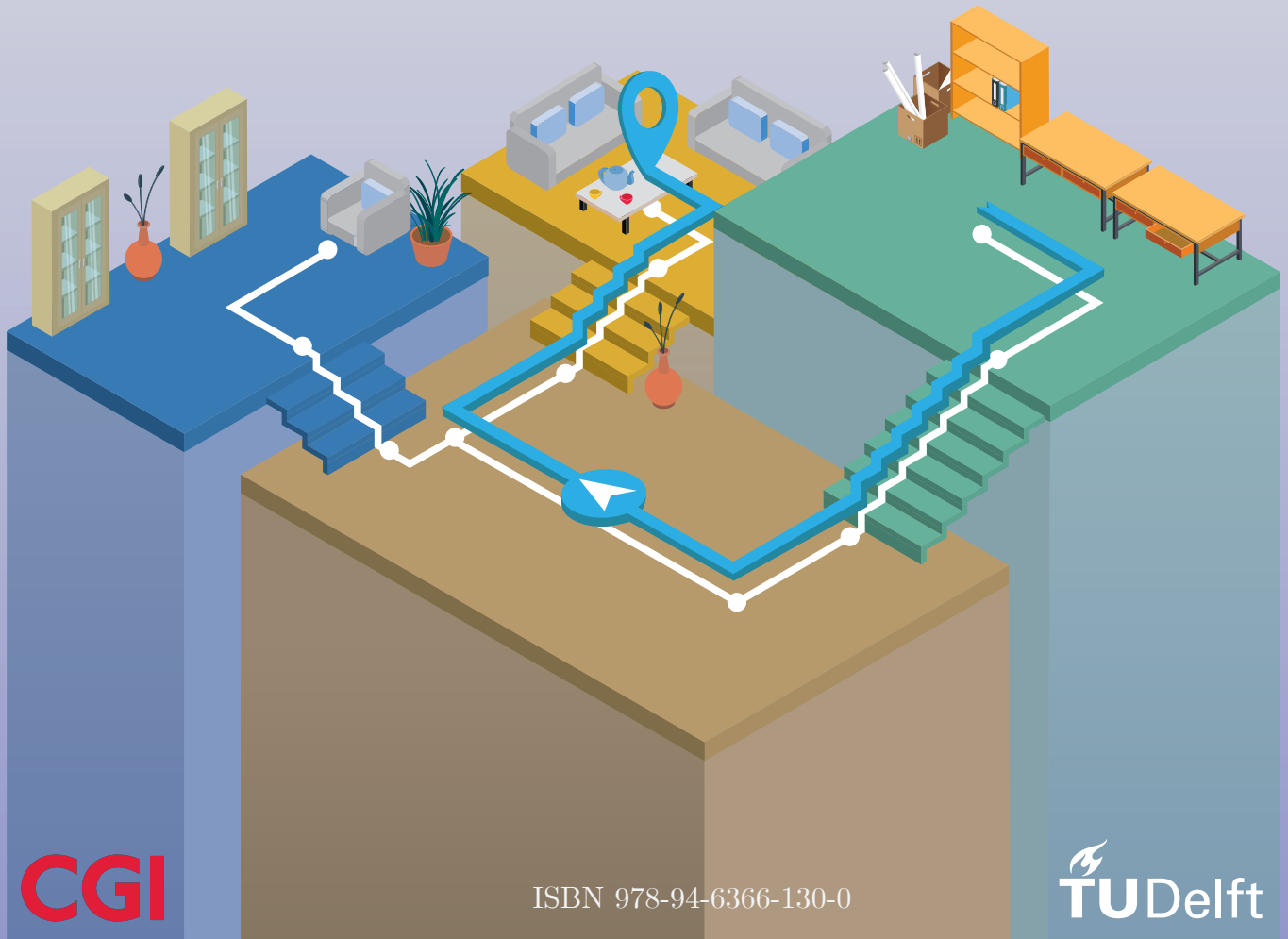
MSc. Thesis
Geomatics for the Built Environment

Automatic Extraction of an IndoorGML Navigation Graph from an Indoor Point Cloud

Puck Flikweert

January 2019



ISBN 978-94-6366-130-0

CGI

TU Delft

AUTOMATIC EXTRACTION OF AN INDOORGML NAVIGATION GRAPH FROM AN INDOOR POINT CLOUD

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Geomatics for the Built Environment

by

Puck Flikweert

January 2019

Puck Flikweert: *Automatic extraction of an IndoorGML navigation graph from an indoor point cloud* (2019)

ISBN 978–94–6366–130–0

The work in this thesis was made in the:

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

in cooperation with:

CGI The Netherlands
Transport, Post & Logistics

Supervisors:	Dr. Ravi Peters
	Dr. Lucía Díaz Vilariño
	ir. Robert Voûte (CGI Nederland)
Co-reader:	ir. E. Widyaningrum

ABSTRACT

Because of urbanisation, more than 50% of the global population now lives in cities, a number that is ever-increasing. This leads to a need for more dense and complex building constructions, driven by the fact that people are living increasingly close to one another. These structures are often so large that it becomes difficult to navigate in the environment without any assistance. Especially for people with restricted mobility, such as wheelchair users, or the blind or partially sighted, it can pose a challenge to find the best path in a building.

Consequently there is a need for navigation graphs that provide a way to structure connectivity information and routes in an indoor environment. Existing methods generally obtain information on the location of various rooms, corridors and their interconnectivity from 2D floor plans or 3D building models. However, these plans and models often were created by the architect when the building was planned, but never updated with new information after the building was built and came into use. Manually updating them is very labour-intensive. Therefore, in this research a method is developed for obtaining a navigation graph from an indoor environment by automatically extracting the information needed from a point cloud. The navigation graph is modelled according to the Open Geospatial Consortium (OGC) standard IndoorGML, which provides a basic structure for indoor navigation. The point clouds used in this research are gathered with a hand-held Mobile Laser Scanner (MLS), which also saves the path (trajectory) taken in the building.

This all leads to the main research question: *How can a navigation network in IndoorGML format automatically be extracted from a cluttered indoor point cloud and its trajectory?*
In order to answer this question this research focuses on the detection of doorways, and how they connect indoor spaces, such as rooms and corridors. A new way of door detection is proposed, which is based on the identification of walls using the 3D Medial Axis Transform (MAT) of the point cloud. After this it is explored how the established connectivity relationships can be extended with accessibility information, such as the location of stairs, and the dimensions of doors. Additionally it is researched how a more detailed navigation graph can be created by subdividing large spaces that have multiple connections.

In this thesis it is proven that geometric input for a navigation graph can automatically be obtained from a point cloud. 100% of doors could be detected in the dataset that the methods were developed on, and when testing the methods on another point cloud, this number decreased to 70% , due to unforeseen situations. All spaces connected by detected doors, stairways and sloped surfaces were included in the graph, from which a more extensive navigation graph could be produced by subdividing corridors. This final product was compared to networks drawn by humans on a corresponding floor plan, from which it can be concluded that the graph produced by the methodology of this thesis is a good basis for a navigation.

ACKNOWLEDGEMENTS

First of all I would like to express my gratitude to my supervisors for guiding me through this research. Many thanks to my first mentor Ravi Peters, who provided me with plenty of feedback, programming support, and always had a quick response to my questions. I would also like to thank my second mentor Lucía Díaz Vilariño for figuratively, but also literally, going to great lengths to support me by showing enthusiasm and interest in my research. And Robert Voûte, thank you for your positive energy, constant input and for motivating me to always take the extra step. I also want to thank my co-reader Elyta Widyaningrum for providing additional feedback on the research.

I would also like to thank all of the other tutors involved in the Geomatics programme for teaching us all they know, and making the lectures and projects challenging and engaging. Besides this, I want to thank my colleagues at the department of Design Informatics for teaching me even more about spatial algorithms, building modelling and architecture, and giving me input for my research.

In addition, I am grateful for all the support given by everyone at CGI The Netherlands, who provided me with a great environment to develop this research in. I would especially like to credit Bart Staats for making me acquainted with the field of indoor modelling, and offering help in developing some of the methods. Moreover, many thanks to all my fellow trainees for your daily support and sometimes necessary distractions.

A special thanks to Kotryna Valečkaitė for creating the front page of this thesis. Furthermore many thanks to Matthijs Theelen for adding the final touches, and supporting me in many other ways during the final months of my thesis.

Last, but not least, I would like to express my gratitude to my friends and family who supported me during my years at TU Delft. Especially many thanks to my parents, providing me with endless encouragement and comfort, to the friends who are always there for me, and to my fellow Geomatics students, for challenging me and making the past years a few of the most enjoyable of my life.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Algorithms

ACRONYMS

BIM Building Information Modeling

BLE Bluetooth Low Energy

CRS Coordinate Reference System

GHT Generalized Hough Transforms

GML Geography Markup Language

GPS Global Positioning System

IMU Inertial Measurement Unit

ISPRS International Society for Photogrammetry and Remote Sensing

***k*-NN** *k*-Nearest Neighbours

MAT Medial Axis Transform

MLS Mobile Laser Scanner

NRG Node-Relation Graph

OGC Open Geospatial Consortium

PCL Point Cloud Library

RFID Radio-frequency identification

S-MAT Straight-Medial Axis Transformation

SLAM Simultaneous Localization and Mapping

SSM Structured Space Model

UML Unified Modeling Language

UWB Ultra-wideband

VDN Variable Density Network

WKB Well-Known Binary

WKT Well-Known Text

WLAN Wireless Local Area Network

1 INTRODUCTION

Availability of outdoor navigation applications has seen an enormous increase in the past 20 years, among others due to the deactivation of the Selective Availability act on Global Positioning System (GPS) in May 2000. This discontinuation of adding intentional errors to the civilian GPS signal lead to higher positioning accuracy [Adrados et al., 2002]. Together with the explosive rise in the number of people owning a mobile computing device, it paved the way for more widely available outdoor navigation aids. However, seeing that a GPS receiver needs an unobstructed line of sight to at least four satellites, this positioning technique can almost never be used in an indoor environment, which initially resulted in less significant progress in the field of indoor navigation. But, with the development of indoor positioning systems, using for example Bluetooth Low Energy (BLE) beacons, Wireless Local Area Network (WLAN), Ultra-wideband (UWB) and Radio-frequency identification (RFID) tags [Deak et al., 2012], the field is rapidly catching up.

Besides needing a way to localise the object or person to be navigated, there are other requirements for navigation [Nakagawa and Nozaki, 2018], including: a map or model of the environment that is traversed; a graph or network model representing navigable routes in the environment, a way for the person to be navigated to interpret the path (wayfinding); and, a path-finding algorithm calculating the optimal path. The focus of this thesis lies with the second: automatically generating a network graph of an indoor environment. A fundamental requirement for this is to have up-to-date detailed information on how the environment looks like, in a format that can be used in an automated process. For these purposes, point clouds are an ideal fit, because they can be quickly gathered from an indoor environment, making use of either static or dynamic laser scanners.

1.1 USE OF INDOOR POINT CLOUDS

With the development of hand-held 3D laser scanners, it has become easier to generate point clouds of indoor environments. Various authors acknowledge the need for indoor point clouds to assess the way a building was built, versus the way a building was planned [Bosché et al., 2014; Alattas et al., 2017; Hong et al., 2015; Volk et al., 2014; Staats et al., 2017]. This is because recent indoor point clouds contain up-to-date information on how the interior of a building looks like, as opposed to a floor plan or 3D model that was created during the planning phase. Point clouds can be obtained using static or dynamic laser scanners, the advantage of the first being that the data is often more accurate, but of the latter that it is faster in use and less occlusion will take place, because one can also walk behind occluding objects. Dynamic laser scanners are also called Mobile Laser Scanner (MLS) and have as an additional benefit that some of them also include the path that was walked with the scanner, which can be used in extracting useful information from the point cloud [Díaz-Vilariño et al., 2017; Staats et al., 2017].

Point clouds do not only include information about the location of structural elements, such as walls, floors, ceilings and stairs, but also about the placement of obstacles. Therefore this means that they contain information on empty or free space in an indoor environment [Broersen et al., 2015], in which can be navigated. However, a point cloud can easily contain millions of points, taking up gigabytes

of memory [Schnabel et al., 2007]. Although a human can detect various elements in a point cloud by looking at it, to a computer it is nothing more than millions of unrelated xyz coordinates. Hence, there is need for point cloud processing algorithms, with which a computer can automatically find and reconstruct elements in the indoor environment.

1.2 INDOOR NAVIGATION NETWORKS

Besides using indoor point cloud data for the creation of 3D geometric models, it can also be used in generating a navigation network model of the environment. As well as knowledge about structural elements in a building, network models require information on different spaces and connectivity relationships between them.

With an increasing number of people living in densely populating areas around the world, buildings are becoming larger and more complex. Kwan and Lee [2005], Brown et al. [2013], Rueppel and Stuebbe [2008] and Boguslawski et al. [2016] draw attention to the need of a 3D network model for indoor navigation, the latter stressing the importance of routing networks for complex buildings in disaster scenarios and emergency response. Navigation networks can especially be helpful to people restricted in their mobility, such as wheelchair users, or blind or partially sighted people, who cannot always follow traditional route signs [Mirza et al., 2012].

Sithole [2018] and Nakagawa and Nozaki [2018] identify different kinds of indoor navigation networks, which can be summarised as:

1. graph structures linking spaces (e.g. rooms and corridors) in the indoor environment;
2. grid models based on the tessellation of indoor space in cells or voxels;
3. 'continuous' potential field methods, that have the starting point serve as repelling force, and the ending point as attracting force.

Potential field methods [Koren and Borenstein, 1991; Ge and Cui, 2002] are used most often with autonomous drones and robots, having walls and obstacles serve as repelling forces to prevent collision. As the focus of this thesis is on pedestrian navigation, and processing costs of potential field methods are high, they are not considered in this research.

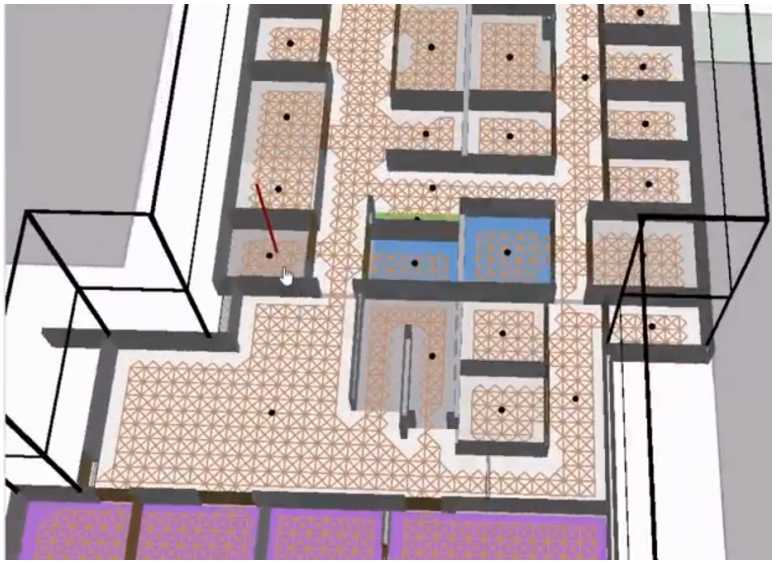


Figure 1.1: Example of the lattice model (brown lines) created in Esri Campus Viewer (screenshot taken from `https://youtu.be/Gb1pk0Xzyhg`)

There are existing methods that create grid model navigation networks from indoor point clouds, usually based on pixels (2D grid cells), or on 3D volumetric pixels, called voxels. Example of this are implemented by Nakagawa and Nozaki [2018], who project a point cloud in a 2D square grid, detecting the location of walls and objects based on density and height of points in a cell, and Wang et al. [2018], who create a 2D hexagonal grid model from a floor plan. Mahdaviparsa and McCuen [2018] create a network model for emergency response by incorporating 3D models of buildings in the Esri Campus Viewer Tool to make floor plans. Using the tool all floors are covered by a continuous lattice (see Figure 1.1), which are used as a dense navigation network. Different floors can be manually linked in the Campus Viewer by adding vertical lines at locations of stairs and elevators.

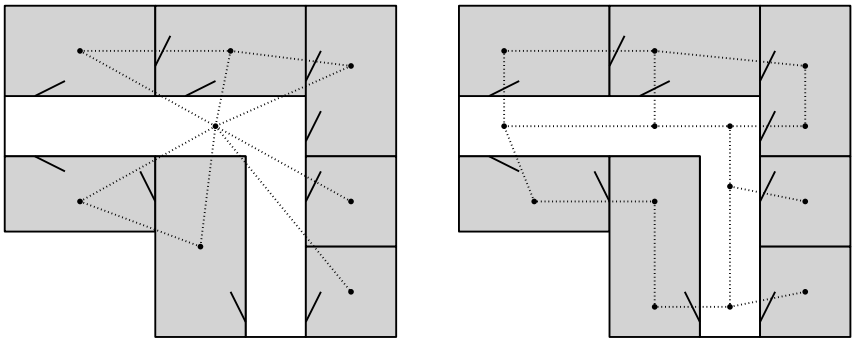
Staats et al. [2017] and Staats et al. [2018] create a voxelised point cloud and classify voxels that are 'walkable' (floor, stairs, ramps), and Fichtner et al. [2018] create an octree structure in the free space of a point cloud, defining connectivity between neighbouring free cells. However, the networks generated are usually very dense. For example in the lattice graph of Mahdaviparsa and McCuen [2018] a room can be filled with hundreds of nodes, which are the locations where the lattice has junctions. Pathfinding algorithms such as breadth-first and depth-first search run in an order of time $O(|V| + |E|)$, where $|V|$ corresponds to number of vertices, and $|E|$ to number of edges [Cormen et al., 2009] in a navigation graph. This proves that a less dense navigation network would lead to faster pathfinding; a useful trait in for instance emergency response.

1.3 INDOORGML

For this reason, this research will focus on the first type of navigation network: a graph structure linking indoor spaces. In 2014 the Open Geospatial Consortium (OGC) published IndoorGML, a standard providing a framework for this type of indoor navigation network [OGC, 2014]. IndoorGML is an application schema of the Geography Markup Language (GML), focussing on semantics, geometry and topology of indoor environments.

However, IndoorGML does not extensively cover geometric and semantic properties, to prevent too much overlap with other indoor building modelling standards such as CityGML and Building Information Modeling (BIM). Networks in IndoorGML are based on Poincaré duality [Munkres, 1984]. Distinction is made between primal space and dual space. The primal space defines the separate spaces in an indoor environment, such as rooms and corridors, also called 'cells' or 'states'. The dual space defines a node in every cell, and edges between nodes that have a relationship ('transitions'). Relationships in IndoorGML can be described as adjacency, connectivity or accessibility, and are all three separately saved as a Node-Relation Graph (NRG). Adjacency occurs when two cells share a wall or another type of border. An example of a connectivity relationship between nodes is when they share a wall with a door, or when they are connected by stairs. Accessibility is described using geometric information, such as whether a wheelchair user can traverse a certain transition, which is not possible if there are stairs.

IndoorGML provides support for the decomposition of large indoor spaces into subspaces (called *subspacing* or *subdivision*), which is very useful for the definition of a more complete navigation network. When subdividing indoor spaces into smaller cells a better assumption for indoor distances can be made, allowing for better route calculation and visualisation. An example of a hallway where subspacing would be desirable, is shown in Figure 1.2. Without subspacing the route is drawn through walls (Figure 1.2a), but when the hallway is subspaced (Figure 1.2b) the navigation graph becomes more representative of real paths a person would take. A more detailed explanation of IndoorGML is given in Section 2.1.



(a) Example of a Connectivity NRG for a hallway and rooms

(b) Subspaced Connectivity NRG for a hallway and rooms

Figure 1.2: Connectivity NRGs for an floor plan

1.4 THESIS OVERVIEW

In conclusion, a graph-based indoor navigation network is a valuable asset for fast route calculation, which is necessary for emergency response, and is also useful in the navigation of people restricted in their mobility. Up-to-date point clouds are a suitable basis for indoor navigation networks, provided that geometric information and connectivity relationships can be extracted from them. Moreover, point clouds can be generated in every kind of indoor environment, making them usable in many situations, especially when obtained with a MLS. By constructing the network as an IndoorGML model, it will be less dense than the network of a grid-based model, and support is given for a fast calculation of routes through the indoor environment. However, methods that automatically construct an IndoorGML model from an indoor point cloud are not extensively developed or researched yet, which provides the motivation behind this thesis.

Therefore a methodology is introduced to obtain input for an IndoorGML model by automatically detecting different indoor spaces, stairs, slopes and doors in a point cloud, and determining connectivity relationships between them. A new way of door detection is introduced, making use of the 3D Medial Axis Transform (MAT) (see Peters [2018]) to detect walls in a point cloud. Indoor spaces with multiple connections, such as corridors, are subspaced by analysing the 2D floor geometry of the space, modelled as a polygon. Its 2D MAT is created, forming the skeleton of the polygon, and on this skeleton new nodes are created at doors and junctions in the space. Besides this accessibility information, such as the location of stairs and the dimensions of doors, are retrieved from the point cloud and added to the model.

1.5 RESEARCH QUESTIONS

Based on the problem statement in the introduction the main research question is defined as:

How can a navigation network in IndoorGML format automatically be extracted from a cluttered indoor point cloud and its trajectory?

In this context a cluttered indoor point cloud is a point cloud with dynamic and static objects, such as people and furniture, blocking construction elements of the building. The end product of this graduation thesis is an IndoorGML model that is automatically generated from an indoor point cloud.

The following sub-questions to make the aims more specific:

1. What information can be extracted from a point cloud in order to create an IndoorGML **connectivity** and **accessibility** Node-Relation Graph (NRG)?
2. In what way can **connecting elements** (e.g. doorways and entries) between two indoor spaces be detected?
3. In what way can separate **indoor spaces** (e.g. rooms and corridors) be detected, and how can these, in combination with the **connecting elements**, be used in modelling the connectivity NRG?
4. **In what circumstances** should indoor spaces in the navigation network be subdivided into **subspaces**, and **how** can this be done in a way that is efficient in terms of size of the network?
5. How **suitable** is IndoorGML as a **data model** for mapping a navigation network from an indoor point cloud?

1.6 RESEARCH SCOPE AND GOAL

The focus of this thesis will be on the generation of a connectivity NRG for IndoorGML using indoor point clouds gathered in a dynamic way with a hand-held MLS, also making use of the trajectory of the scanner. Extension of the connectivity NRG to an accessibility NRG is researched, based on geometric properties in the point cloud. The other type of graph, adjacency NRG, will not be considered in this research, because it is the least relevant for navigation purposes. Moreover 3D geometry modelling of indoor space will only be done as far as deemed necessary for the IndoorGML model. Multiple floors of buildings will be considered in the method, but the use case of the topic concerns human navigation, so only navigable floor space will be regarded. For the subdivision of certain cells in the graph, only geometric or topologic characteristics will be taken into account. Semantic information, such as the location of important objects and points of interest in the environment, cannot always be extracted from an indoor point cloud.

This research is a proof of concept; an exploration of whether it is possible to directly obtain an IndoorGML-structured navigation graph from an indoor point cloud. It does not consider the search for the fastest or best method to accomplish the results. However, the focus will be on creating a generic method that can work in different types of buildings. This excludes for instance Manhattan world solutions, which can only be applied on buildings that fall in a rectangular grid. Finally the application of the IndoorGML model in a navigation system will not be researched, as it is out of scope of the problem statement.

1.7 READING GUIDE

The theoretical background needed to understand important concepts in this thesis are discussed in Chapter 2. Related work on the research question and subquestions is presented in Chapter 3. In Chapter 4 the methodology developed to answer the research questions is explained, followed by the implementation details in Chapter 5. Results are shown and discussed in Chapter 6, after which answers to the research questions and conclusions are debated in Chapter 7.

2 THEORETICAL BACKGROUND

In this chapter theories that are used in the methodology of this thesis are explained. First of all IndoorGML is described, going into detail on the specifics of the standard (see Section 2.1). The information that can be used as an input for an IndoorGML file is obtained by analysing an indoor point cloud. A large part of the analysis is done after voxelisation of the point cloud, which is explained in Section 2.2.1. Stairs, floors and slopes are identified from these voxels with the method described in Section 2.2.2. Voxels representing separate rooms and corridors are modelled as polygons using α-shapes, explained in Section 2.3.1. These polygons are simplified with the Douglas-Peucker algorithm (Section 2.3.2). Finally, in this thesis the 2D medial axis is used to get the centrelines of hallway polygons, whereas the 3D version is applied on the point cloud to obtain information on the location of walls. The concepts behind this are described in Section 2.4.

2.1 INDOORGML

IndoorGML [OGC, 2014] is a standard developed by the Open Geospatial Consortium (OGC) for the exchange of indoor spatial information, focused especially on indoor navigation. It aims to provide support in location based services and indoor route analysis, and is based on the notion that an indoor environment can be divided into organisational or structural cells, which represent for instance rooms or corridors. These cells are non-overlapping and each have a unique identifier. Relationships between cells can be represented by the Node-Relation Graph (NRG), of which there are three types: adjacency NRG, connectivity NRG and accessibility NRG (Figure 2.1). In the latter more user specific information can be saved, such as the width of a door, or whether a certain edge is traversable by a wheelchair user. Relationships in IndoorGML are based on Poincaré duality [Munkres, 1984], in which the cells are regarded as primal space, and connections between cells as the dual space. The NRG represents this dual space, and this graph (V, E) consists of nodes (V), representing the cells (*States*), and edges (E) indicating the relationships (*Transitions*).

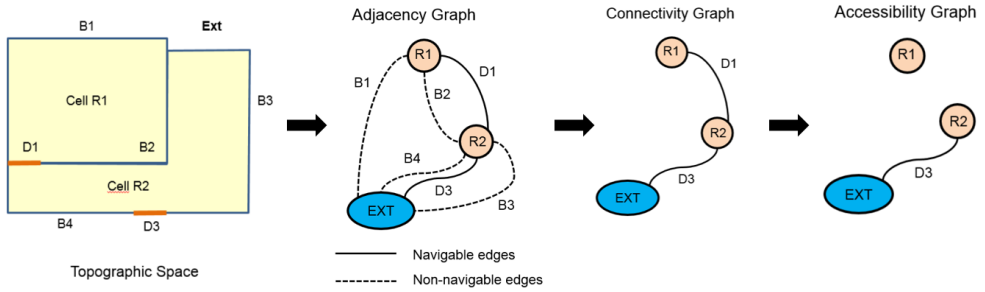


Figure 2.1: The different NRGs in IndoorGML [OGC, 2014]

There are two different types of representations of cellular space in IndoorGML. The first being the thin wall model, used in Figure 2.1, in which rooms and corridors are represented by nodes, and doors and walls as edges. The other type is the thick wall model (see Figure 2.2), in which besides rooms and corridors also doors and

walls are represented by nodes, and their relationships by edges. This is a more extensive way of modelling the indoor environment and, because of the inclusion of doors, can also be more useful for indoor navigation purposes. Information on the location and adjacency of walls is less important for navigation, because it is more useful to know which nodes **can** be traversed, instead of those that **cannot**, which is also represented in the connectivity NRG.

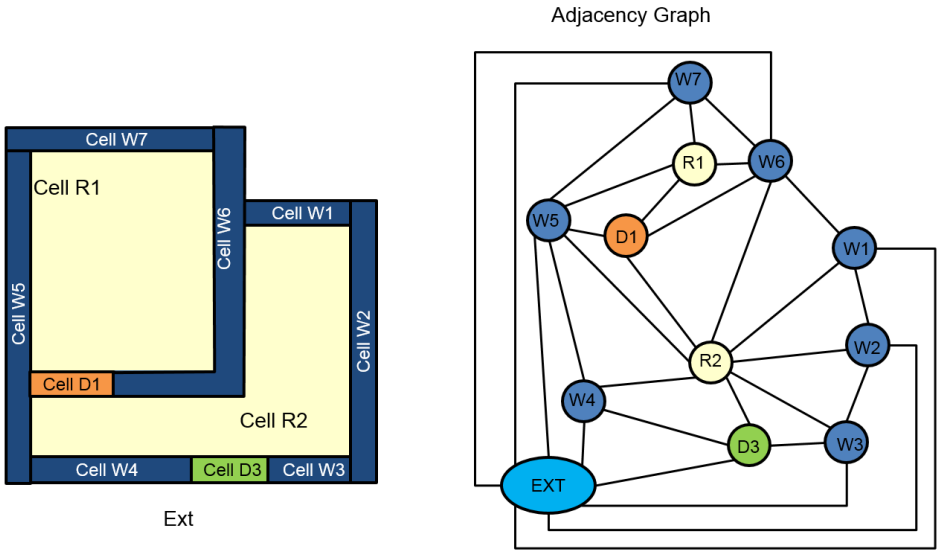


Figure 2.2: The thick wall model in IndoorGML [OGC, 2014]

In IndoorGML the inclusion of geometry is not compulsory. A valid IndoorGML model can be a logical NRG, meaning that there is only topological information present, and no information about the location of cells, or their geometry. However, for navigation purposes it is more convenient to include geometry in both the primal (cell) space, as 2- or 3D geometric information about the cell, and in the dual (graph) space, as location of a node. This idea is represented in the Structured Space Model (SSM), as shown in Figure 2.3.

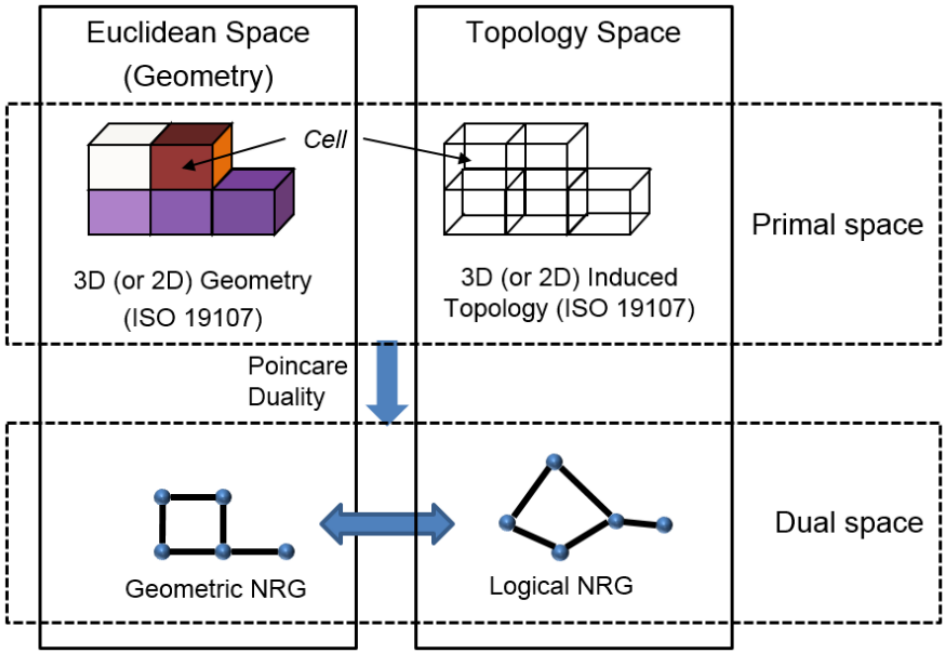


Figure 2.3: SSM as defined in IndoorGML [OGC, 2014]

In practice IndoorGML has been implemented by various authors, including a method to go from CityGML to IndoorGML [Kim et al., 2014], and from Building Information Modeling (BIM) to IndoorGML [Teo and Yu, 2017]. A general study on the implementation of IndoorGML is described by Kang and Li [2017], who describe criteria for when to partition a cell into subspaces, such as when a space contains two different functions (e.g. kitchen and living room), is 'big', like a hallway or convention hall, and when obstacles may limit movement in the space. They also describe the concepts of horizontal and vertical distance, and how different building floors can be connected. No related work has been found on automatically obtaining an IndoorGML model from a point cloud, at the time of writing this thesis. An alternative to IndoorGML would be a Chinese standard, called IndoorLocationGML [Zhu et al., 2016], which is more concerned with the relative and absolute location of objects in the indoor environment and indoor distances, and not so much with semantics and accessibility of places. Because IndoorGML provides a better opportunity to save information on the type of node (stairs, corridors, doors, rooms), which gives more information to users on the accessibility of a place, it has been decided not to work with IndoorLocationGML.

2.2 VOXELISATION AND WALKABLE SPACE

Point clouds of buildings are usually large in size and unstructured, needing algorithms to process them, filter noise and obtain useful information from the data. A way to structure a point cloud for easier and faster processing is voxelisation (Section 2.2.1. From the voxels that are created in this process, the floors, slopes and stairs can be detected by the methods described in Section 2.2.2. These methods are used in this thesis as a basis for the navigation graph.

2.2.1 Voxelisation of point clouds

Voxels are defined as 'volumetric pixels' by Nourian et al. [2016], who present a computational method to get a voxelised model from a point cloud. A bounding box is defined for the extent of the point cloud, which is then divided by the voxel size in all three directions (x, y and z), finding out how many voxels fit in each direction. For each voxel it is determined whether it is filled with point cloud points, or empty. Only the filled voxels are saved in the data model. Noise filtering can be applied by setting a threshold for the minimal number of points to be in a voxel in order for it to be saved.

Broersen et al. [2015] obtain the voxelised model by using an octree structure (Figure 2.4), which is the three dimensional version of a quadtree. In this method the space is recursively divided into eight cubes, called octants or leaves, until certain boundary conditions are fulfilled. These boundary conditions can for instance be the smallest defined leaf size, number of recursions, or a certain density of points in a leaf. In the case of Broersen et al. [2015] the maximum number of octant divisions can be defined by the user. Each voxel in the octree gets assigned a locational code, based on a space filling curve. This code can be used for a fast lookup of neighbouring voxels, or parent nodes in the octree. Instead of looking at which leaves are filled, Broersen et al. [2015] look at the leaves that are without points. These leaves can be regarded as empty, and thus they are traversable for navigation. This idea can especially be applied for drone routing, because drones are not in need of a floor surface directly below them.

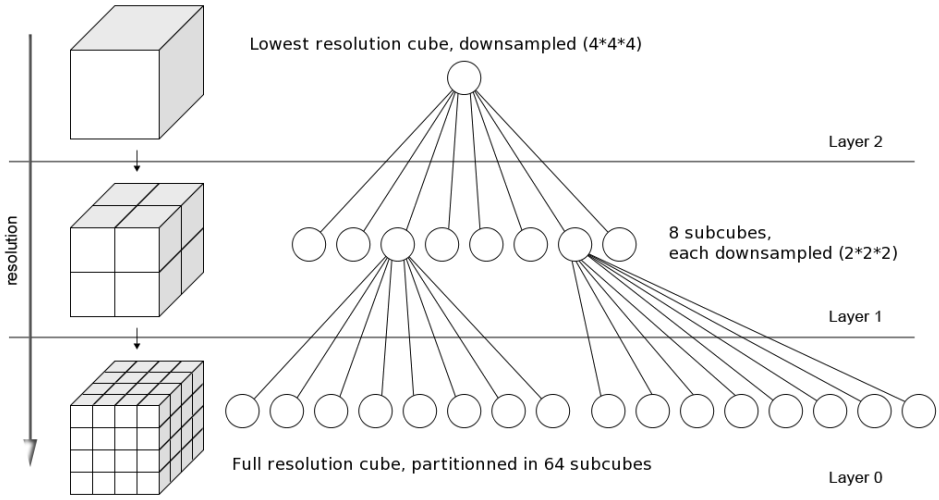


Figure 2.4: Example of an octree structure [Lobos et al., 2010]

2.2.2 Obtaining the walkable voxel space

Staats [2017] developed a method to find all walkable voxels in a voxelised point cloud, and automatically classified them as stairs, ramps or regular floor space. The voxelisation method used is the one based on an octree [Broersen et al., 2015], in which the smallest leaves are used as voxels to do processing on. The point clouds researched are gathered with a ZEB-REVO laser scanner, which is a hand-held Mobile Laser Scanner (MLS) and can be operated by a walking person. The research makes use of the idea that the person gathering the point cloud was walking on a floor. Besides making use of the voxelised point cloud, also the trajectory of the scanner is voxelised. Voxels from the point cloud that lie directly underneath trajectory voxels are classified as seed voxels, because it can be said that these voxels have been walked upon. A region growing step is added to cluster voxels that lie in the same floor space. Based on the angle that the trajectory makes with the horizontal, stairs and ramps can be identified, because the steeper a person walks, the more likely they are to be walking on stairs.

In the clustering step the voxelised point cloud is sliced at every z (height) interval of the voxels, so the point cloud should be oriented correctly with regard to the horizontal. Every slice consists of all voxels at a certain height value, so can also be regarded as a layer of 2D pixels. When there are voxels present that were identified as floor space, according to the trajectory, their neighbours are identified as the same, and their neighbouring voxels as well. Neighbourhood in this case is based on 8-connectivity for pixels (see Figure 2.5). A schematic visualisation of all steps in this method are shown in Figure 2.6.

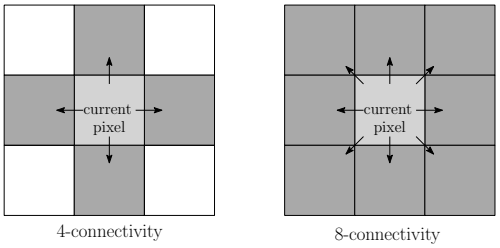


Figure 2.5: 4- versus 8- connectivity for pixels

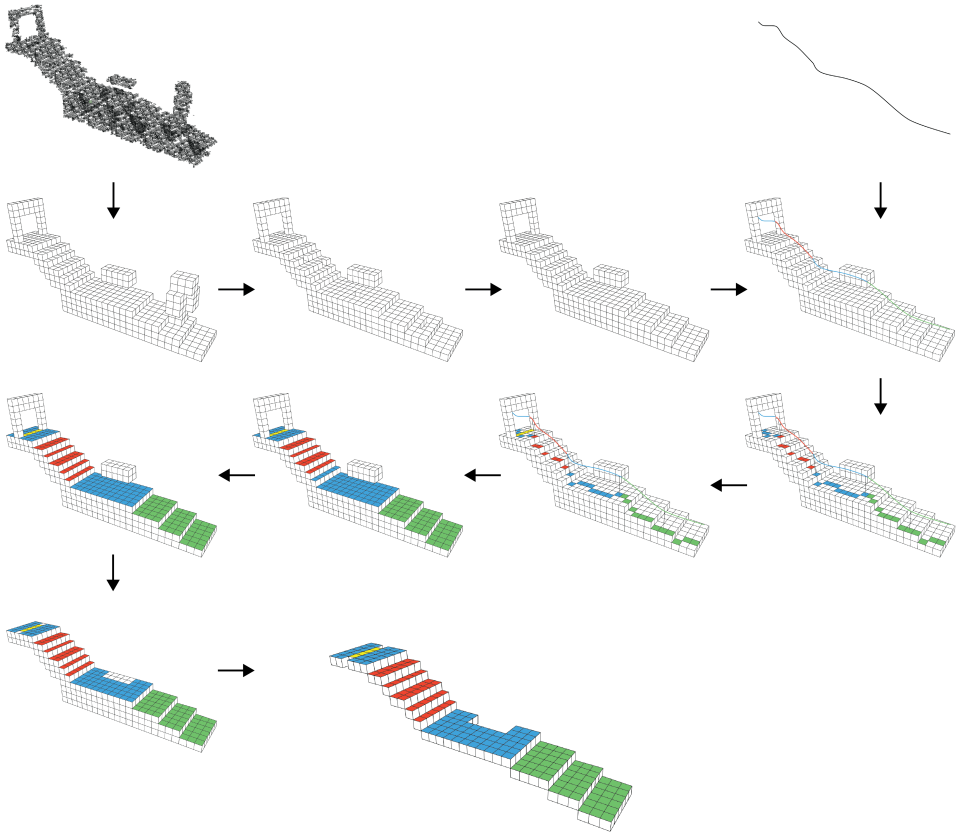


Figure 2.6: Obtaining the walkable voxel space from an indoor point cloud as proposed by Staats [2017]

2.3 2D POLYGON MODELLING

The walkable space, obtained with the methods described in Section 2.2, can be divided into different indoor cells (rooms, halls and corridors) after the locations of doors are found, assuming that a space always ends at a door, stair or slope. An indoor cell consists of all voxels belonging to that space, which eventually are made up of centre points and size of the voxels (see Figure 2.7 as an example of voxels belonging to one indoor space). These groups of voxels can be modelled as a polygon using α-shapes, as explained in Section 2.3.1. In order to simplify the polygons, which often create too detailed skeletons in the 2D Medial Axis Transform (MAT) (see Section 2.4.1), the Douglas-Peucker algorithm is applied (see Section 2.3.2).

Figure 2.7: Top view of all voxels belonging to one room

2.3.1 α-shapes

For the modelling the geometry an α-shape approach is used [Edelsbrunner et al., 1983], which is very suitable for modelling polygons around sets of points. The floor voxels that are defined for each indoor space can basically be considered as sets of 2D x,y-coordinates, for now called a set of points, for which the α-shape can be defined. The shape of the output polygon is dependent on the parameter α (Figure 2.8). Generally it can be said that the smaller the value for α, the smaller the area of the polygon created. When $\alpha \to 0$ the shape is just the input point set and when $\alpha \to \infty$ the output will be the convex hull of the input points.

The first step in this algorithm is to apply Delaunay triangulation [Delaunay, 1934] for all points. Then, for each Delaunay triangle, it is tested if the radius of its circumcircle, r, is smaller than α. If this is true, the Delaunay triangle is added to the set for the output polygon.

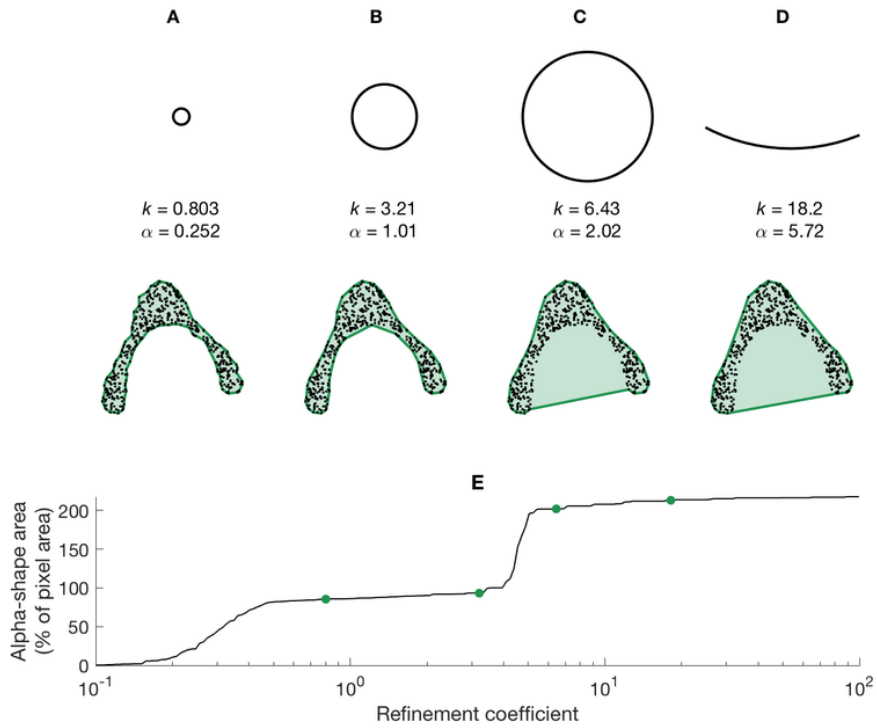


Figure 2.8: Different values for α resulting in different polygons [Gardiner et al., 2018]

2.3.2 Douglas-Peucker

The Douglas-Peucker algorithm [Douglas and Peucker, 1973] can be used to downsample linestrings and in this way simplify their geometry. This can in turn be applied on a polygon, which is basically a collection of line segments that make up a closed curve. The algorithm is visualised in Figure 2.9. In step 1 the original linestring is shown, and threshold ϵ. A straight line is drawn between the first and the last point (AB), shown as dashed line a in step 2. The point furthest from this straight line is searched, and it is checked whether the distance from this point C to the closest point on the straight line is smaller than the threshold value ϵ. In step 2 it can be seen that $b > \epsilon$, so C is kept, and two new straight lines are drawn, AC and CB. In step 3, again, for both lines the furthest remaining points are searched, and their distances to the lines are checked. Distance $d > \epsilon$, so point E is kept, but distance $c < \epsilon$, so point D is not kept in the new linestring. The final results are shown in step 5.

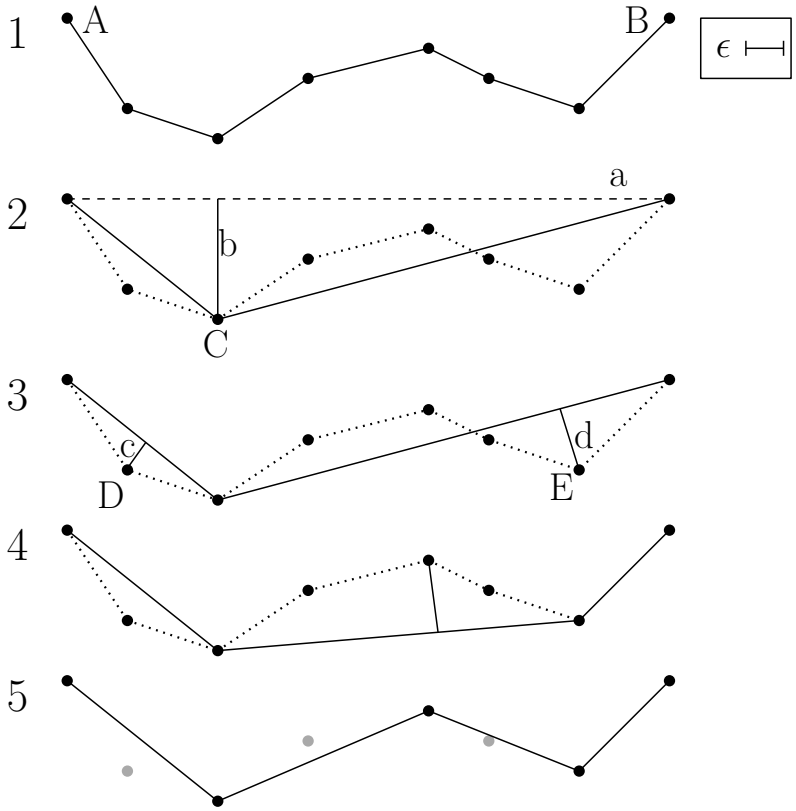


Figure 2.9: The steps of the Douglas-Peucker algorithm visualised for a linestring

2.4 MEDIAL AXIS TRANSFORM

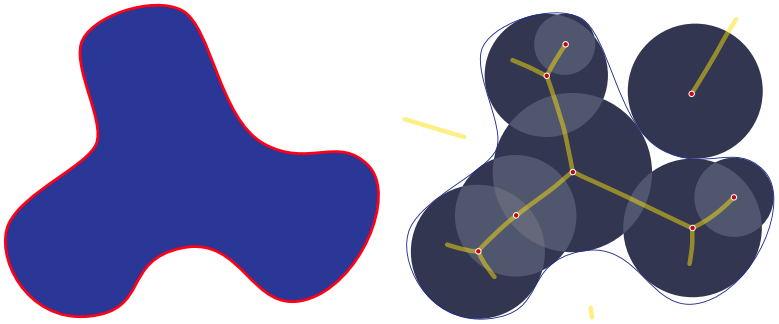
The Medial Axis Transform (MAT) was originally proposed by Blum [1967], who used it in the analysis of biological shapes. The 2D MAT for polygons is used in this thesis in the partitioning of indoor spaces in the connectivity NRG to subspaces (see Section 4.4) and is explained in Section 2.4.1. Moreover, the 3D MAT for point clouds [Peters, 2018] is used for a novel way of detecting walls in an indoor point cloud, as presented in Section 4.1, which is discussed in Section 2.4.2.

2.4.1 2D MAT for polygons

The medial axis of an object is made up of all points that have more than one closest point on the exterior (boundary) of the object. From these points circles can be drawn with a radius equal to the distance to the boundary, which are thus touching the object in two or more points. Together, the points and their corresponding circles make up the MAT, representing the skeleton of the object (see Figure 2.10).

2.4.2 3D MAT for point clouds

3D MAT
The MAT for a 3D object includes all points that have more than one closest point to the surface of the object. Instead of circles, balls can be drawn that touch these two or more points on the surface. These balls, together with their centres, are called medial balls. The circles in Figure 2.10 can be considered as cross-sections of 3D balls, as the general concept of the algorithm is similar in both 2- and 3D. In the method of Peters [2018] a 3D MAT is automatically generated for geographical point clouds. This method consists of two steps. At first a set of medial balls is calculated



(a) A shape

(b) Corresponding medial axis and some medial circles

Figure 2.10: 2D example of medial balls in a shape [Peters, 2018]

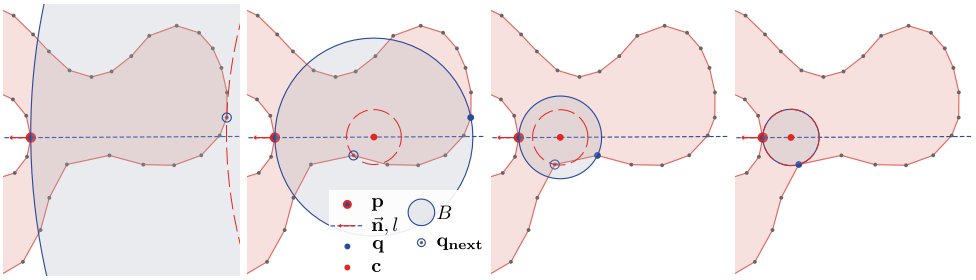
for the point cloud, using the ball-shrinking algorithm [Ma et al., 2012]. Secondly, by applying a region growing algorithm the medial balls are grouped into medial sheets. When applied on a geographical point cloud the results can be used for object recognition in landscapes, such as the identification of waterways [Broersen et al., 2017]. In this thesis the medial sheets are used for wall recognition in an indoor point cloud (see Section 4.1).

To begin with, medial balls have three main characteristics:

1. each medial ball touches two or more points of the original point cloud,
2. each medial ball is tangent to the surface wherever it touches points, and
3. there are no points of the original point cloud inside a medial ball.

The first characteristic is similar to the definition as discussed in Section 2.4.1, but instead of the ball touching the boundary of an object, it can touch two or more points in a point cloud.

According to the second characteristic a medial ball should be tangent to the local surface where it touches a point. This means that centre (c) of the medial ball should be aligned with the normal vector ($\vec{n}$) of the point. In a geographic point cloud normal vectors are usually not inherently present. Peters found the normal of each point p by using a KD-tree to find its k-Nearest Neighbours (k-NN), and fitting a plane through them. The vector orthogonal to this plane is the normal vector of the point. It can be concluded that c should be somewhere on infinite line L that goes through p and has direction $\vec{n}$.



(a) Initial ball

(b) Second iteration

(c) Third iteration

(d) Fourth iteration yields an empty ball

Figure 2.11: Ball shrinking iterations [Peters, 2018], legend in 2.11b

In order to comply to the third characteristic, the ball-shrinking algorithm is applied [Ma et al., 2012], for which the series of iterations is illustrated in Figure 2.11.

In the first step (Figure 2.11a) for point p in the point cloud the first medial ball centre c is computed, based on the location of p, $\vec{n}$ and initial ball radius r_i. Then the nearest neighbour for c in the original point cloud is computed, which is called point q. For the second iteration (Figure 2.11b), a new radius r and centre c are computed for the triplet p, $\vec{n}$ and q. This process is repeated until the new radius does not change compared to the previous iteration. When this happens the medial ball complies to the third characteristic, because the closest point to c is found to be q, meaning there are no points left inside the ball. When the ball-shrinking algorithm is applied on all points in the point cloud, the set of centres and corresponding radii (c,r) of all medial balls represents the MAT.

Medial sheets

In order to structure the set of medial balls created for a point cloud, they are grouped into medial sheets. These sheets are the 3D equivalent of the skeleton that can be seen in the 2D example in Figure 2.10.

To group medial balls together in sheets, we look at geometric characteristics of medial balls, as shown in Figure 2.12. In this Figure we can see the following characteristics:

- the radius, r: the distance from the centre of the ball to a point on the surface, either p or q;
- the vectors $\vec{cp}$ and $\vec{cq}$: vectors that are drawn between the centre of the medial ball and the points of the original point cloud that touch the medial ball;
- the separation angle, θ: the angle between vectors $\vec{cp}$ and $\vec{cq}$;

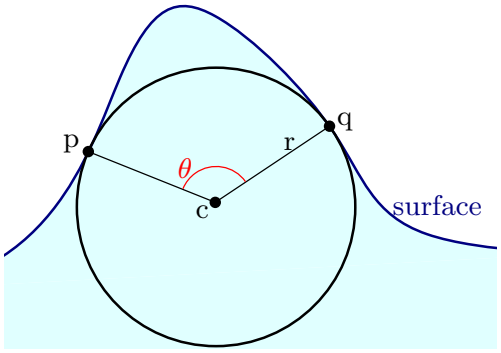


Figure 2.12: Geometric characteristics of a medial ball (2D side view)

Region growing is applied by Peters in order to get the medial sheets. A random point is selected from the MAT, and its neighbours are checked whether they have a similar value for θ, within a certain threshold. If a neighbouring point satisfies that condition, it is saved as part of a medial sheet, and the neighbouring points of that point are checked as well. This process continues until there are no neighbouring points left to check. In this way all MAT points are structured into medial sheets.

3 RELATED WORK

In this chapter an overview is given on related work to the topic of this thesis. It starts with a general overview about reconstruction from point clouds in the built environment (Section 3.1). Then a summary is given of work related to indoor navigation networks (Section 3.2). In Section 3.3 the detection of doors in point clouds is discussed, and finally in Section 3.4 the division of subspaces for indoor navigation networks is described.

3.1 OBJECT AND MODEL RECONSTRUCTION FROM POINT CLOUDS

Indoor point clouds are often gathered in two ways, either with a static or dynamic scanning device. Static laser scanners, such as terrestrial laser scanners, can be set up at one location at the time, and each time obtain parts of the point cloud, which have to be stitched together [Dold and Brenner, 2006]. This is called registration and can be done manually or automatically with software.

The point clouds used in this research are gathered by a dynamic laser scanner: the Mobile Laser Scanner (MLS) device ZEB-REVO, which uses a Simultaneous Localization and Mapping (SLAM) algorithm [Durrant-Whyte and Bailey, 2006] to process the output point cloud. SLAM detects the position of the scanner, and the change in position of certain objects that are automatically detected. Based on this the output point cloud is adjusted. The ZEB-REVO has an Inertial Measurement Unit (IMU) to detect movements of the device, but does not fully depend on this. To account for drift, shapes are detected in the point cloud, and in the final processing step the point cloud points are adjusted according to these shapes. Moreover, the locations of the MLS are saved as the trajectory, which provides useful information to the point cloud (see Section 3.3).

The advantage of static laser scanners is that they gather more accurate point clouds. However, because they have to be set up at a new spot in the environment every time, they are more labour-intensive. Generally there is also more noise present, because of occlusion by objects. With a MLS one can walk more easily behind an object and capture more of the environment, which is its biggest advantage besides obtaining the trajectory of the scanner.

In order to extend the IndoorGML model with information about indoor spaces, it is necessary to detect and reconstruct geometric elements, such as walls, floors, and furniture. The methods in this thesis especially focus on the detection of walls and floors, obtained from a voxelised point cloud. Besides being able to detect elements in a voxelised model, as described for floors, stairs and slopes in Section 2.2.2, many authors reconstruct elements from the points in the point cloud directly.

Hichri et al. [2013] give an overview of various techniques that are used in reconstructing a Building Information Modeling (BIM) from point clouds. They identify three steps: acquisition of the point cloud, segmentation of the point cloud, and BIM creation. Hichri et al. [2013] argue that segmentation can be done based on similarity and closeness of points, region growing with normal vectors of points, or distance between planar faces. From the segmented point cloud objects can be modelled, and attributes can be assigned to them based on the semantics detected. Tang et al. [2010] also give an overview modelling BIM from point clouds, but define

a step of 'data preprocessing', in which unwanted data is filtered, and surfaces are estimated from the point cloud. They define BIM modelling not only as geometry modelling, but also categorising objects and relationships between objects.

A well-known method for shape detection and reconstruction in point clouds is RANSAC [Schnabel et al., 2007], which is for instance used for wall surface reconstruction by Quirk et al. [2006], and wall and floor plane detection in point clouds by Thomson and Boehm [2015]. By random sampling of points in a point cloud it can detect shapes like planes, spheres, cylinders, and more. This method is also included in the Point Cloud Library (PCL) [Rusu and Cousins, 2011], an open project focused on the processing of point clouds, in order to detect and reconstruct elements. The PCL contains more algorithms that can help structure a point cloud, speeding up other processes. Examples of these are normal vector estimation, and kd-tree construction. It also includes algorithms for noise filtering, surface reconstruction, and more. They can be used for reconstruction of objects in an indoor environment [Rusu et al., 2008].

Pu and Vosselman [2009] apply another way of shape detection in point clouds, in which first the point cloud is segmented, using surface growing algorithm. This algorithm is based on region growing of points in the point cloud, starting from a seed point, and checking whether its neighbours are on the same plane (or other predefined shape) defined from the point and its normal vector, and have a small difference in normal vector. After all points are clustered, surface is modelled, usually as a convex or concave polygon (see Figure 3.1).

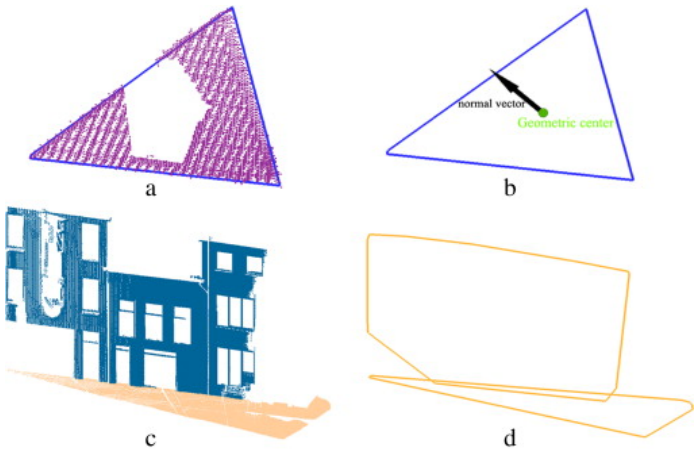


Figure 3.1: Modelling a convex hull for segmented point cloud parts [Pu and Vosselman, 2009]

A disadvantage of both the RANSAC and planar surface growing methods is that the shape of the surface to be detected needs to be known. In case of walls that are curved instead of planar this needs to be manually defined beforehand, and adjusted in the threshold. Also the convex hull modelling can be inaccurate, as can be seen in Figure 3.1d. Moreover the methods are both very reliant on thresholds, such as the distance of a point to the surface, and difference in normal vectors.

A benchmark test for indoor modelling was developed by the International Society for Photogrammetry and Remote Sensing (ISPRS), also freely providing indoor point clouds to researchers to test their methods on [Khoshelham et al., 2017]. In the benchmark three elements of an indoor model are evaluated: the geometric reconstruction, the semantics (type, function or material) of objects, and topology of spaces (connectivity and adjacency). Khoshelham et al. [2017] note that most researchers now focus on the first: geometric reconstruction, but recommend researchers to also put more focus on the last two elements.

3.2 INDOOR NAVIGATION NETWORKS

Duality is a reoccurring theme in graph-based indoor navigation networks. Yang and Worboys [2015] describe how to obtain a navigation graph from existing 2D building plans using dual graph to model relationships. First a structure graph is created from the building plans, which shows walls and doors, and, based on the same idea as IndoorGML where every room gets assigned a node, and rooms sharing a wall get an edge in the dual graph. Then the dual graph is filtered to only keep the edges that pass a wall with a door, just like the connectivity Node-Relation Graph (NRG) in IndoorGML. These steps are shown in Figure 3.2.

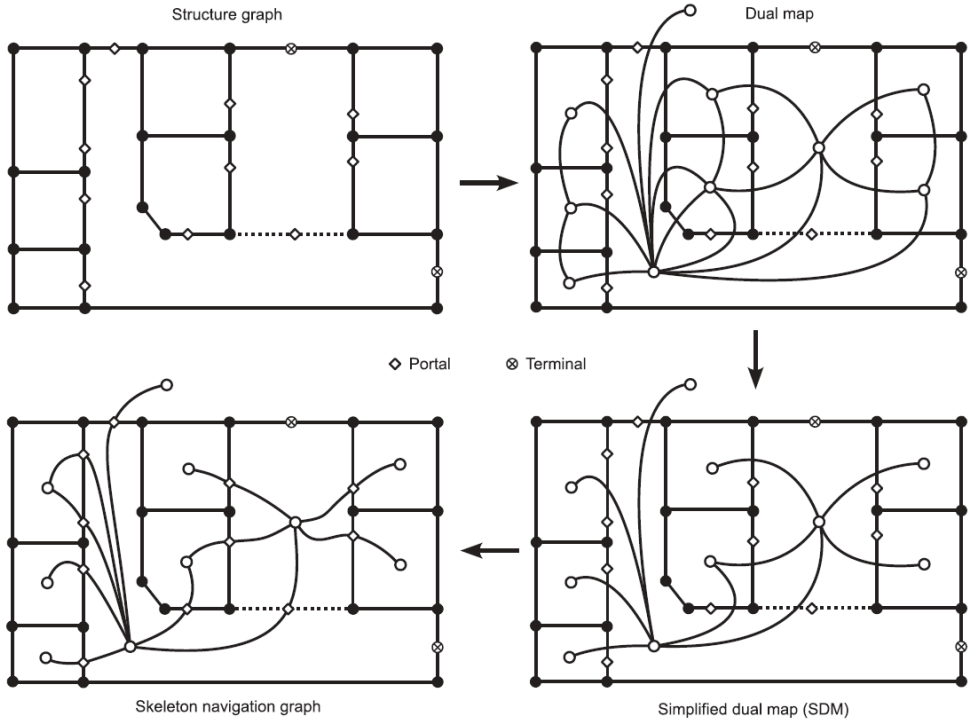


Figure 3.2: Steps done in creating a dual graph from a 2D floor plan [Yang and Worboys, 2015]

The same idea of duality is also used by Boguslawski et al. [2011], who applies it on a 3D building model, placing nodes in rooms, and connecting rooms that share a wall surface in the model. Jamali et al. [2015] also make a 3D network, but do not need a complete 3D building model. They use data gathered by a laser rangefinder, which can use laser beams to determine the distance from the scanner to an object. Of every room all corners are measured as x,y and z coordinates. At the location of the laser rangefinder in the room a node is placed. The connection between the nodes is obtained by Delaunay triangulation.

The idea of using the 2D Medial Axis Transform (MAT) (see Section 2.4.1) in building plans as a basis for routes in corridors is discussed by Meijers et al. [2005] and Lee [2004] (see Figure 3.3). It is also applied on obtaining outdoor road centrelines [Haunert and Sester, 2007]. An advantage named is that it also works for concave shaped polygons, and polygons with holes (representing e.g. columns or furniture).

A more recent development introduces the use of indoor point clouds as an input for a navigation network. Díaz-Vilariño et al. [2016] use region growing for the segmentation of a point cloud, and later classification of the segments. In this method obstacles are taken into account in the network. A Voronoi Diagram is created, with walls serving as constraint edges. The dual of the Voronoi Diagram, which is a Delaunay triangulated network, makes up the navigation graph. All Voronoi cell centres serve as nodes, and a connection is made with cell centres of Voronoi cells

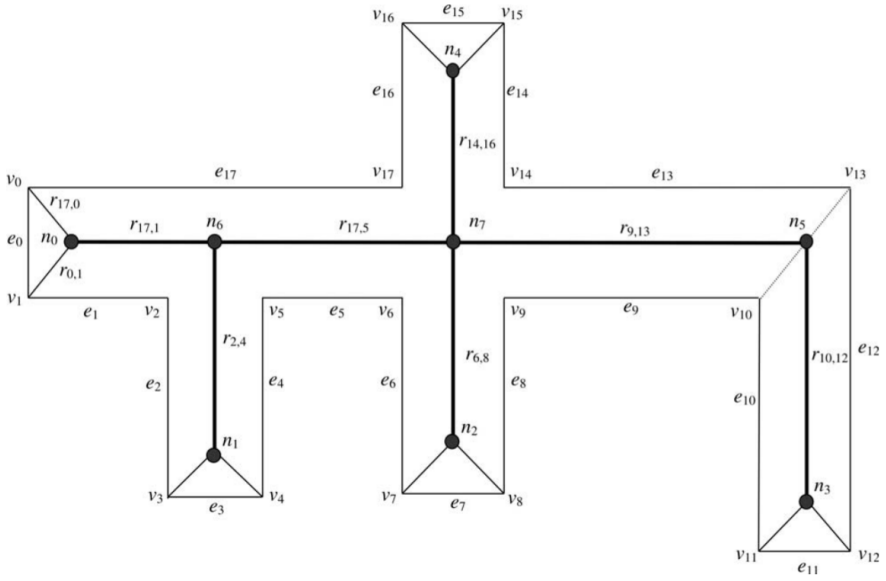


Figure 3.3: Straight-MAT in a 2D polygon representing a hallway [Lee, 2004]

that share an edge. In this method a Variable Density Network (VDN) is created, meaning that the density of Voronoi cells can increase around important areas, so extra nodes are added. VDN is also discussed by Boguslawski et al. [2016], who use it to generate a navigation graph from a 3D model of a building without obstacles. For this method corners of cells are detected, as an input for the graph. However, the density of the cells produced by this method is too high for the purposes of this thesis. As discussed in Section 1.2, larger scale networks based on connection between indoor spaces are considered in this thesis, because of efficiency and compatibility with the IndoorGML standard.

Another method that has been researched is cuboid reconstruction and merging in a point cloud [Tran et al., 2017], which creates a more general graph, having a node in every cell. This results in connectivity and adjacency relationships similar to IndoorGML. However, this method is limited by a Manhattan world assumption, and connectivity relationships are defined by manual insertion of doors.

A recent development in the generation of navigation graphs from indoor point clouds using hand-held MLS is making use of the scanner trajectory. This is for instance applied by Staats [2017] and Staats et al. [2018], where both point cloud and trajectory are voxelised (see Section 2.2.2). The detection of walkable space is a good basis for an indoor navigation network, because it includes information on where a person can be. Trajectory information can also be used in door detection [Díaz-Vilariño et al., 2017], which is extended on in Section 3.3. By segmenting the trajectory at the locations of doors it can be identified which points in the point cloud belong to a certain indoor space, with exception of points that are gathered through the doorway. When classifying points based on the room they belong to, different indoor spaces can be detected, and an IndoorGML navigation network can be generated.

3.3 DOORWAY DETECTION IN POINT CLOUDS

One of the most crucial steps needed to create a connectivity graph from an indoor point cloud is the detection of traversable doors or openings. When this information is known, connectivity relationships can be defined between different indoor spaces.

Díaz-Vilariño et al. [2014] and Díaz-Vilariño et al. [2015] establish door and doorway detection in a point cloud by using Generalized Hough Transforms (GHT) [Bal-

lard, 1981] to detect edges in orthoimages generated from a point cloud. By assuming general parameters for these edges, doors can be classified. Edge detection is also implemented by Díaz-Vilariño et al. [2016]. Instead of using colour information, wall planes are rotated in such a way that they can be read as a binary image. Pixels are assigned a value based on whether they contain points or not. These edge detection methods highly depend on the parameters that are defined for the size of doors. Doors that have non-standard sizes are ruled out for detection. Also closed doors are detected, but there is no way of knowing whether they can be traversed for real, or are closed permanently.

Besides methods based on edge detection, there are methods that use the trajectory of the MLS. One is described by Díaz-Vilariño et al. [2017], in which the height of the point cloud along the trajectory is analysed. When seeing a decrease in height, a door is probable to be at that location (see Figure 3.4). This method only works in buildings where door frames are clearly lower than the ceiling. Moreover a decrease in height could also indicate low hanging objects, such as low hanging lamps, making the method dependent on parameters indicating what value for drop in height is enough. In the research the point cloud is labelled into different regions, each containing all points that were scanned between two doors in the trajectory. A ray-casting method is carried out to evaluate the completeness of these regions, in order to find out whether parts of the indoor environment are occluded by objects. This is done by projecting the point cloud in a 2D grid, and selecting grid cells with the highest density of points as walls. From a grid cell selected as the scanning origin, it is checked which cells are occluded by walls. A ray-casting algorithm is also used by Nikoohemat et al. [2017] to detect openings in a voxelised point cloud. In this method distinction can be made between real openings and false ones caused by occlusions. By searching for voxels nearby trajectory points that represent door centres, doors can be detected.

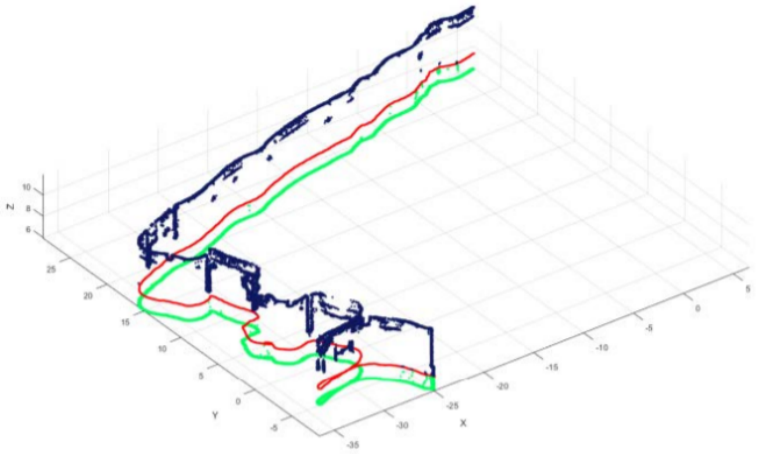


Figure 3.4: The trajectory of the laser scanner (in red), the vertical profile of the floor (in green) and profile of the ceiling (in blue) [Díaz-Vilariño et al., 2017]

From research it appears that in order to guarantee that a door can be traversed, and is not a permanently closed door or glass pane in the shape of a door, a human is required to walk through the door. This can be implemented by making use of the trajectory of a hand-held MLS.

3.4 DIVISION OF SUBSPACES

To provide for a more accurate measure of distance in a navigation network, and thus better route calculation and visualisation, some cells should be subdivided into

smaller subspaces (see Section 1.3). There are two main questions that come forth from literature: which cells should be subdivided, and how should the subdivision be done?

Jung and Lee [2015] define a set of rules in a flowchart as to when an indoor space should be subdivided. The space should be both navigable and accessible, and moreover a transition space, connecting at least two spaces in a navigation graph. In an indoor environment these kinds of spaces are usually squares or hallways. Kang and Li [2017] also discuss about when to apply subspacing, and mention size of the space and the presence of obstacles. If a space is large, has a concave structure, or has many objects blocking sight to other parts of the space, subspacing should be done.

Some authors define semantic criteria on how to subspace a indoor cell, such as subdivisions based on 'functional area' or 'functional space' around objects [Krūminaitė and Zlatanova, 2014; Diakité and Zlatanova, 2017]. However, from a point cloud it is difficult to get semantic information on the use of cells and objects they contain. Information like functional space around objects would be difficult to automatically extract. This leads to the conclusion that when using only a point cloud, the subdivision should be based on geometric elements. There is a distinction in literature between the subdivision of an area to provide more nodes, and the addition of extra nodes in a navigation network at convenient locations.

Methods based on the subdivision of areas geometrically include ones based on Voronoi diagrams [Wallgrün, 2005; Boguslawski et al., 2016], triangulation [Lamarche and Donikian, 2004], or the complete subdivision of a floor plan to a grid [Afyouni et al., 2012]. However the subdivision of space in a grid is very coarse and will slow down calculation times for path-finding algorithms. Diakité et al. [2017] discuss criteria for how to subdivide indoor environments for IndoorGML. Distinction is made between types of criteria, the first one being geometry-driven criteria, such as the shape of the cell. Topology-semantic-driven criteria take into account the placement of doors, and subdivide a corridor such that each subspace contains a door. The last one are navigation-driven criteria, which are based on the walkable surface in the cell. The visualisation of *Transitions* in a navigation graph is also discussed. Instead of a straight line, a *Transition* can be visualised as a gml:LinearString, which means that intermediate points on the line can be defined. This means that in some cells it is not necessary to add extra nodes, but just to edit the geometry of the Transition.

A method that focuses more on the addition of nodes in a navigation network is the Straight-Medial Axis Transformation (S-MAT) [Eppstein and Erickson, 1999]. It abstracts the skeleton of a polygon, that represents a route in the middle of the cell. This is also shortly discussed earlier in this chapter in Section 3.2, where Meijers et al. [2005] and Lee [2004] are named to have implemented this in corridors from 2D floor plans. A method based on the S-MAT is the 'door-to-door' approach [Liu and Zlatanova, 2011]. Instead of creating a route in the middle of a polygon, routes are created between doors, and corners of concave polygons. Both methods are shown in Figure 3.5. The 'door-to-door' approach argues that routes can be calculated and visualised in a more natural way, especially in situations like D1-D2 ("door-to-door") versus D1-M1-D2 S-MAT in Figure 3.5.

In conclusion, geometric subdivision of indoor cells is in most literature based on the floor plan of the space. This could be extracted from an indoor point cloud, by extracting the outline of the floor. Depending on the level of noise that polygon generated from the point cloud floors will have, nodes could be added based on one of the discussed methods.

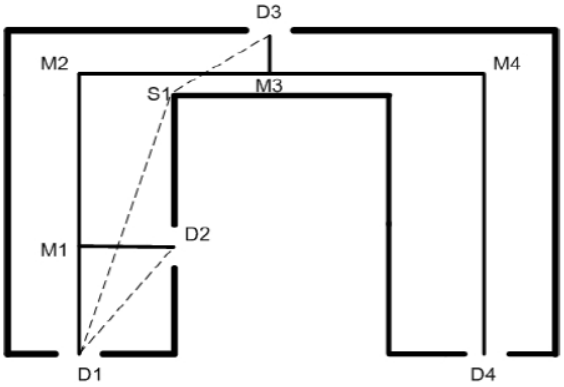


Figure 3.5: Door-to-door (D1-S1-D3) versus S-MAT (D1-M1-M2-M3-D3) route [Liu and Zlatanova, 2011]

3.5 CONCLUSIONS RELATED WORK

As discussed in Section 3.1 there is a need for obtaining useful information from point clouds, and reconstruction of geometric elements from a point cloud is an essential element in the modelling of a navigation graph. Existing methods focus on the modelling of 3D building models, such as BIM, and most start with a step of structuring the point cloud before modelling objects. Most methods are sensitive to noise and settings of different parameters.

The majority of research on creating indoor navigation networks (see Section 3.2) is based on existing 2D floor plans or 3D building models, which can be outdated, as discussed in Section 1.2. Many of them are based on the idea of duality, which is a reoccurring theme in IndoorGML, or the MAT of 2D polygons to get the middle line of corridors. Navigation graph extraction from point clouds often result in grid-based, dense networks, not suitable for IndoorGML. Methods making use of the trajectory of MLS show promising results, and can for instance be used in door detection.

Doorway detection is discussed in Section 3.3. Doors are an important element of an IndoorGML connectivity NRG and can be detected from point clouds using edge detection methods, or methods using the trajectory of the MLS. The advantage of the second is that these methods provide a check on whether a door-shaped object can also be walked through, making it less prone to errors.

Finally subspacing of indoor navigation graphs can be done to improve the measure of distance and visualisation and calculation of routes (see Section 3.4). It can be done based on knowledge on functional objects in an indoor environment, but this information is hard to obtain from a point cloud. Therefore it would be more obvious to apply subspacing based on the geometry of a space, adding nodes at convenient places in the environment. This can be done using the S-MAT, and another approach based on this is 'door-to-door' routes, which cut off routes in corners and therefore can give an even more accurate measure of distance.

4 METHODOLOGY

The goal of this thesis is to automatically construct a navigation network in IndoorGML format from an indoor point cloud. The methodology used to obtain the network is explained in this chapter. First the 3D Medial Axis Transform (MAT) of a point cloud is found (see Section 2.4.2), which results in sheets of points representing the skeleton of the point cloud. These sheets are filtered to only keep the ones that were formed inside walls, as explained in Section 4.1. The original point cloud, wall sheets and trajectory are voxelised (see Section 2.2.1), and are used together to detect doors in the point cloud (Section 4.2). Then the voxelised walkable space is created, as explained in Section 2.2.2. For the segmentation of walkable voxels into separate spaces, using the detected doorways. Connectivity relationships between spaces are defined, from which a connectivity Node-Relation Graph (NRG) can be created (Section 4.3), and extended to an accessibility NRG. All doors and separate spaces are assigned a node, and connectivity relationships between them define the edges. The addition of geometry information for each node in the network is discussed in Section 4.3.3. Finally, based on the geometry of the spaces, subdivision of connecting nodes in the navigation network is discussed in Section 4.4.

A summary of these steps is shown in Figure 4.1.

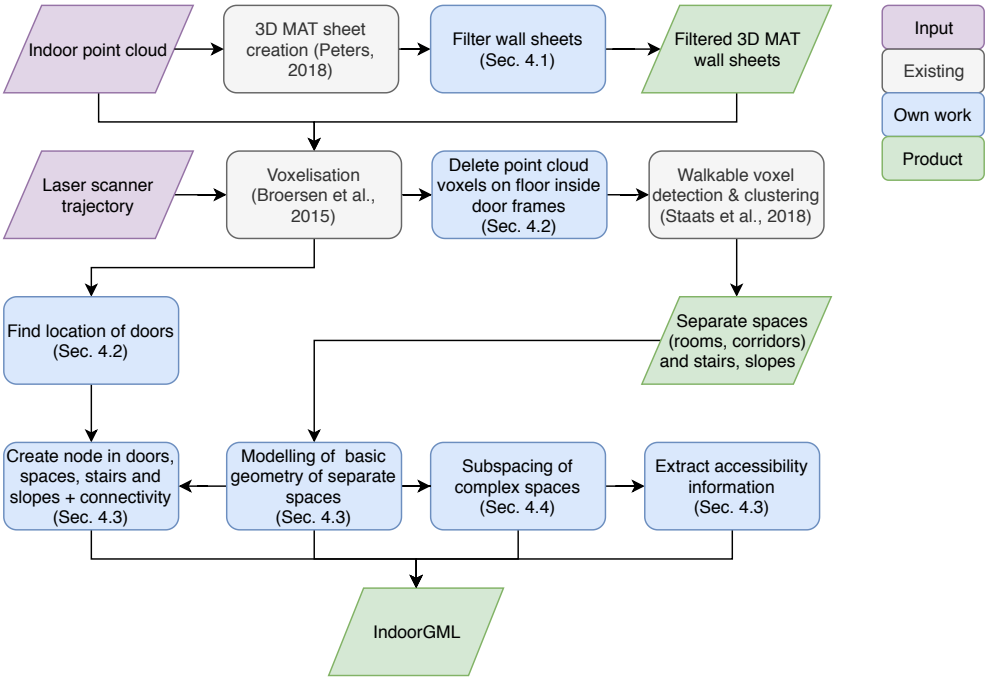
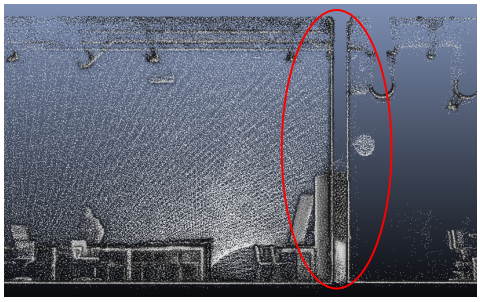


Figure 4.1: An overview of the steps done to obtain an IndoorGML graph

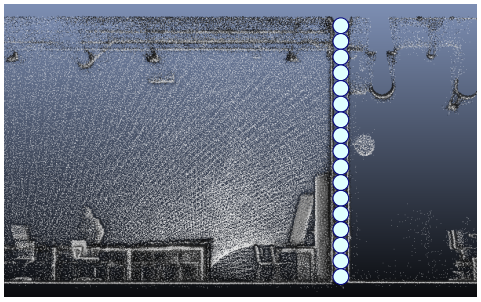
4.1 WALL SHEET EXTRACTION USING 3D MAT

Finding the MAT for 3D point clouds is discussed in Section 2.4.2. It has previously been applied to aerial point clouds, in which it can be used to identify buildings, waterways and other structures in the landscape. The 2D MAT has been applied in

floor plans to obtain polygon centrelines, which can be used as a basis of a route in a corridor, as described in Section 3.2. However, when applying the 3D MAT on an indoor point cloud, the results do not give clear centrelines of corridors and rooms. Unlike a 2D floor plan, which is often a complete, structured representation of the indoor environment, a point cloud is cluttered with static and dynamic objects (e.g. people and furniture), occluding parts of the space. This often leads to medial sheets not being found in the middle of corridors, providing an unreliable basis for route generation. However, medial balls are created inside walls that were scanned on both sides (see Figure 4.2). These balls are close together and have similar characteristics, so are grown into medial sheets (see Section 2.4.2). The centres of all medial balls belonging to a medial sheet in a wall are shown in Figure 4.3. By applying filtering parameters these wall sheets can be distinguished from the other sheets. This does mean that only walls that were scanned on both sides can be detected. It is however assumed that the location of other walls is not essential in a navigation network, just the ones that include doors that have been walked through, so that the two rooms adjacent to it can be separated.

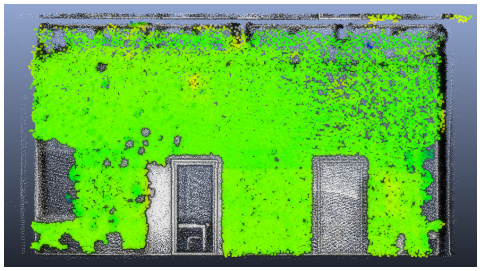


(a) Empty space in a wall that was scanned on both sides

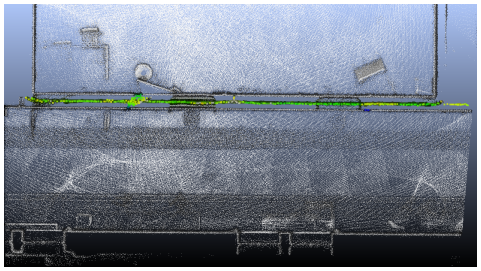


(b) Example of some medial balls formed in the empty space

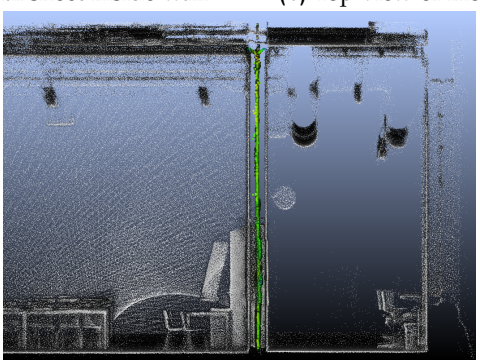
Figure 4.2: Example of a wall that was scanned on both sides



(a) Front view of medial sheet inside wall



(b) Top view of medial sheet inside wall



(c) Side view of medial sheet inside wall

Figure 4.3: Medial sheet formed inside a wall (green)

In order to distinguish between noisy sheets and wall sheets the geometric characteristics of medial balls (see Figure 2.12), and the total number of points in a sheet are used. These together make up five apparent characteristics with which wall sheets can be identified. First of all the separation angle θ is approaching 180°, which is the maximum value of θ in any medial ball, meaning that the average value of θ of all balls a wall sheet should be high. The second characteristic is the consistently low value of the radius r. Seeing that the centres of the medial balls are inside the walls, the distance from the centre to the corresponding point cloud points is generally quite small. For example, if the thickness of a wall is around 0.3 m, this would result in a radius of approximately $0.3/2 = 0.15m$. One of the medial balls as seen in Figure 4.2b is schematically visualised in Figure 4.4, together with its θ and r.

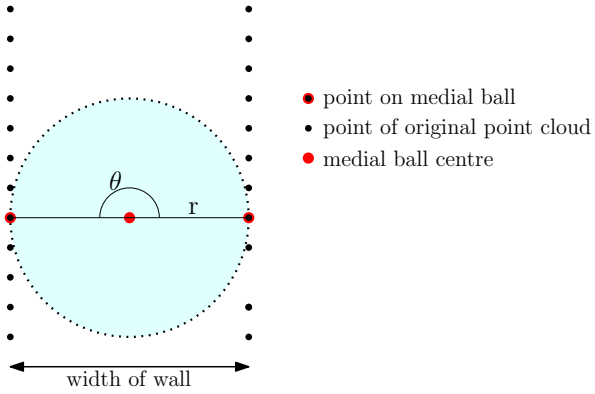


Figure 4.4: θ and r of a medial ball in a wall (side view)

The third characteristic also has to do with the radii. As already concluded, compared to other medial sheets the radius of medial balls inside walls is very low. Moreover, because the thickness of a wall does not change, the value is also constantly low. This can be used for more advanced filtering, by taking the standard deviation of radii in a medial sheet into account. This will help avoiding a medial sheets falsely being selected as wall sheet. A histogram of radius values in a medial sheet inside a wall is shown in Figure 4.5, as well as a histogram of a medial sheet that has a low median radius value, but is not inside a wall. As can be seen, the standard deviation of the distribution of radii in the wall sheet is much lower than that of the sheet not inside a wall.

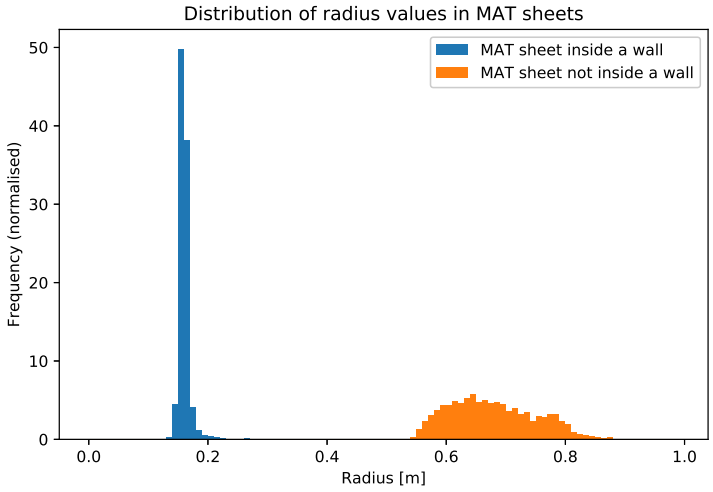


Figure 4.5: Histograms showing normalised distribution of radius values in two MAT sheets

The fourth characteristic used to filter the medial sheets is based on the direction of the normal vectors of the medial balls. It is used to filter out medial sheets generated between two floors in a building. In wall sheets normal vectors are pointed sideways, whereas in floor sheets they are pointed either upwards or downwards, resulting in a large absolute value for n_z, the z component of the normal vectors. This means that when the median n_z is low, the sheet is vertical, and thus could be inside a wall.

Lastly most of the sheets generated have a low number of points. The sheets that are formed inside walls commonly have a high number of points, because walls are relatively large objects in an indoor environment. A summary of requirements for these five characteristics is given in Table 4.1. Tests on these characteristics are described in Section 5.2.2 to define values for the parameters.

Parameter	**Value in MAT sheet**
Total number of points	High
Radius r	Median $<$ half the max. thickness of walls
Separation angle θ	Median approaching 180 °
Radius r	Low standard deviation
Z value normal vector n_z	Absolute median approaching 0

Table 4.1: Theoretical parameters for filtering MAT sheets

4.2 DOORWAY DETECTION FROM 3D MAT WALL SHEETS AND TRAJECTORY

In the proposed method indoor spaces need to be separated at the location of doors, to distinguish between different rooms. For this purpose not only the location of doors need to be detected, but also the voxels that lie on the floor inside a door frame (Figure 4.6). Two rooms connected to one another with a door need to be separated by at least an 8-neighbourhood (see Section 2.2.2), creating an adequate gap for the walkable space clustering algorithm.

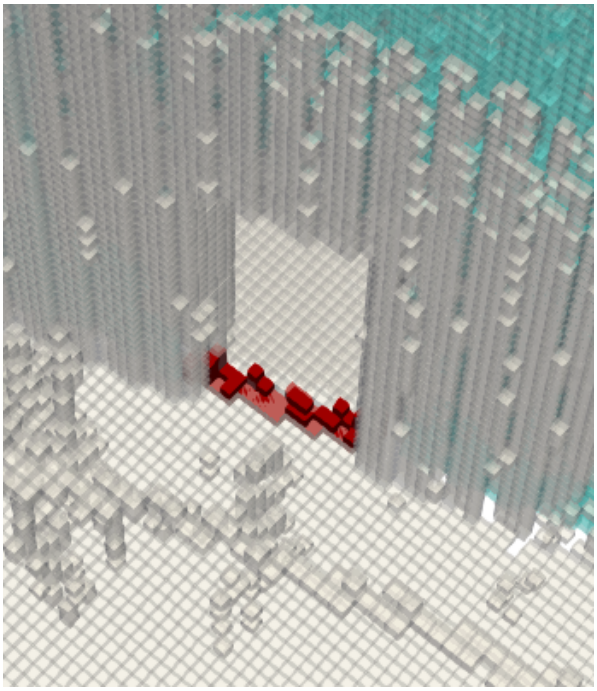


Figure 4.6: Voxels (in red) that lie on the floor inside a doorway

The doorway detection method proposed in this research can only detect doors that have been walked through with the laser scanner, making use of the trajectory. It has been decided not to use an edge detection or ray-casting method that does not include the trajectory, in order to prevent false positives - doorways being detected at locations where there is no opening - from occurring. Examples of potential false positives, that could be detected by those methods, are shown in Figure 4.7. Both occur in the point cloud used to develop the methods on, also described in Section 5.1.3. Making use of the trajectory of the laser scanner, the validity of doors is ensured. It should be noted that these non-valid doors could sometimes be of use in emergency response, but in this case excessive force is needed, equivalent to breaking a thin wall. The methods used in this thesis are described in the following Sections 4.2.1 and 4.2.2.

(a) A permanently closed door **(b)** A glass pane in place of a door frame

Figure 4.7: Examples of false doors in building of development point cloud

4.2.1 Geometry intersection

The flowchart in figure 4.8 represents the steps taken to get locations of doors in a point cloud from a medial sheet in a wall, using the geometry intersection method. This method fits in the 'Find location of doors' step of Figure 4.1, but does not require the point clouds to be voxelised. At first a plane is fitted through the points in a medial sheet selected as a wall (see Figure 4.9 step 1 and 2). This is done by taking the mean normal vector from all points, and defining the mean x, y, z coordinate as a point on the plane. The trajectory needs to be transformed from points to line segments. The points in the trajectory are saved in order of time, so a line segment is generated between each point (i) and the next ($i+1$). The line equation of these line segments is intersected with the plane equation, after which it is calculated whether the intersection point lies on the line segment. The intersection points that apply, are kept for the next check.

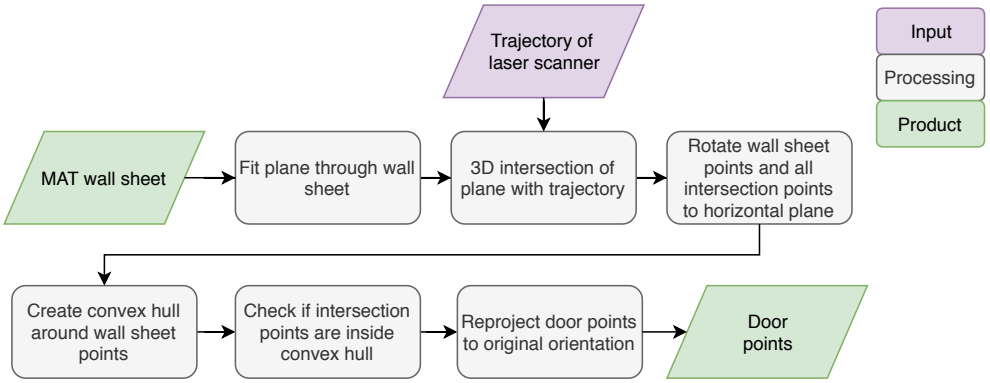


Figure 4.8: Flowchart of door detection in a point cloud using 3D MAT sheets and trajectory

A rotation matrix is applied on the medial sheet points and intersection points, to project them parallel to the XY plane (see Figure 4.9 step 3). The normal vector to the original plane is $\vec{v} = (a, b, c)^T$, and the normal vector to the XY plane is $\vec{w} = (0, 0, 1)^T$.

The rotation matrix (R) is defined based on the work of Cole [2015].

$$R = \begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix}$$

In which $\vec{u} = (u_x, u_y, u_z)$ is the unit vector, where $u_x^2 + u_y^2 + u_z^2 = 1$, and θ the rotation angle about an axis in the direction of $\vec{u}$. $\vec{u}$ is defined as:

$$\vec{u} = \frac{\vec{v} \times \vec{w}}{|\vec{v}|} = \frac{(b, -a, 0)^T}{\sqrt{a^2 + b^2 + c^2}}$$

And θ is defined as:

$$\theta = \frac{(\vec{v}, \vec{w})}{|\vec{v}|} = \frac{c}{\sqrt{a^2 + b^2 + c^2}}$$

In this case u_z is always 0, so the rotation matrix can be rewritten as:

$$R = \begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) & u_y\sin\theta \\ u_y u_x(1-\cos\theta) & \cos\theta + u_y^2(1-\cos\theta) & -u_x\sin\theta \\ -u_y\sin\theta & u_x\sin\theta & \cos\theta \end{bmatrix}$$

After applying the rotation matrix, the Z component of the transformed points is temporarily removed, after which a 2D convex hull is created around the sheet points (see Figure 4.9 step 4). For each intersection point it is then checked whether they are inside this convex hull, meaning that the trajectory crosses the wall segment there. This is done by reconstructing the convex hull for each intersection point with the sheet points. If the convex hull stays the same, it means that the point is inside. If the convex hull changes, it means that the point is outside. The points that are proven to be inside are rotated back to the original orientation of the plane, showing the location of the doors. For this the inverse of the rotation matrix is used.

This method is limited to the detection of doorways that also have a part of a wall above them, because otherwise the sheets on the left and right side of the doorway will not be connected, and therefore not be able to generate a convex hull that can intersect with the trajectory.

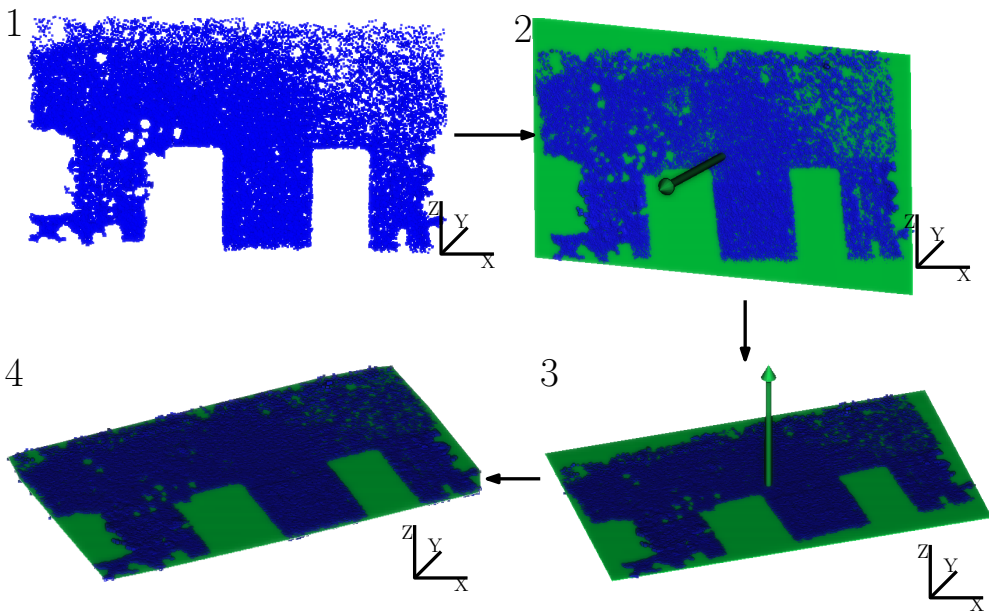


Figure 4.9: Steps in rotating a medial sheet and finding the 2D convex hull

4.2.2 Voxelised method

The method discussed in Section 4.2.1 has some limitations, as mentioned above, which is why another door detection method has been developed, using a voxelised version of the medial sheets. Because of this, it is also better related to the walkable voxels, which are used to define indoor spaces in a later step. For an explanation of the voxelisation method, see Section 2.2.1.

There are two variations on door detection using the voxelised trajectory, wall sheets and point cloud:

1. Door detection by finding wall sheet voxels **above** the trajectory.
2. Door detection by finding wall sheet voxels **around** the trajectory.

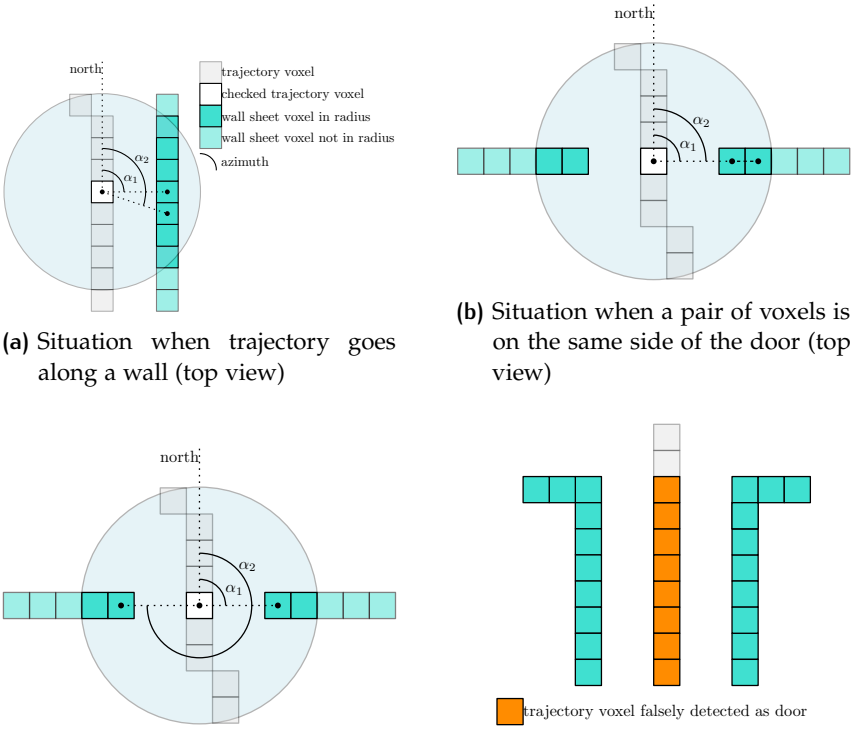
Wall sheet voxels **above** trajectory

The first variation needs a door to have a wall above it. Using the voxelised trajectory, a lookup function is done for each trajectory voxel to check whether there exists a wall sheet voxel *above* it. If such a voxel exists, it can be said that this trajectory voxel is inside a door frame. A key advantage as opposed to other methods that look for walls above the trajectory (as discussed in Section 3.3), is that in this method the walls are already detected from the original point cloud. A drop in height above the trajectory as detected in original methods could have various meanings. Besides it being a door frame, it could be caused by low hanging objects in a room or hallway. If a hallway is cluttered with low-hanging objects that are the same height above the floor as a door frame, the 3D MAT wall sheets will ensure detection of exclusively all voxels that are inside a door frame. It will prevent the possibility of false positives to occur.

Besides using the wall sheet voxels to obtain trajectory voxels that are inside a doorway, they can also be used to detect voxels on the floor in a door frame, as visualised in Figure 4.6. This can be done by again projecting the wall sheet voxels down, and checking whether original point cloud voxels exist below them.

Wall sheet voxels **around** trajectory

Not all doors will have a wall above them, as some are of the same height as the ceiling. This is why the second variation of the method exists. In this method all wall sheet voxels that lie in a specific radius *around* a trajectory voxel are detected. In order to speed up the algorithm a 2D 'slice' is taken at the height of the trajectory,



(a) Situation when trajectory goes along a wall (top view)

(b) Situation when a pair of voxels is on the same side of the door (top view)

(c) Situation when a pair of voxels is opposing each other in a door (top view)

(d) Situation when a corridor is narrow and door voxels are falsely detected (top view)

Figure 4.10: Door detection method by looking at wall sheet voxels around a trajectory voxel. Legend in (a)

and only wall sheet voxels inside the radius at this height are regarded. For each of these wall sheet voxels, the angle that it makes with the north, as seen from the trajectory voxel, is calculated. This angle is also called the *azimuth*, shown in Figure 4.10. In the case of pixels in a Cartesian grid the direction of the North is the same as the direction of the y-axis.

Each wall sheet voxel within the radius of a single trajectory voxel is paired with all other wall sheet voxels in this range. This means that for a trajectory voxel with n wall sheet voxels in its radius, there will be $\binom{n}{k} = \left(\frac{n!}{k!(n-k)!}\right)$ pairs. For each pair the difference in azimuth is calculated. Trajectory voxels that lie alongside walls, will have only pairs of wall sheet voxels that have a small difference in azimuth ($\_1$-α_2) (see Figure 4.10a), whereas trajectory voxels that lie inside doors will also have pairs of wall sheet voxels that have a big difference in azimuth (see Figure 4.10c). This difference in angle is approximately 180°. In this way doors can be detected by looking *around* the trajectory voxels.

However, in a narrow corridor the *around* method would define many consecutive trajectory voxels to be inside a door (see Figure 4.10d). This situation can be prevented, by grouping all detected door voxels with their neighbours. If a group of neighbouring voxels is small, the group is probably inside a door, while when there is a large number of voxels in a neighbour group, it is not.

After detecting the trajectory voxels that are inside a door frame, these voxels can be used to detect voxels on the floor inside the doorframe (see Figure 4.6). For each trajectory voxel that is defined to be in a door, one pair of MAT sheet voxels needs to be selected. Specifically it should be the pair that lies opposite of each other in a doorframe and has the shortest distance to one another. An example of this is the pair of selected voxels in Figure 4.10c. A line should be generated between those two voxels, which should be voxelated in a way that 8-connectivity is ensured. The voxels defined to be on this line are used to detect voxels that lie on the floor inside

a door by projecting them down. Because the pair of voxels lies at the same height, a 2D line formula is used.

The pseudocode for the line-pixelation algorithm is given in Algorithm 4.1. The input is a pair of voxels that lie opposite one another in a door frame. Because this voxel pair has the same height, the z-value is not taken into account in the line generation, but added to the pixels on the line later. In the voxelisation step, explained in Section 2.2.1, all voxels get an integer value for x, y and z. This means that the distance between 4-neighbouring voxels at the same height is 1, and between 8-neighbouring voxels at the same height is $\sqrt{2} \approx 1.41$ (Figure 2.5). This is rounded up to 1.5 to ensure 8-connectivity in the line pixelation.

Algorithm 4.1: Line pixelation algorithm

Input: A pair of voxels V_1, V_2 at the same height
Output: A list of voxels that lie on the line between them
$L = \{L_1, L_2, \ldots, L_n\}$

```
min_x ← min(V_1x, V_2x)
max_x ← max(V_1x, V_2x)
min_y ← min(V_1y, V_2y)
max_y ← max(V_1y, V_2y)
z ← V_1z
L ← ∅
# generate a grid of voxels between the 2 segment voxels
for i ← min_x to max_x + 1 do
    for j ← min_y to max_y + 1 do
        # calculate distance from voxel to line generated between V_1 and V_2
        distVoxLine ← Distance(Point(i, j), Line(V_1, V_2))
        if distVoxLine ≤ 1.5 then
            L+ = Point(i, j, z)
```

4.3 INDOORGML NETWORK CREATION

After voxels on the floor inside door frames are detected, the clustering method as discussed in Section 2.2.2 is used to find the walkable voxels. Clusters of voxels all get a unique identifier. The geometry of the clusters and locations of doors can be used together to define connectivity relationships between indoor spaces via doors. First a merging step is done on the results of the clustering method (Section 4.3.1), because some spaces are separated in height. In Section 4.3.2 it is explained how the connectivity graph is created, and after that how the spaces get geometry (Section 4.3.3). Finally in Section 4.3.4 it is explained how accessibility information is added to the graph.

4.3.1 Merging clusters separated by height

Merging overlapping clusters

In the walkable space detection algorithm, described in Section 2.2.2, the voxelised point cloud is sliced at all height intervals, so the clusters consist of voxels at only one z-value. They can be regarded as slices of 2D pixels. The voxelisation method results in a voxel size of approximately 5 to 8 cm in all directions. This means that when there is a slight change in height, two layers of clusters that are in the same space, but have a different z value, will not be defined as belonging to the same room (see Figure 4.11).

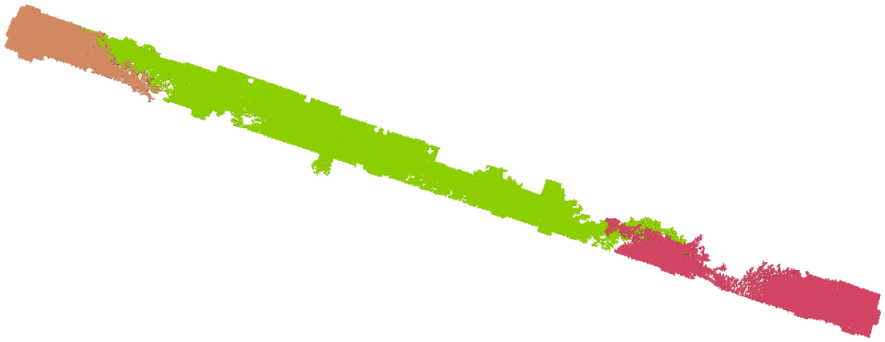


Figure 4.11: Clusters in one corridor having a different ID, because of slight variation in z value

In order to solve this problem, a merging step is added after clustering, explained in Algorithm 4.2. It finds clusters that are overlapping and only 1 voxel in height apart from each other. The clusters are merged by updating the cluster ID of one of them, and the process is repeated until the number of distinct cluster IDs remains the same, meaning that there are no more clusters to be merged in this way.

Algorithm 4.2: Merging clusters that differ one voxel in height

Input: A set of walkable voxels V
Output: An updated set of walkable voxels V

$diffLen \leftarrow 1$
$newLen \leftarrow length(distinct(V_{id}))$
while $diffLen > 0$ **do**
  $oldLen \leftarrow newLen$
  # execute SQL query
  "$SELECT\ V1.V_{id}, V2.V_{id}$
  $FROM\ V$ AS $V1$, $V\ AS\ V2$
  $WHERE\ (V1.x = V2.x\ AND\ V1.y = V2.y\ AND\ V1.id \neq V2.id\ AND$
  $|V1.z - V2.z| \leq 1)$"
  $V2.V_{id} \leftarrow V1.V_{id}$
  $newLen \leftarrow lenght(distinct(V_{id}))$
  $diffLen = oldLen - newLen$

Merging clusters based on trajectory

When larger variations in floor height occur in one room, for example with the presence of stairs, the merging of clusters based on vertical neighbourhood will not work (see Figure 4.12). To join those clusters the trajectory of the laser scanner can be used. The trajectory does not only hold information on *where* the laser scanner was, but also *when* it was there. There is a time stamp included in all trajectory voxels. Together with the location of doorways (described in Section 4.2), the trajectory can be split at the location of doors. Each part of the trajectory, which goes from a door to another door, is given a similar ID. All separate clusters that are below a trajectory part are grouped in the same cluster. This only needs to happen in the case of stairs included in a single room, and not when a stair connects two spaces. Therefore for all separate clusters that are to be grouped, it is also checked if they are connected to more than one door. If so, the cluster is not grouped.

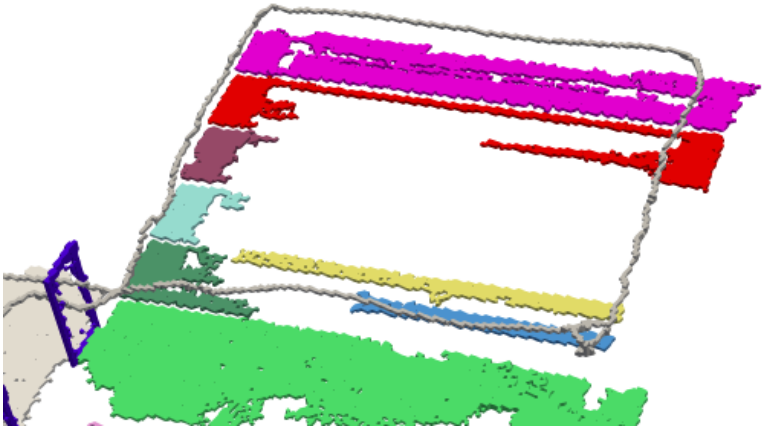


Figure 4.12: Floor space in same room assigned different ids, because of larger variation in z value

4.3.2 Extracting the connectivity graph

When all walkable voxels are clustered based on the indoor space they belong to, a connectivity graph can be extracted from this. The type of graph that is aimed for in this thesis is the IndoorGML thick wall model (see Figure 2.2, which also contains nodes at the location of doors. In this subsection it is explained how a basic IndoorGML connectivity graph is obtained from the walkable voxel clusters and locations of nodes.

Node creation

Initially, in the method discussed up until this point, every time the laser scanner trajectory crosses a door, a node is created in this door. This will lead to doors that were walked through multiple times having multiple nodes, which should be merged into one. This can be done by executing a neighbour search for all trajectory voxels in doors, and grouping the ones that are close to each other.

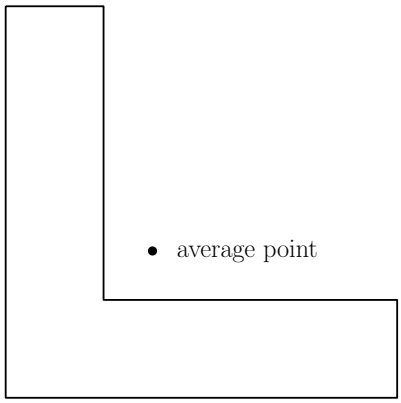


Figure 4.13: Average x,y of an L-shaped polygon

The location of room and corridor nodes should be determined by the geometry of these spaces. The node that represents such a space should of course also be inside the space. This is straightforward for convex spaces, because simply the average x and y value can be taken from all voxels that belong to that space. Finding a point inside a concave space is more complicated, because this average value could end up being outside of the shape (Figure 4.13). For this a polygon first needs to be modelled around all voxels belonging to the room or corridor cluster. This is

discussed further in Section 4.3.3. Another solution is to randomly pick one of the voxels belonging to the cluster, but this will often not be a general representation of the shape.

Stairs and slopes

Besides detecting which voxels are walkable floors, Staats [2017] also defined which voxels are part of stairs or slopes. In IndoorGML stairs are part of the class *TransitionSpace*, just like corridors, and can thus be separate indoor cells, represented with a node in the network. Using the timestamps of voxels in the trajectory, as done in Section 4.3.1, it can be checked which two indoor spaces the stair is connecting. If one of those spaces is not connected to a door, as for instance the top floor in Figure 4.12, they should not be assigned a separate ID, but the same as the other floor.

Connectivity determination

Connectivity is determined based on the location of door, stair and slope nodes and the coordinates of the voxels belonging to floor clusters. Every door, stair and slope that has been detected is connecting two indoor spaces with each other, which is why the connectivity network is built up from these nodes. By applying a neighbourhood search to every door node it is detected which two spaces lie right beside it, and thus which two spaces it is connecting. For the stairs and slopes a node is created at the top and bottom of the stairs (see Section 4.4.2). For the top stair node it is known that it is connected with the bottom stair node, and a space or doorway at the top. The bottom stair node is connected with the top stair node and a space or doorway at the bottom. With this information determined the first version of the connectivity NRG can be generated.

An example of an IndoorGML connectivity NRG that is not subspaced is shown in Figure 4.14.

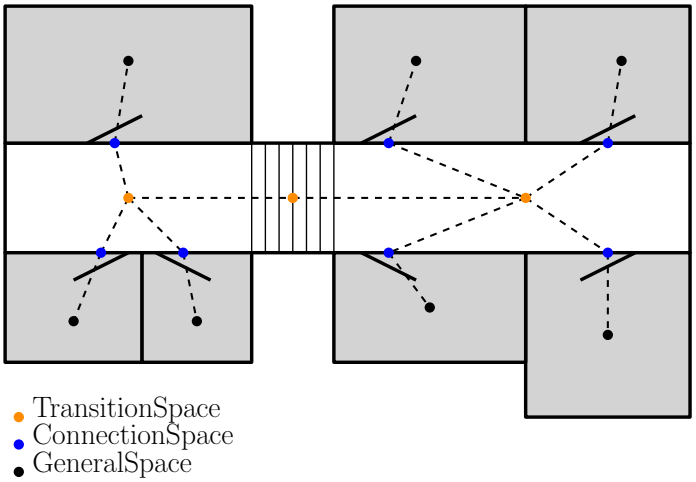


Figure 4.14: Example of an IndoorGML connectivity graph with a stairway

4.3.3 Geometry modelling

The modelling of a 2D geometry for all rooms and corridors in the navigation network is important for various reasons. Firstly, it is necessary to find the shape of an indoor space for the placement of nodes in the graph, as mentioned in Section 4.3.2. Not only is it needed for the placement of single nodes in rooms, but also for the placement of extra nodes in more complex shapes, such as long corridors with multiple connections. This idea is discussed in further detail in Section 4.4. Secondly, the geometry of a space can be used as an addition to the IndoorGML model. In this case the geometries of the spaces should be non-overlapping, and

ideally the geometries of adjacent spaces should touch. For IndoorGML geometry modelling the 2D model can be extruded to a 3D model, using height information from the point cloud.

The 2D geometry of a group of voxels representing a certain room is modelled using the α-shape, as described in Section 2.3.1. This method is very dependent on the value chosen for parameter α, which is why in the implementation (Section 5.2.5) tests are discussed that provide an overview of the optimal value for α.

4.3.4 Accessibility information extraction

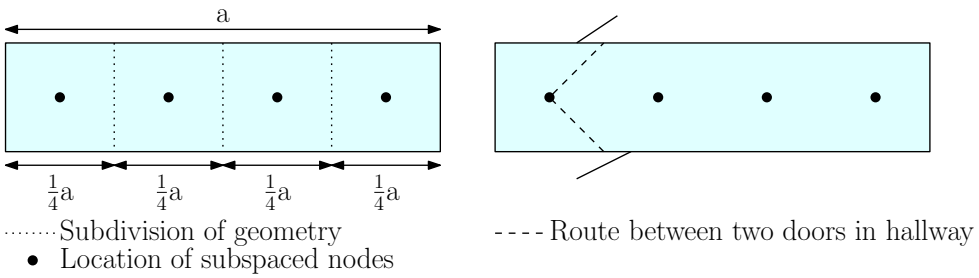
As concluded in Section 3.4 it is not possible to obtain some kinds of accessibility information from a point cloud, such as opening hours, or accessibility rights for certain people. As a proof of concept, it is therefore decided to include only geometric accessibility information, such as the location of stairs. However, it is not specified in IndoorGML how to save this information in the model, because an accessibility NRG varies depending on the different restrictions for different people. This is why the method in this thesis is limited to saving this information in a database, so that it can be obtained at a later moment to construct a person-specific accessibility graph. Two types of accessibility information are extracted as a proof of concept:

- The location of stairs and sloped surfaces, which is already available from previous methods discussed.
- The width and height of doors. The width can be detected by finding the closest point cloud voxels around a trajectory voxel marked as a door in the horizontal plane. Using the same method as the door detection method looking **around** the trajectory (see Section 4.2.2) the distance between the pair of voxels on both sides of the doorway can be calculated. The height of the door can be found by counting how many voxels there are above and below the trajectory voxel inside a door before a point cloud voxel is detected. This number has to be multiplied with the voxel size.

4.4 SUBSPACING

Based on related work on subdivision of indoor spaces for navigation networks (Section 3.4) it is decided to only apply subdivision on indoor spaces that connect multiple spaces, such as corridors (*TransitionSpace* in IndoorGML). This information can be retrieved from the connectivity NRG, by analysing the number of connections that a node makes, and selecting all nodes that are connected to two or more nodes.

The α-shapes created by the method described in Section 2.3.1 are used as input for the subspacing step. Various methods are described in Section 3.4 for the subspacing of 2D polygons for navigation networks. Probably the most straightforward and easy, especially for straight corridors, is subdivision of the geometry into equal parts, and placing a node in the centres of the parts (Figure 4.15a). Although this method will produce extra nodes in an indoor space, these nodes are often not at the most convenient locations for navigation, as shown in example (Figure 4.15b). The route created between the two doors is not representative of the route that a person would prefer to walk. Therefore a more complex subdivision method is proposed in this section, based on the MAT of the space and location of doors connected to this space.



(a) Subdivision of a polygon into equal parts **(b)** Route from one door to another when hallway is equally subdivided

Figure 4.15: Equal-parts subdivision of a space

4.4.1 Subspacing using the MAT

An explanation of how the MAT of 2D polygons is defined can be found in Section 2.4.1. For an indoor space that has to be subdivided the α-shape generated around all its voxels can be used. The 2D MAT will be in the middle of the corridor, consisting of line segments representing the skeleton of the polygon. For every door that is connected to the corridor the closest point on the MAT should be found, and added as a node to the network (see Figure 4.16). Extra nodes can be defined at T-junctions and in sharp turns, because here more than two line segments of the MAT come together in one point. If a coordinate only occurs once in the set of line segments it means that this point is an endpoint. When the locations of subspaced nodes are found, connectivity relationships between them are not automatically defined. For this an analysis of the order of line segments in the MAT is done.

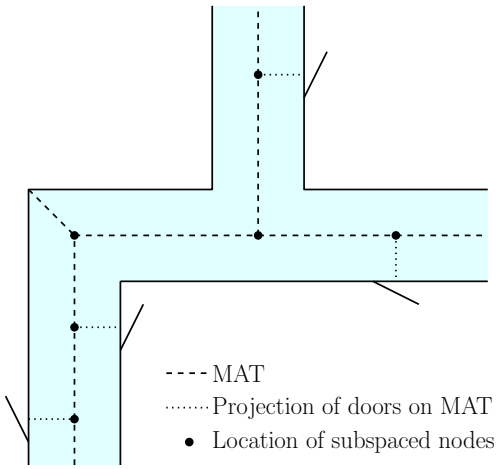


Figure 4.16: Node generation based on MAT in an L-shaped corridor with T-junction

The internal connectivity between subspaced nodes of an indoor space can be found by doing an analysis based on the order of line segments in the MAT. First of all the line segments are revised and line segments with subspaced nodes on them are split at that location (Algorithm 4.3). After this step the connectivity relationships can be defined for a node by looping through MAT line segments and connecting them end-to-end, until a next node is found. The pseudocode for this algorithm can be found in Algorithm 4.5. It makes use of the recursive function *linestringLookup*, in which the lines are connected end-to-end, explained in Algorithm 4.4.

Algorithm 4.3: Linestrings are split at the location of subspaced nodes

Input: List of x,y coordinates of subspace nodes (value) *Coords*, Collection of linestrings representing the MAT *Lmat*, List of end points of the MAT *EndP*

Output: Updated collection of linestrings that are split at nodes *LmatFilt*

```
# First split linestrings that have a node somewhere on them into 2 parts
LmatFilt ← ∅
foreach linestring L in Lmat do
    onDoor ← False
    foreach Coord in Coords do
        if Distance(L,Coord) < 0.001 then
            onDoor ← True
            L_1 = First point on L
            L_2 = Second point on L
            Append Linestring(L_1, Coord) to LmatFilt
            Append Linestring(L_2, Coord) to LmatFilt
    if not ondoor then
        Append L to LmatFilt
```

Algorithm 4.4: Recursive function *linestringLookup*

Input: Next point to be found *FindPoint*, Previous point visited *PrevPoint*, Dictionary of ID (key) and geometry (value) of subspaced nodes and endings of MAT *EndNodes*, Collection of linestrings (split at location of nodes) representing the MAT *LmatFilt*, List of linestrings *Lnodes*

Output: List of linestrings between 2 nodes *Lnodes*

```
# When an end node is reached, stop recursions
foreach ID,EndNode in EndNodes do
    if FindPoint == EndNode then
        return
foreach linestring L in LmatFilt do
    L_1 = First point on L
    L_2 = Second point on L
    #Prevent recursion from going back and forth between the same points
     with the 'and not' clause
    if FindPoint == L_1 and not PrevPoint == L_2 then
        Append L to Lnodes
        Run linestringLookup with L_2 as FindPoint, and FindPoint as
          PrevPoint (the rest stays the same)
        return Lnodes
    else if FindPoint == L_2 and not PrevPoint == L_1 then
        Append L to Lnodes
        Run linestringLookup with L_1 as FindPoint, and FindPoint as
          PrevPoint (the rest stays the same)
        return Lnodes
```

Algorithm 4.5: Finding connectivity between subspaced nodes

Input: Dictionary with ID of subnode (key) and x,y coordinates of node (value) *Sn*, Collection of linestrings (split at location of nodes) representing the MAT *LmatFilt*,Dictionary of ID (key) and geometry (value) subspaced nodes and endings of MAT *EndNodes*
Output: Dictionary with pairs of connected nodes (key) and multilinestring geometry between them (value), *ConnDict*

```
1  foreach ID,Coord in Sn do
2      foreach linestring L in LmatFilt do
3          LineList = List(L)
4          L1 = First point on L
5          L2 = Second point on L
6          if L1 == Coord then
7              LineList = linestringLookup with FindPoint = L2, PrevPoint =
                Coord, EndNodes = EndNodes, LmatFilt=LmatFilt,
                Lnodes=LineList
8              Add (endpoint,ID) to ConnDict with value LineList
9          else if L2 == Coord then
10             LineList = linestringLookup with FindPoint = L1, PrevPoint =
                Coord, EndNodes = EndNodes, LmatFilt=LmatFilt,
                Lnodes=LineList
11             Add (endpoint,ID) to ConnDict with value LineList
```

4.4.2 Subspacing of stairs and slopes

As discussed in Section 4.3.2 in IndoorGML stairs are referred to as *TransitionSpace*. A staircase will be marked with a node in the connectivity NRG. The stair node needs to be subspaced, in order to make a more complete navigation network. It does not state in IndoorGML how this must be implemented, so it has been chosen to divide the stair node into two: one node at the top, and one at the bottom, at which locations the stairs connect two different spaces.

These locations are found by analysing the trajectory voxels in order of capture time, and detecting the locations at which a change occurs between floor voxel and stair or slope voxel. In order to prevent multiple nodes occurring for the same stair that has been walked on multiple times, a neighbour search similar to the one for door nodes is executed, as described in Section 4.3.2.

5 IMPLEMENTATION, EXPERIMENTS AND DEVELOPMENT RESULTS

The implementation details of the methodology of this thesis are discussed in this chapter. First the hardware, software and datasets used are discussed in Section 5.1. Distinction is made between two datasets: the development point cloud, on which the methods were developed, and the test point cloud, of which the results are shown in Chapter 6. The implementation of the methodology proposed in Chapter 4 is presented in Section 5.2. Lastly experiments to evaluate the results of the methods on the test point cloud are presented in Section 5.3

5.1 TOOLS AND DATASETS

The software, hardware and datasets used to develop and evaluate the methodology are discussed in this section, starting with a discussing on software (Section 5.1.1). Secondly the hardware used for gathering the point clouds and type of computer used to run the algorithms are described in Section 5.1.2, and finally the development and test datasets are reported in Section 5.1.3.

5.1.1 Software

In this section an overview is given of software used in this thesis. An elaboration on how they were used is given in Section 5.2.

PYTHON

The programming language Python (version 2.7) is used in the implementation of most steps of the methodology, because of its ease in prototyping and the availability of various external packages to support the algorithms. The following Python packages are used in this thesis:

- NetworkX[1] can be used for creation, manipulation and structuring of network graphs. It is used in this thesis to analyse the navigation network generated by the methodology.
- NumPy[2] is an extension for Python to support numerical algorithms for multi-dimensional arrays. In this thesis NumPy is used for working with multi-dimensional arrays and linear algebra operations.
- psycopg2[3] supports database transactions from Python. It is used to connect to a PostgreSQL database where the point cloud, trajectory and network are stored.
- pykdtree[4] is used to generate a KD-tree for point clouds, in order to execute faster neighbour search.

1 https://networkx.github.io/
2 http://www.numpy.org/
3 http://initd.org/psycopg/
4 https://github.com/storpipfugl/pykdtree

- SciPy[5], and especially SciPy Spatial[6] offers a variety of spatial and geometric functions, of which Delaunay triangulation is used for the creation of the α-shapes for floor plan reconstruction.
- Shapely[7] supports the manipulation and analysis of 2D geometric objects. It is used to read and write Well-Known Text (WKT) and Well-Known Binary (WKB) strings, bridging the gap between Python and PostgreSQL.
- skel3d[8] is developed by Peters [2018] to retrieve the 3D Medial Axis Transform (MAT) from a point cloud in Python. It also offers functions for setting parameters for processing the MAT.

PostgreSQL[9] with PostGIS[10]
PostgreSQL (both version 9.5 and 10 used) is a free database management system used to store the point cloud, trajectory, its octree and the connectivity network. PostGIS (both version 2.3 and 2.4 used) is an extension on PostgreSQL, which provides support for geometric objects, and comes with a variety of spatial functions.

CloudCompare[11]
An open source point cloud and mesh visualisation and processing software used for file format conversions and visualisation purposes in this thesis.

Paraview[12]
An open source analysis and visualisation software for scientific purposes. It is used for the visualisation of the voxelised point cloud and connectivity network.

5.1.2 Hardware

ZEB-REVO
Both the point cloud used in the development phase and the one used in the evaluation of the methods were captured with a ZEB-REVO laser scanner. This is a hand-held Mobile Laser Scanner (MLS), which can be used while walking around in a building. In order to process the output, Simultaneous Localization and Mapping (SLAM) is used, as explained in 3.1. This is an algorithm that uses a combination of the Inertial Measurement Unit (IMU) and object recognition in the point cloud to locate the scanner. The further away the scanner is from the starting point, the worse the accuracy. The following specifications are listed in a brochure for the ZEB-REVO laser scanner [GeoSLAM, 2017]:

- Scanning range: 30 m (outdoors may range 15–20 m)
- Data acquisition rate: 43,200 points/sec
- Relative accuracy: 1–3 cm
- Absolute positioning accuracy: 3–30 cm, after 10 minutes of scanning

Computer
With regards to experiments which discuss processing time of algorithms, the specifications of the computer used are given.

- Laptop: Dell Latitude 5580 running on Windows 10

5 https://www.scipy.org/
6 https://docs.scipy.org/doc/scipy/reference/spatial.html
7 https://github.com/Toblerity/Shapely
8 https://github.com/Ylannl/skel3d
9 https://www.postgresql.org/
10 https://postgis.net/
11 https://www.danielgm.net/cc/
12 https://www.paraview.org/

- Processor: Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz
- Sytem type: 64-bit Operating System, x64-based processor

5.1.3 Datasets

Two indoor point clouds and their trajectories were used in this thesis. The first one is used to develop the methodology on, and the second one is used to test the algorithms and validate the methodology.

Development dataset

The point cloud and trajectory used to develop the methods on is captured in the Architecture faculty of Delft University of Technology, the Netherlands (Figure 5.1), using a ZEB-REVO hand-held laser scanner. The point cloud consists of more than 37.8 million points. It is made up of two floors of a building, but also includes rooms that are on 'half a floor', and a big room that spans two floors in height. It contains multiple staircases. For each experiment and test in this chapter only parts of this point cloud are used for development, to save processing time of the algorithms.

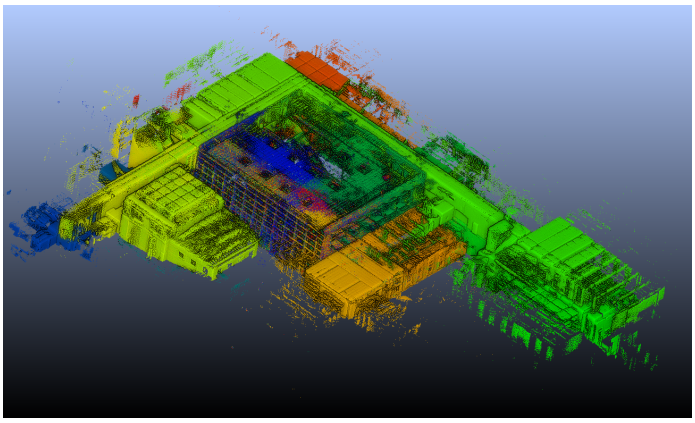


Figure 5.1: Development point cloud, captured in the Architecture faculty of Delft University of Technology, the Netherlands

Test dataset

The test dataset used to independently evaluate the methods proposed in this thesis is one of the International Society for Photogrammetry and Remote Sensing (ISPRS) benchmark datasets [Khoshelham et al., 2017], captured with a ZEB-REVO laser scanner in one of the buildings of Technische Universität Braunschweig, Germany (Figure 5.2). The point cloud consists of almost 21.7 million points. The building was unfurnished at the time of capture, so there is less clutter and occlusion of walls than in the development dataset. It consists of two floors, connected with a staircase. There are no rooms at unequal floors, and both floors are approximately the same height. Some rooms are passed by, but not scanned thoroughly.

A summary of both point clouds and some of their characteristics are given in Table 5.1.

Dataset	Nr of points	Furniture	Multiple floors	Unequal floors
Development	37.8 million	Yes	Yes	Yes
Test	21.7 million	No	Yes	No

Table 5.1: Summary of characteristics of datasets used

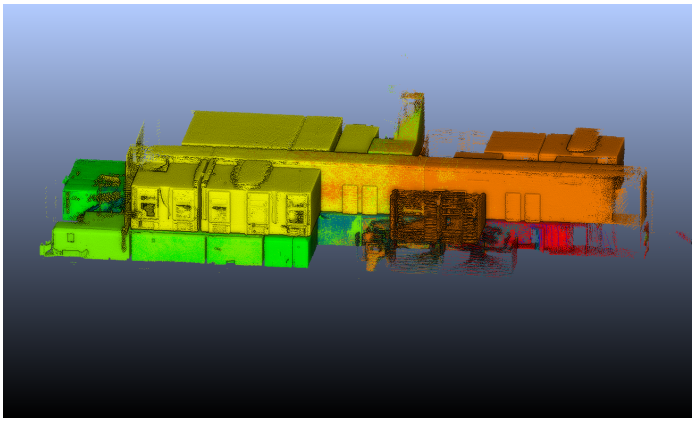


Figure 5.2: Test point cloud, captured in one of the buildings of Technische Universität Braunschweig, Germany

5.2 IMPLEMENTATION

5.2.1 Voxelisation and walkable space

The walkable voxel detection algorithm making use of octree generation is described in Section 2.2. It includes some parameters that influence the outcome.

VOXEL SIZE The ideal size for smallest leaves of the octree for the detection of walkable voxels is determined by Staats [2017] to be 7 cm, because this is small enough to detect stairs, and much smaller voxels will lead to longer processing times. The octree generation depends on input of octree depth, not voxel size. In order to get the right voxel size, the depth has to be adjusted to give the right output.

FILTERING As point clouds gathered with laser scanners often have some noise in the output, a filtering can be applied making use of the voxels. By leaving out voxels that don't have more than a certain number of points inside them, noisy voxels can be eliminated. This number has been tested by Staats [2017], but no specific value is recommended.

DYNAMIC OBJECTS A filtering is also applied to remove dynamic objects, such as people walking through a room. This is done by comparing two different time frames of the captured point cloud. When a room is passed by twice, for instance, it is checked whether all voxels generated from the first round are also generated again in the second round. This is done based on the timestamps of the points. The difference in timestamps is another parameter to be taken into account.

5.2.2 Wall sheet extraction

As discussed in Section 4.1 there are five main parameters to be taken into account when distinguishing between medial sheets inside walls, and medial sheets not inside walls, also expressed in Table 4.1.

In summary the separation angle θ of medial balls inside a wall sheet should be approaching 180°. The radii should have low median values (half the thickness of walls), and low standard deviations. Furthermore the z component of normal vectors of the MAT points should have a low median value, because they point sideways. Lastly the total number of points in a sheet should have a high value. This has been translated in a number of tests on a clipped part of the development point cloud, shown in Figure 5.3. The point cloud covers two floors, both having hallways with rooms adjacent to them.

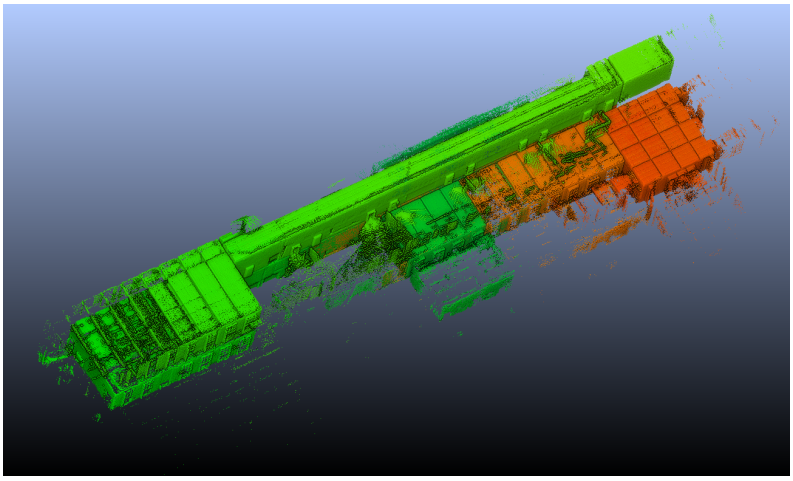
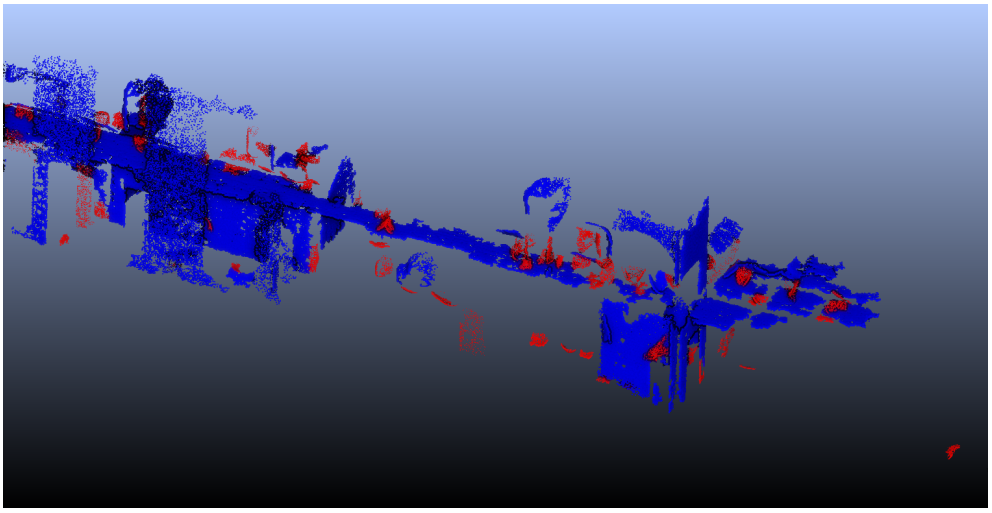


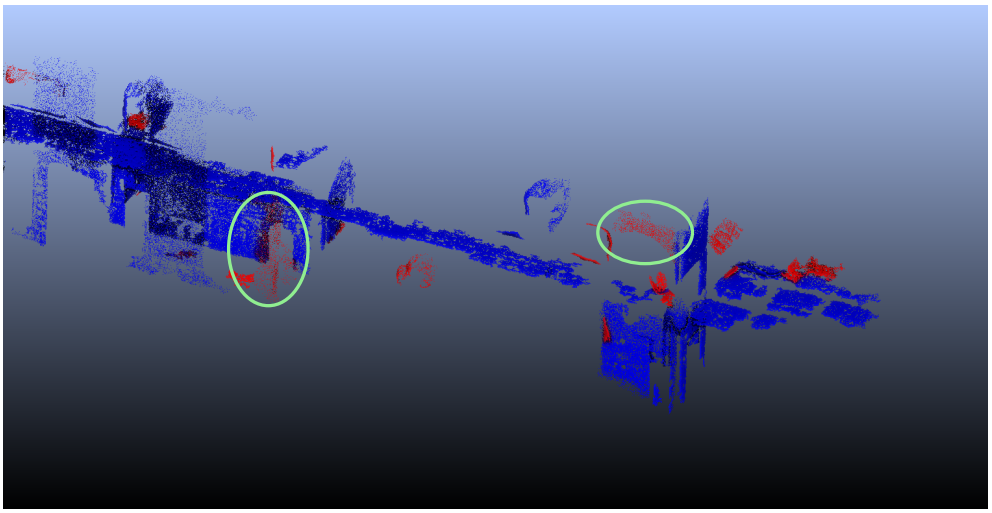
Figure 5.3: Clipped part of development point cloud, used for testing wall sheet generation

Test 1: Number of Points

One of the theoretical requirements is that the number of points in a sheet should be high. In this test it is determined what number is high enough. Firstly all sheets that have a radius smaller than 0.6 m are left out, to remove the most apparent noise. The results for sheets having less than 500 points gave much noise, so it is decided to show results for sheets that have number of points between 500 and 1000 in red, and more than 1000 points in blue in Figure 5.4a. In Figure 5.4b sheets with a number of points between 1000 and 1500 are shown in red, and above 1500 in blue. As can be seen, the first criteria filter out a lot of noise, and no wall sheets are removed. With the second criteria some (parts of) wall sheets are removed, so the best value for the parameter would be a minimum of 1000 points per sheet. It should be noted that these parameters are highly dependent on the density of a point cloud.



(a) In red sheets left out when nr of points is between 500 and 1000

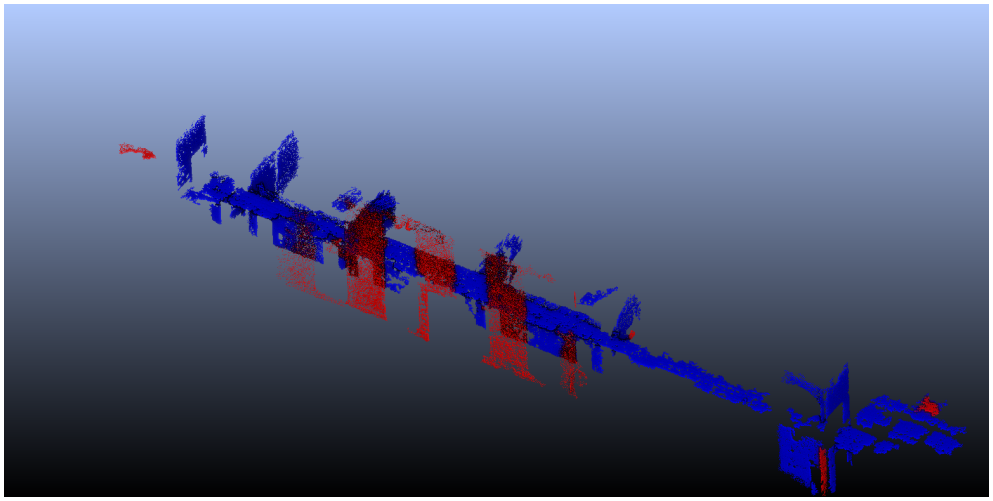


(b) In red sheets left out when nr of points is between 1000 and 1500

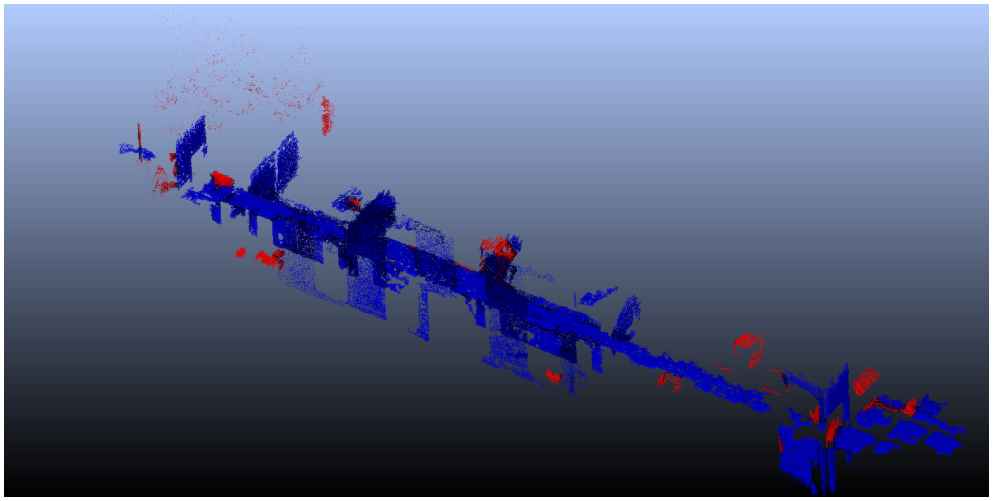
Figure 5.4: Tests executed for filtering on number of points (initial filter of median radius $<$ 0.6m)

Test 2: Median Radius

The second test done is by filtering the sheets based on median radius. As discussed the median radius should be comparatively small, because the sheets formed inside walls should have a median radius about the same size as half the thickness of the wall. Tests are shown in Figure 5.5, with initial filtering of sheets that have a number of points higher than 1000. These tests show that filtering of sheets with median radius between 0.2 and 0.4 m could leave out wall sheets, whereas a filtering with median radius above 0.4 m only removes non-wall sheets.



(a) Blue: sheets with median radius < 0.2 m, Red: sheets with median radius between 0.2 and 0.4 m

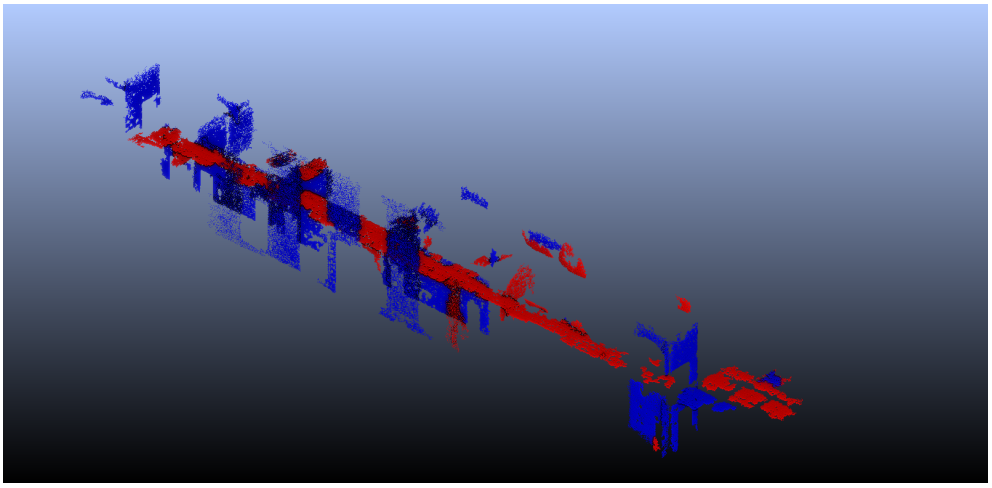


(b) Blue: sheets with median radius < 0.4 m, Red: sheets with median radius between 0.4 and 0.6 m

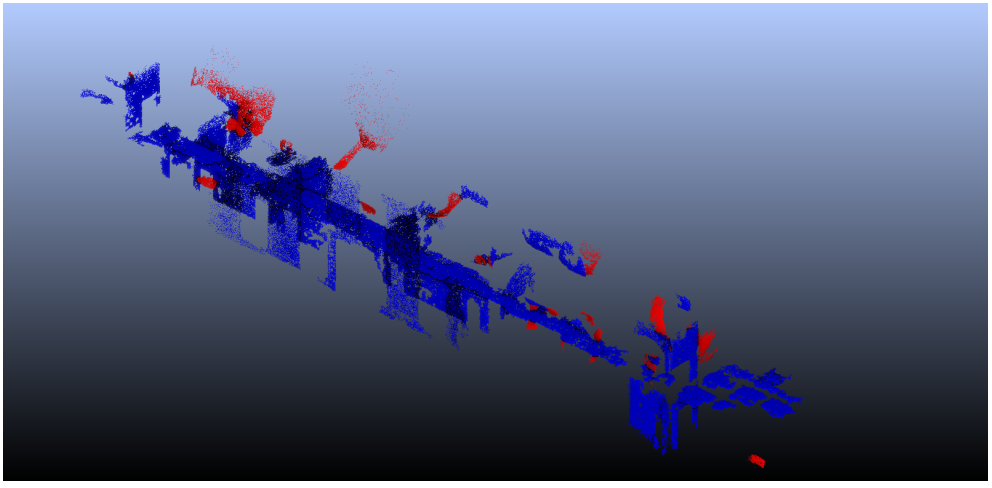
Figure 5.5: Tests executed for filtering on median radius, with initial filter of number of points > 1000

Test 3: Median separation angle

Theoretically the value of the separation angle of medial balls, θ, should be approaching 180° when in a wall sheet. Experiments have been done with initial filtering that sheets must have more than 1000 points. In the wall sheets the separation angle is saved in radians, so the maximum value is π radians. Test are done with θ having a median of minimum 0.9π, 0.8π and 0.7π. In Figure 5.6 the results of tests are shown. As can be seen in Figure 5.6a there wall sheets removed when filtering sheets with a median θ below 0.9π, but below 0.8π there are no wall sheets removed. The ideal value for θ would thus be 0.8π radians.



(a) Blue: sheets with median θ having a median of minimum 0.9π, Red: sheets with median θ between 0.8π and 0.9π

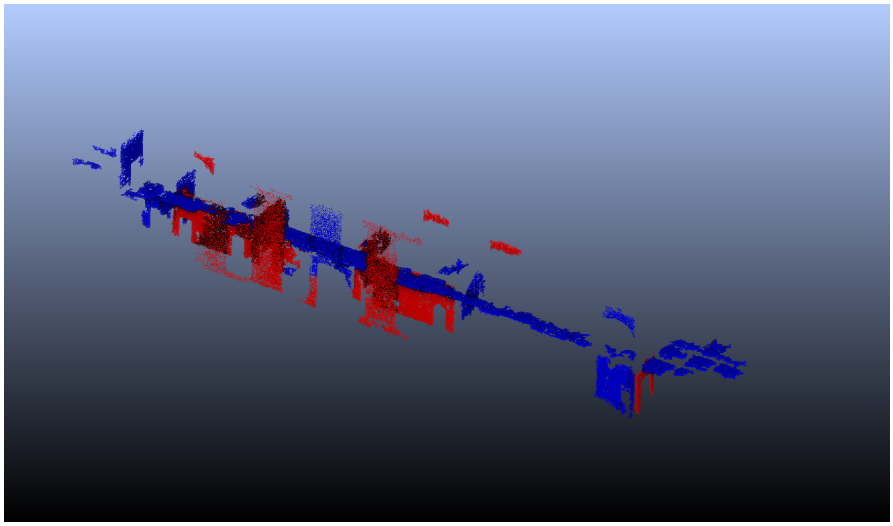


(b) Blue: sheets with median θ having a median of minimum 0.8π, Red: sheets with median θ between 0.7π and 0.8π

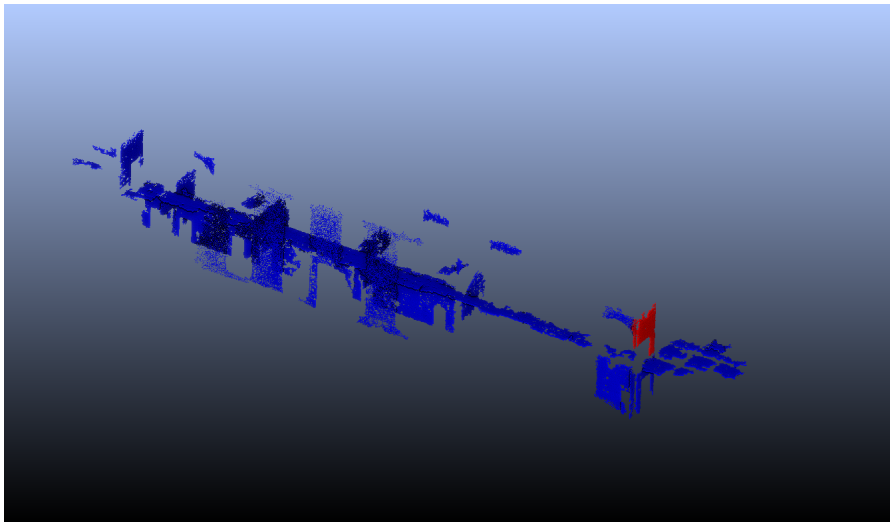
Figure 5.6: Tests executed for filtering on median separation angle, with initial filter of number of points > 1000

Test 4: Standard deviation radius

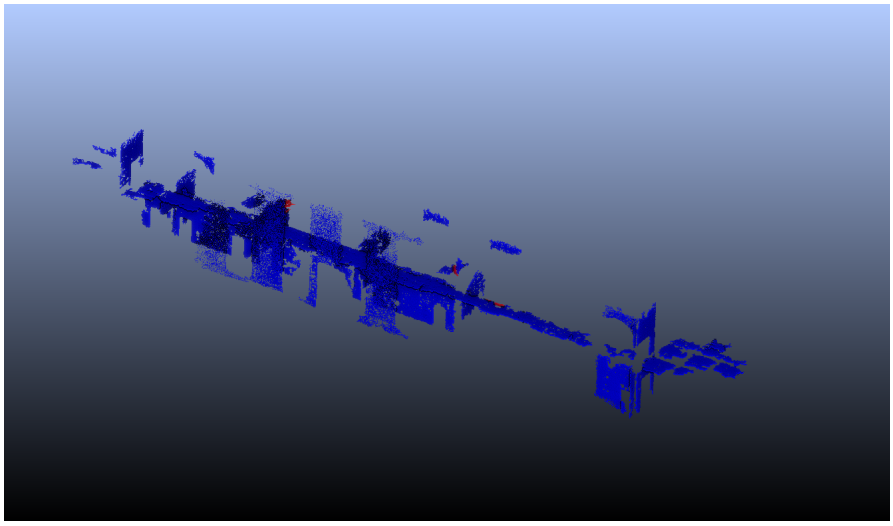
The fourth test for wall sheet filtering is done based on the standard deviation of radii in sheets. As discussed the smaller the standard deviation, the larger the chance that a MAT sheet is inside a wall. The results of the tests executed are shown in Figure 5.7. It can be seen that when filtering sheets with a standard deviation of the radius below 0.03 m will filter out wall sheets, whereas above 0.04 m the filtering has no effect on the wall sheets any more. The value of this parameter is thus set at 0.04 m.



(a) Blue: sheets with the standard deviation of the radius < 0.02 m, Red: sheets with standard deviation of the radius between 0.02 and 0.03 m



(b) Blue: sheets with the standard deviation of the radius < 0.03 m, Red: sheets with standard deviation of the radius between 0.03 and 0.04 m



(c) Blue: sheets with the standard deviation of the radius < 0.04 m, Red: sheets with standard deviation of the radius between 0.04 and 0.05 m

Figure 5.7: Tests executed for filtering standard deviation of radius in sheets, with initial filter of number of points > 1000

Test 5: z-value normal vector n

The final test done is to find the optimal value for the absolute value of n_z, the z component of the normal vector of the points. This is done to distinguish between wall and floor sheets. The initial filtering for this combines all previously found parameters in test 1 to 4, to keep all sheets in walls and floors at first. There is a big gap in the z-value of normal vectors in all filtered sheets, because the sheets are either floor-, or wall sheets. This is shown in the histogram in Figure 5.8. The limit for n_z can be put anywhere between 0.2 and 0.9. The results of testing with a value of 0.5 for n_z is shown in Figure 5.9.

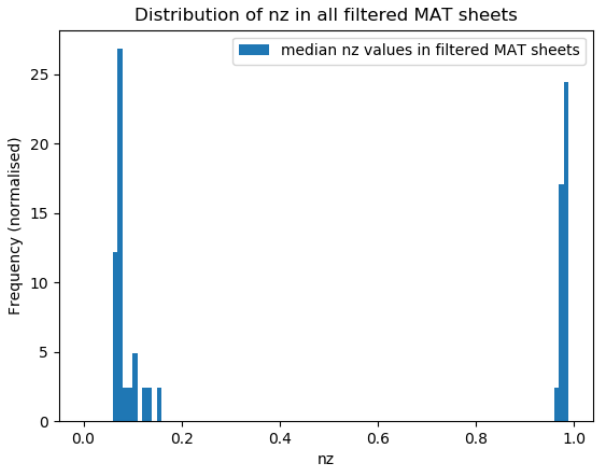


Figure 5.8: Histogram showing dispersion of median n_z values in filtered sheets

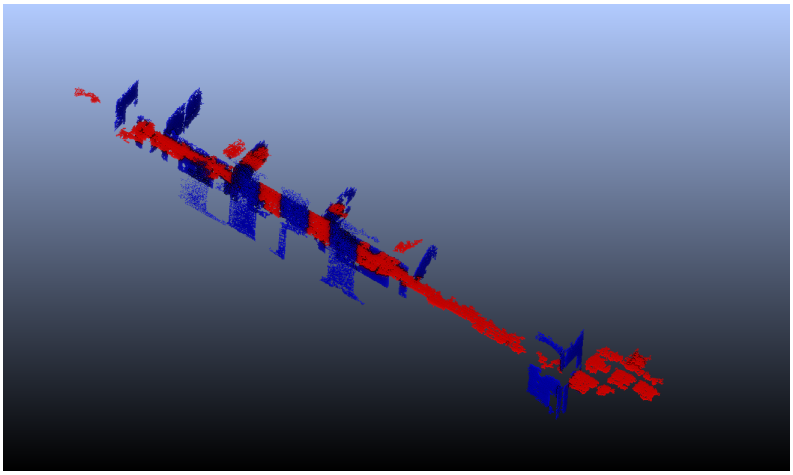


Figure 5.9: Blue: sheets with the absolute value for $n_z < 0.5$, Red: sheets with the absolute value for $n_z > 0.5$

A summary of the found values for each parameter is shown in Table 5.2.

Parameter	**Value in MAT sheet**
Total number of points	> 1000
Median radius r	< 0.4 m
Separation angle θ	$> 0.8\pi$ radians
Standard deviation radius r	< 0.04 m
Z value normal vector n_z	Between 0.2 and 0.9

Table 5.2: Values for parameters for filtering MAT sheets

5.2.3 Doorway detection

GEOMETRIC DOOR DETECTION

The geometric door detection algorithm, as explained in Section 4.2.1, is implemented in Python. A plane is fitted through all points belonging to one MAT wall sheet. The trajectory is tested for intersections with this plane. Using the plane parameters the wall and intersection points are rotated to the horizontal plane, so the z value can be temporarily removed. A convex hull is fitted around the 2D wall points, and it tested for the intersection points whether they lie inside this convex hull. The *ConvexHull* function from scipy.spatial is used for this. For a wall sheet spanning two floors, with four door-shaped gaps, three locations of doors are found (Figure 5.10). This is because the fourth door is a closed portal, and was not walked through (Figure 5.11).

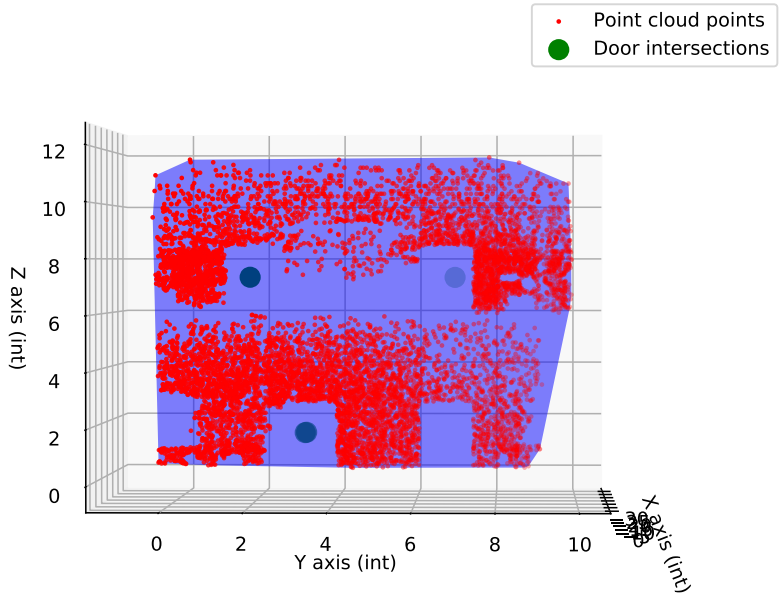


Figure 5.10: Three doors are found when intersecting the convex hull of a MAT sheet with the trajectory

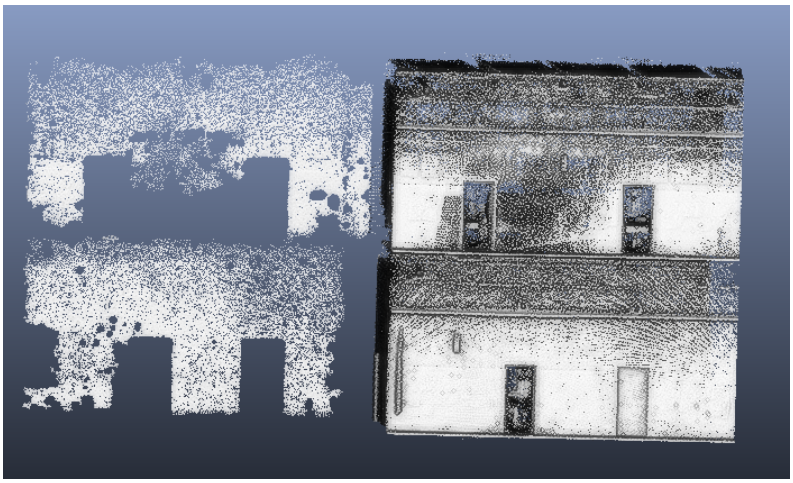


Figure 5.11: A MAT wall sheet (left) generated in a point cloud (right) with one closed door

However, the method is not flawless. An example where this method does not find a door that has been walked through is given in Figure 5.12. Moreover, if a

doorway is of the same height as the ceiling, first the wall sheets to the right and left side should be found to create a convex hull for, adding another step of complexity. Because of the limits that this method poses, and the fact that it needs an extra step to separate the voxelised walkable space - this method has not been included in the final results, and only the voxelised door detection has been regarded.

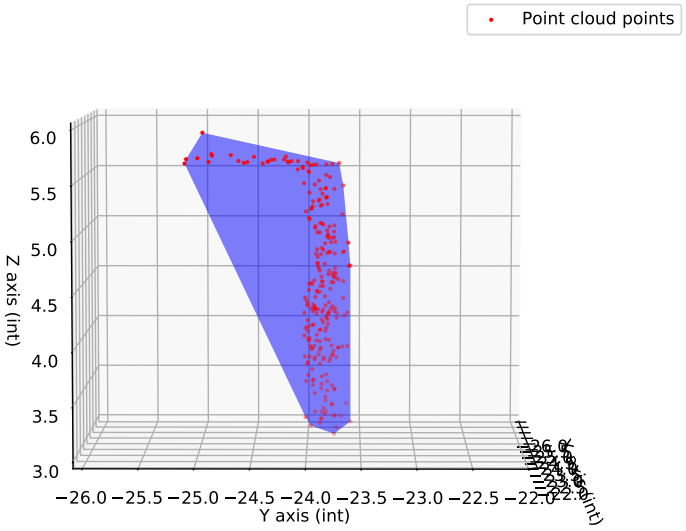


Figure 5.12: When a wall sheet does not cover the whole door frame, a door is not detected when testing for intersection

Voxelised door detection

The voxelised door detection method, as explained in Section 4.2.2, is developed for doors with a MAT wall sheet above them (upwards-looking method), and doors that are of the same height as the ceiling (sideways-looking method). Both are implemented by executing database statements from Python, making use of the psycopg2 module. For the first method an example of finding voxels of the original point cloud below the MAT wall sheets is shown in Figure 5.13. These voxels are given an attribute in the database that marks them as unfit to be used in the region growing algorithm. There are also some voxels found inside walls, but this does not have any effect on the region growing, because they are not on a floor anyway. Besides detecting voxels on the floor inside a doorway, also the locations of doors in the trajectory are detected, which are later used for node creation inside doors.

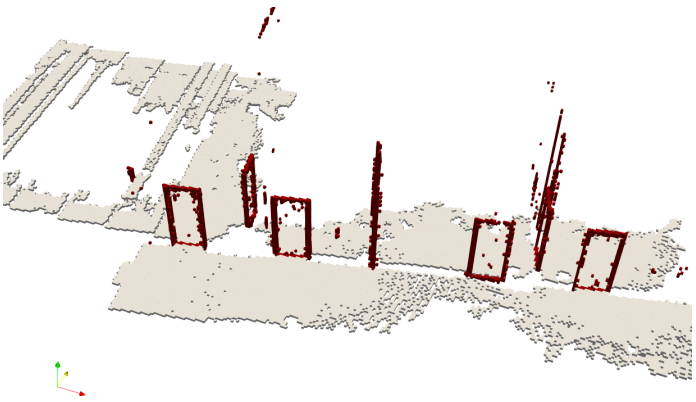
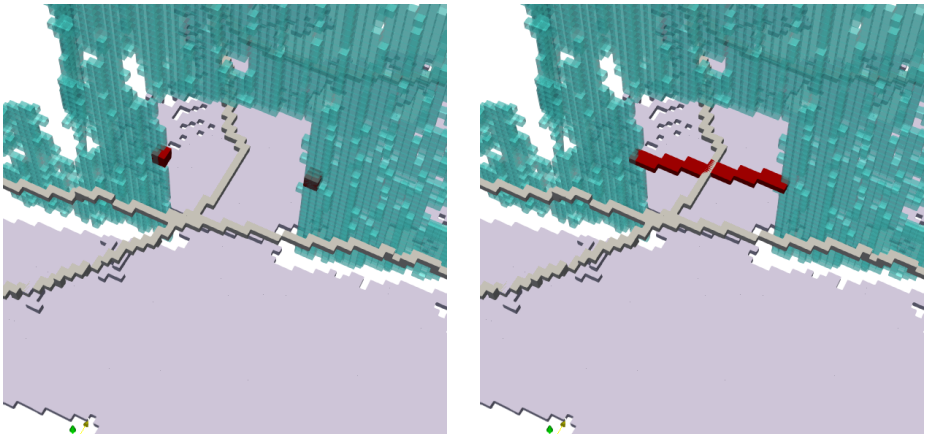


Figure 5.13: Voxels (in red) marked as unfit to be used in region growing algorithm, because they could be inside door frames

Although the development point cloud had no examples of doors being of the same height as the ceiling, the second method, which looks at both sides of the trajectory to find potential doors, is tested as well. An example is shown in Figure 5.14a, where two voxels are detected in the MAT sheet that lie on opposite sides of each other in a door frame. A line is generated between the two points in Figure 5.14b. The voxels of the original point cloud that lie directly beneath this line are also marked as unfit for region growing, making sure they are not taken into account for this next step. The location of this door is marked in the trajectory as well.



(a) Points that are opposite one another in a door frame, having the shortest distance to one another

(b) Line with 8-connectivity generated from those points

Figure 5.14: Line generation for voxel detection on floor in doorways

5.2.4 Connectivity graph

Connectivity relationships need to be established in the indoor environment by looking at which two spaces are connected by a door node, and which spaces are connected to both the top or bottom of a stairway or slope. As described in Section 4.4.2 stairs and slopes get a node at their top and bottom, for which connectivity relationships are established. This is done based on the locations where the trajectory changes from floor to slope or stair, and vice versa. After this walkable voxels are found by applying the region growing algorithm, as explained in Section 2.2.2. It is executed on a part of the development point cloud, which also incorporates slopes, stairs and different floor levels (Figure 5.15).

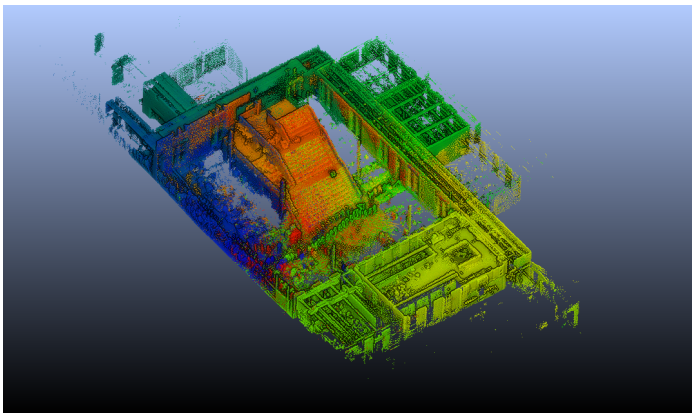


Figure 5.15: Point cloud used for testing the connectivity graph creation on

The walkable voxel region growing algorithm is executed with the restriction that voxels that were detected inside door frames cannot be used. This generates clusters representing separate indoor spaces (see Figure 5.16). For every door, stair and slope it is then determined which two spaces it is connecting. This information is represented in the connectivity Node-Relation Graph (NRG), shown in Figure 5.17. The nodes in rooms are defined at convenient locations for visualisation. As can be seen, the resulting graph is not useful for navigation yet. Both the large corridor and hall, which are respectively shown as dark blue and yellow in Figure 5.16, only have one node, making routes through these spaces differ a lot from reality. This shows that subspacing is necessary for the connectivity NRG, of which the results are shown in Section 5.2.6.

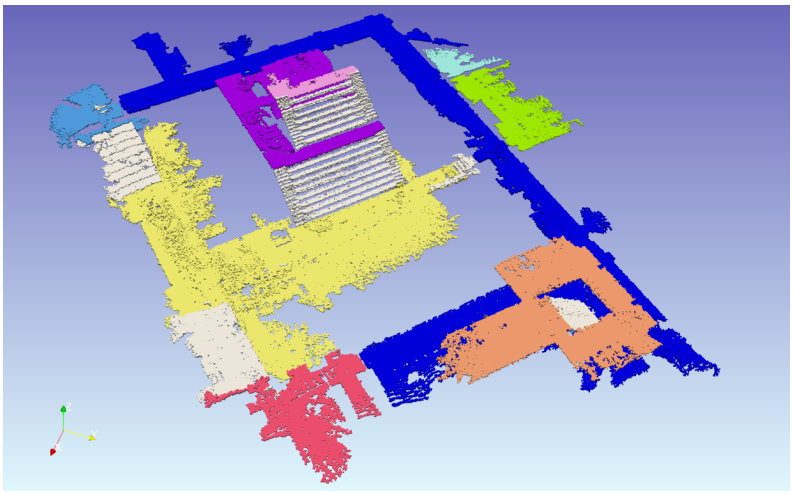


Figure 5.16: Result of walkable space detection where all separate rooms and corridors are marked in different colours, and stairs/slopes shown in light grey

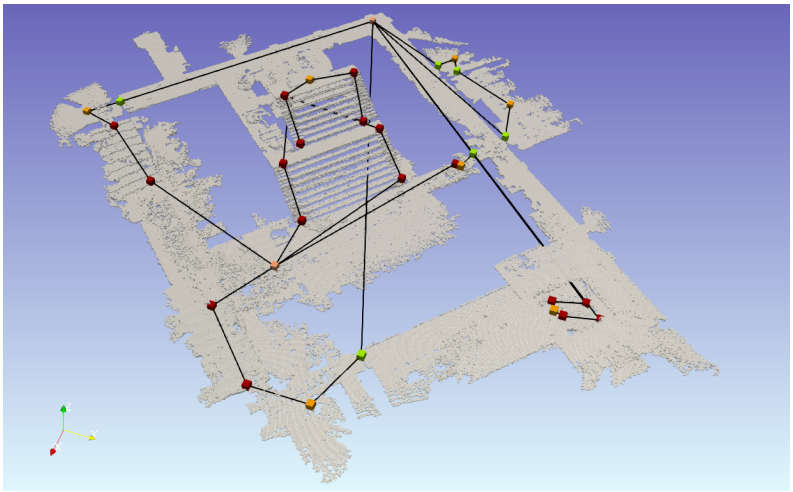


Figure 5.17: Connectivity graph of an indoor point cloud (red = stair or slope, green = door, orange = room)

The red region in the bottom left of Figure 5.16 actually consists of two separate spaces. A door location has been detected between them, but not enough voxels have been removed between the two spaces, which leads to them still being connected in the doorway (see Figure 5.18). This has been overcome for the subspaced connectivity graph (see Figure 5.24) by creating a buffer around the voxels inside the doorframe that were removed, and removing more voxels inside this buffer.

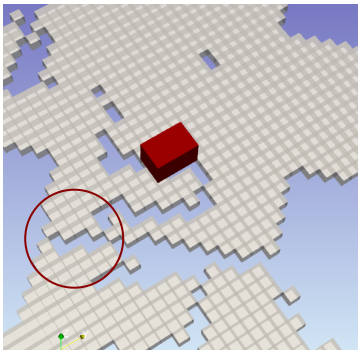


Figure 5.18: Example of two rooms not being separated enough

5.2.5 Geometry modelling for floor voxels

In order to get a 2D polygon representing the floor space of every room and corridor in the network, an α-shape is modelled around all voxels belonging to each indoor space. An α-shape is very dependent on the parameter α, that defines which triangles of the Delaunay triangulation must be kept in the point cloud, and which can be discarded. Therefore multiple values for this parameter are tested on a set of voxels belonging to a corridor that has both an L-shape and a T-junction (Figure 5.19). The best value of α also depends on the shape of the 2D MAT it will create in the resulting polygon, which is discussed in Section 5.2.6.

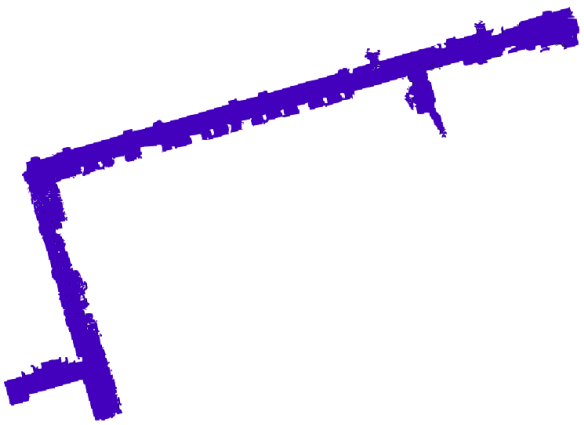


Figure 5.19: The voxels belonging to the corridor used to test the best value for α

α-shape creation is implemented in Python, using the Delaunay triangulation function from the scipy.spatial package, and filtering triangles based on whether size of their circumcircle is smaller than α. For different values of α the results in Figure 5.20 are obtained. It can be observed that a polygon created with an α value of 5 will be very sharp around the edges of corners, which is an advantage, but it also produces more noise in the rest of the corridor, for instance at locations with small gaps in the data. An α value of 15 does not cover more gaps, but instead produces more noise, which makes the value of 10 a good option for approximating the shape of a hallway. It takes 8.6 seconds to create this shape using the script in Python.

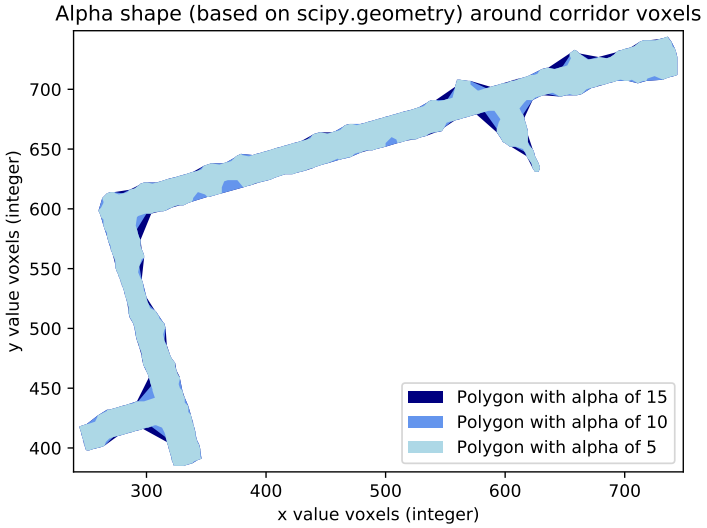


Figure 5.20: Polygons as result of different values for α

Besides the conventional way to create the α-shape – by applying Delaunay triangulation and filtering the triangles based on their circumcircle – the *ST_ConcaveHull* function of PostGIS is also tested. This function does not filter triangles based on α, but produces a shape that has an area at 'target percentage' of the convex hull. This target percentage is the variable that can be changed in the function. As shown in Figure 2.8 the area of the output shape decreases with smaller α, although not linearly. The results of *ST_ConcaveHull* with different target percentage values are shown in Figure 5.21. The shapes look very much alike, although they should be differing in area. The documentation[13] notes that the result often 'overshoots', so this could be the cause of the shapes looking much alike. When low values for the target percentage are used the processing time increases rapidly, but no results are obtained that represent the geometry of the corridor in an according way. The time it takes to create the shape with target percentage of 0.1 is 1025.24 seconds, so approximately 17 minutes. Since the output shape and processing time of the α-shape in Python are both better, it can be concluded that for this purpose the α-shape creation in Python is a better option than the concave hull creation in PostGIS.

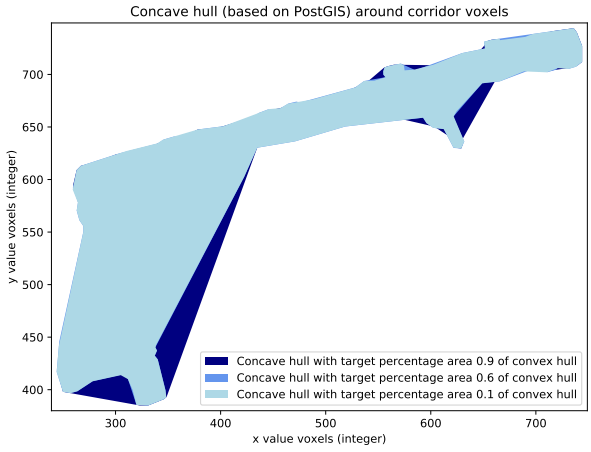


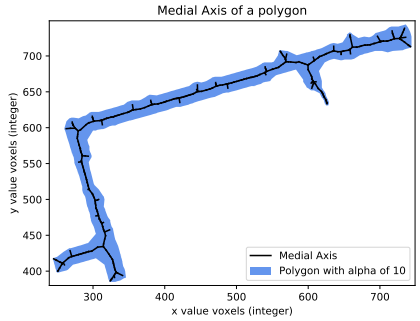
Figure 5.21: Polygons as result of different target percentages in *ST_ConcaveHull*

13 https://postgis.net/docs/ST_ConcaveHull.html

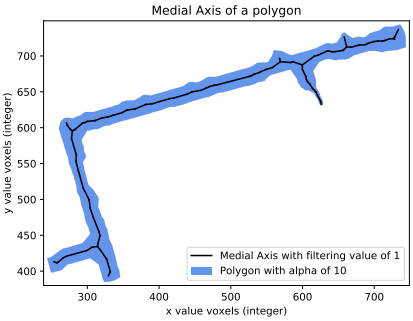
5.2.6 Subspacing the network

In order to subspace the nodes in corridors the 2D MAT is created for the geometry representing the corridor (see Section 2.4.1). The resulting skeleton should not have too many branches, to prevent nodes being formed at locations where they are unnecessary. On the other hand could the lack of branches cause nodes not to be formed at junction points and in L-turns. An optimal value for filtering the shape of the 2D MAT is discussed in this Section.

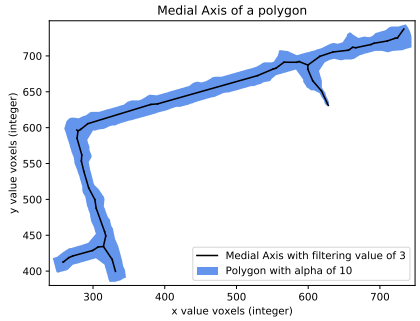
The MAT is produced for the polygon created with an α value of 10. For this the *ST_ApproximateMedialAxis* function of PostGIS is used (Figure 5.22a). As can be observed, the unfiltered result has too many for the purposes of this research, and should be filtered. A filtering function of PostGIS is used for this: *ST_Simplify*. This function uses the Douglas-Peucker algorithm (see Section 2.3.2) to simplify the polygon first, after which the MAT is defined for it. The results of this are shown in Figure 5.22b-5.22d. With a filtering value of 1 there are still some branches at unwanted locations, whereas with a filtering value of 3 the branch in the L-turn is still kept, but no branches due to noise are present. When a value of 4 is used the branch in the L-turn disappears.



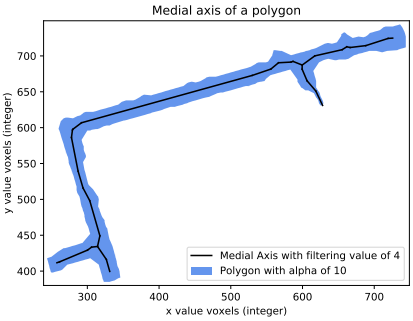
(a) Medial axis of the corridor without filtering



(b) Medial axis of the corridor with Douglas-Peucker filtering value of 1



(c) Medial axis of the corridor with Douglas-Peucker filtering value of 3



(d) Medial axis of the corridor with Douglas-Peucker filtering value of 4

Figure 5.22: Medial axis of the corridor with Douglas-Peucker filtering

After the medial axis of the polygon representing a hallway is found, the closest location for each door on the medial axis is found and saved as a node. Coordinates in the medial axis that have three or more connections, such as junction points in corners, are also assigned a node. The connection between these nodes is determined, and all are saved in the subspaced connectivity graph. The results of applying this on the corridor in the point cloud that was used to create the connectivity graph for (marked in blue in Figure 5.16) are shown in Figure 5.23.

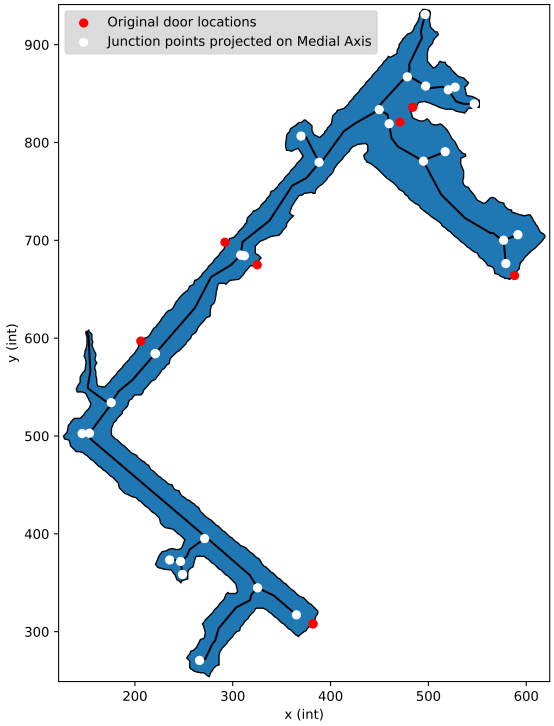


Figure 5.23: Results of subspacing a corridor by projecting the location of adjacent doors on the medial axis

Finally, when applying subspacing on the connectivity NRG that is shown in Figure 5.17, the results are as shown in Figure 5.24. The routes represented by this graph are much more realistic than the non-subspaced graph, although the 2D MAT does not provide the best subdivision of the large hall (shown in yellow in Figure 5.16). In a wide open hall there are multiple routes a person can take, instead of only the centre line.

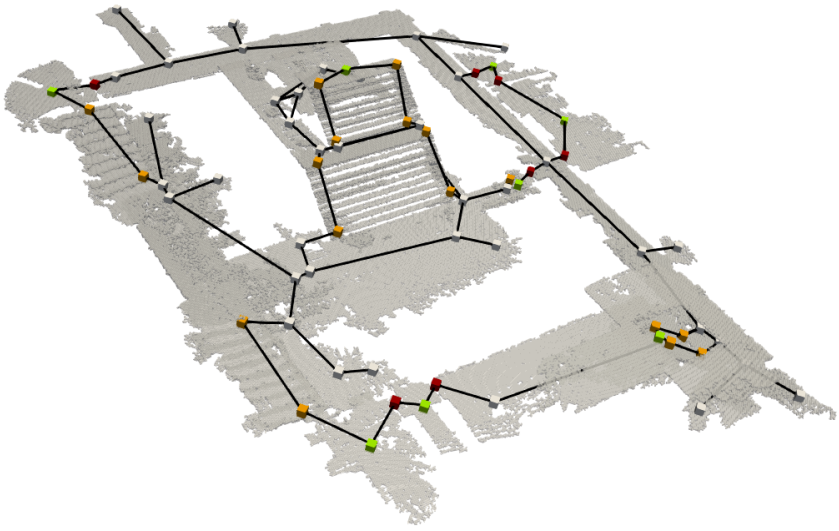
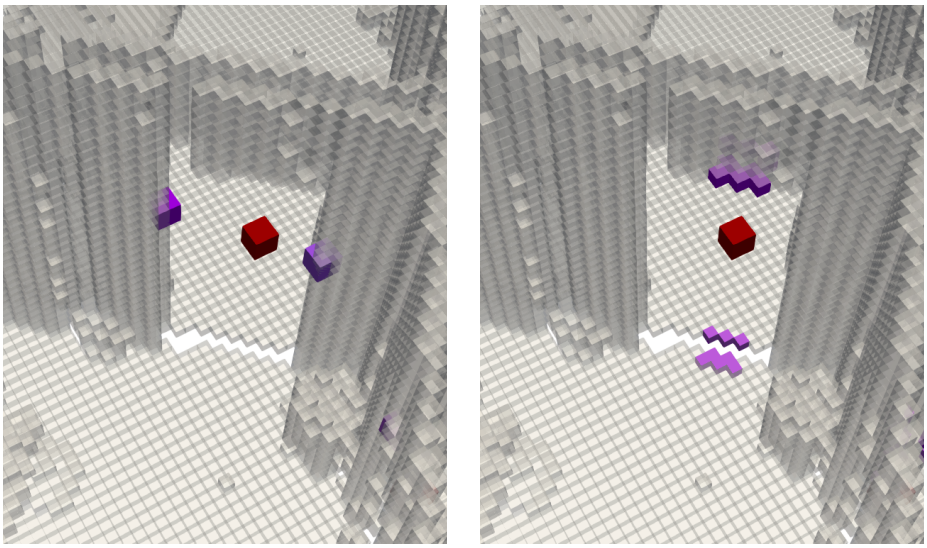


Figure 5.24: Subspaced connectivity graph of an indoor point cloud (green = room/corridor, orange = stair, red = doorway, white = subspaced node)

5.2.7 Accessibility information

Besides information on the location of stairs and slopes in the network, the width and height of doors are also extracted as accessibility information (see Figure 5.25). The door height is obtained by finding the closest voxels in the original point cloud in the positive and negative z direction from the door node. The same algorithm as described for finding MAT wall sheet voxels **around** the trajectory (see Section 4.2) is used in finding the closest voxels on opposing sides of a door frame, from which the door width can be obtained. The height and width are calculated in terms of voxels, and are converted to meters by multiplying the number of voxels in the height or width by the voxel size. This information is saved in the database table that contains the location of the doors.



(a) Points found in the width of the door, used for calculating door width

(b) Points found in the height of the door, used for calculating door height

Figure 5.25: Door width and height extracted for the accessibility NRG (red= door node, purple = point cloud voxel)

5.2.8 Structure of the database

For every step in the methodology that deals with voxels and the final network, the results are stored in a database, from which the final connectivity NRG can be retrieved. The structure of database and dependencies of tables is shown in a Unified Modeling Language (UML) diagram in Figure 5.26, only for the tables needed in a non-subspaced graph. The *Stairs&Slopes* table has an *id*, which is unique to the node, a *stair_id*, that refers to the stair or slope it belongs to, and a *stair_route_id*, existing for the situation when a staircase is very wide and needs multiple routes across it. A stair is connected to one space or door, and the other part of the *stair_route_id*. A door is connecting two spaces, which can be either *Stair&Slope* spaces, or *Rooms*. The name *Rooms* refers to either corridors or rooms; indoor cells that have a flat floor.

The table that contains information on subspaced nodes is added in Figure 5.27, and relationships of the three other tables with the *Subspaces* table are shown. A record in the *Subspaces* table refers to which room it belongs to, and saves an array of neighbours for that node. This array can contain *Stair&Slope* nodes, *Door* nodes or other *Subspace* nodes. When a subspaced connectivity graph is constructed, spaces listed as connections for the *Doors* or *Stairs&Slopes* table are first checked whether they occur in the *Subspaces* table.

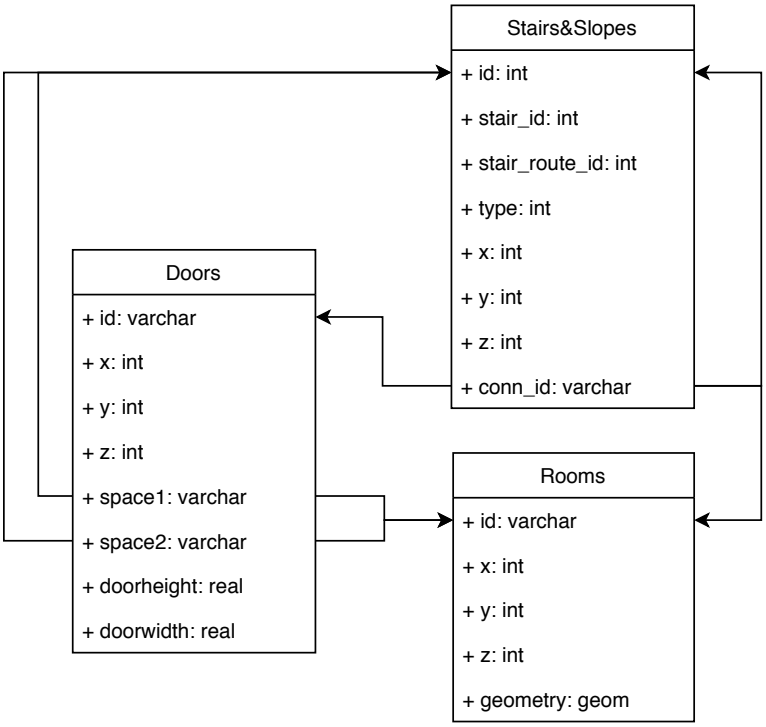


Figure 5.26: Relationships between three tables making up the non-subspaced connectivity graph

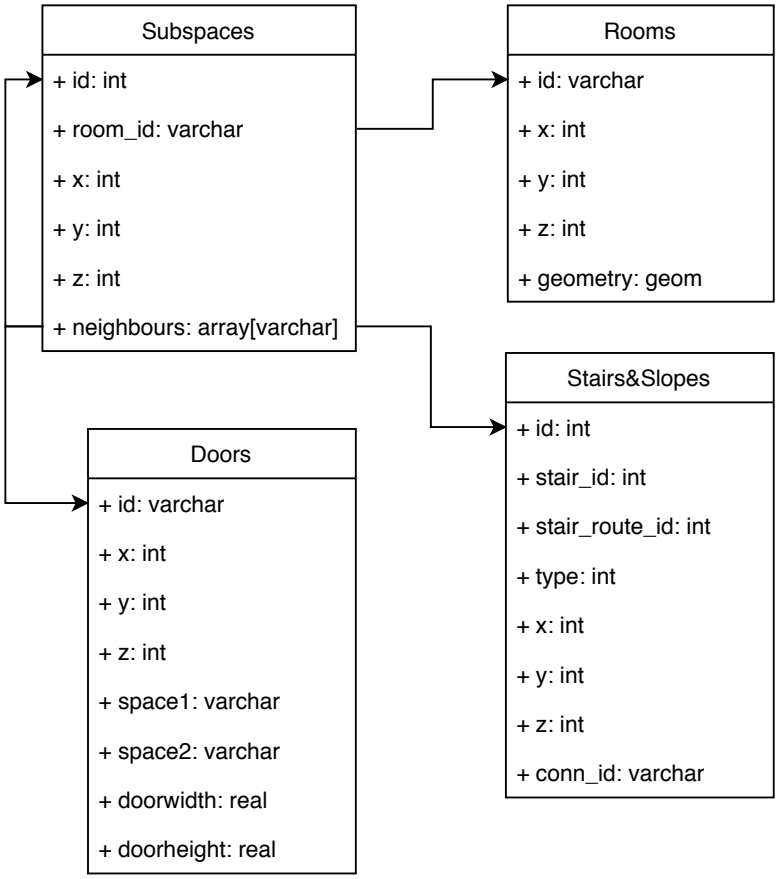


Figure 5.27: Dependencies of the Subspaces table in the connectivity graph

5.3 EXPERIMENTS

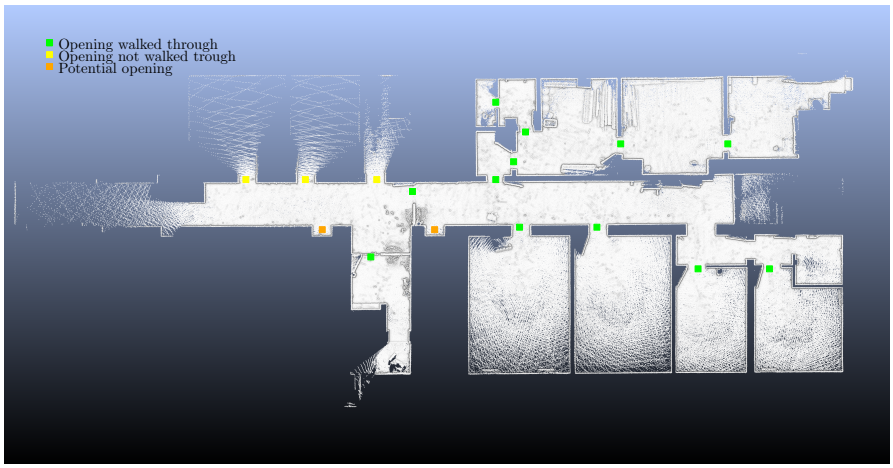
In this section the experiments used to evaluate the functioning of the methods on the test point cloud are described.

5.3.1 Door detection

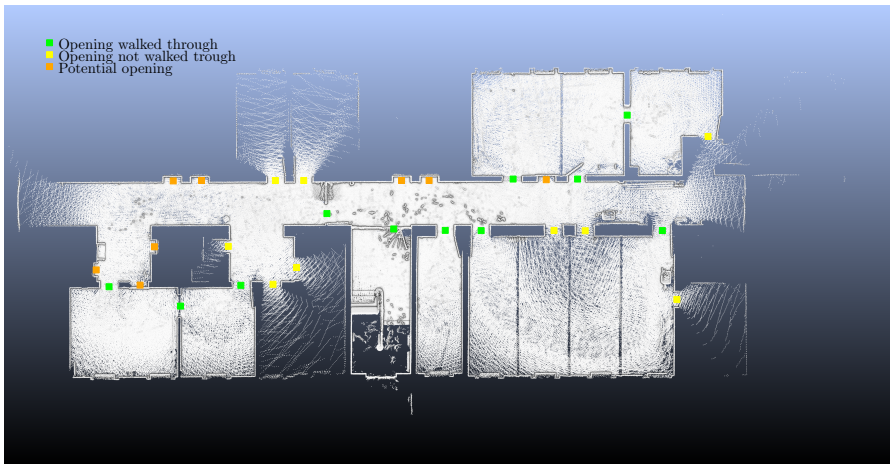
One metric for evaluating the door detection method is by calculating the percentage of correctly detected doors. An analysis is done on the following:

1. How many correct doors are identified?
2. How many doors are identified that are not really there?
3. How many doors are not identified, but are actually there?

In order to do this correctly doors in the test point cloud have been manually detected. For the test point cloud, all doors are identified, as shown in Figure 5.28. With the methodology proposed in this thesis only the doors that are walked through (marked in green) can be detected. For the doors in orange it is unsure whether these are openings that can be walked through, but they look like doors in the point cloud. In Chapter 6 the results of this experiment are shown.



(a) Doors indicated in 1st floor of test point cloud



(b) Doors indicated in 2nd floor of test point cloud

Figure 5.28: All doors manually indicated in the test point cloud dataset

5.3.2 Navigation network

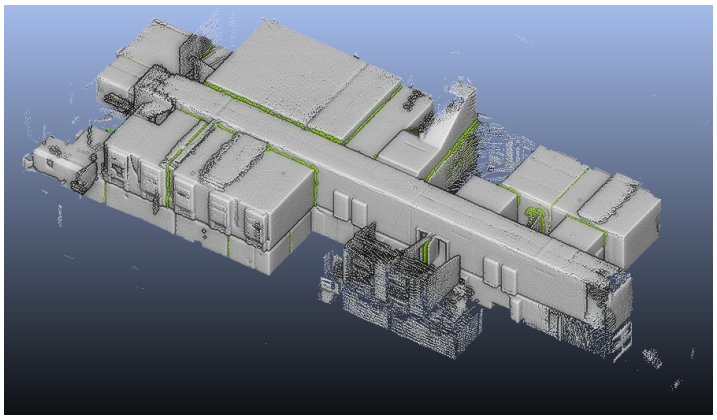
After doors are detected and the walkable space is divided into separate rooms and corridors, the navigation network is generated. In the connectivity NRG the structure of this network is already fixed. Each indoor space should have a node, as do all doorways, and their connectivity determines the edges. The subspacing of indoor spaces with multiple connections is something to be left to interpretation, because there are no strict guidelines found in IndoorGML. The choice of constructing a MAT in the corridors, and then creating nodes on this linestring at the location of doors, is not proven to be the best way of subspacing. This is why three individuals have been asked to draw a navigation network, with nodes in all doors, and at least one node in a room. The networks drawn are shown in Appendix A. They are compared with the results of subspacing on the test point cloud in Section 6.3.4, to see whether the way that nodes were drawn in corridors corresponds with the way the MAT subspaces a corridor. In order to do so, the location of nodes and the connections between them will be compared.

6 RESULTS AND EVALUATION OF TEST DATA

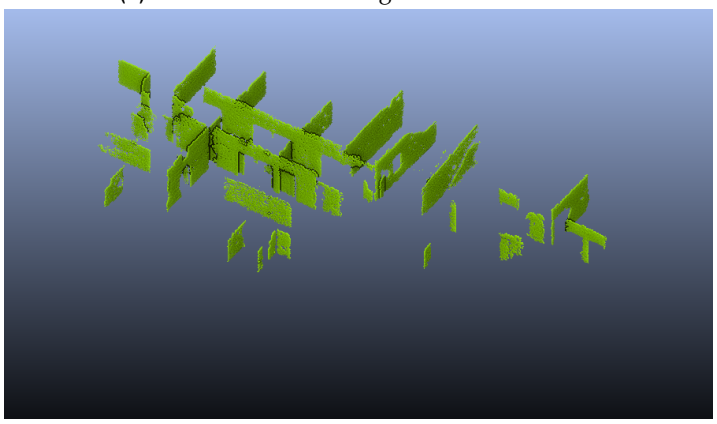
In this chapter the results of applying all methods on the test point cloud (Figure 6.1a) are presented and analysed. First of all the Medial Axis Transform (MAT) wall sheets that were generated for the point cloud are visualised and discussed in Section 6.1. After this the results of door detection based on these wall sheets is shown in Section 6.2. Stairs and rooms are found, and together with the doors the navigation graph is created (see Section 6.3). The results evaluated using the experiments defined in Section 5.3.

6.1 WALL SHEET EXTRACTION

The results of growing 3D MAT sheets and filtering them according to the parameters found in Chapter 5.2.2 are shown in Figure 6.1b.



(a) Point cloud used for generation of results



(b) Results obtained by applying MAT growing algorithm and filtering sheets based on parameters

Figure 6.1: Side-by-side comparison of test point cloud and results obtained by 3D MAT wall sheet generation

At first glance the sheets look indeed like they are grown in walls, and that they can be used for the further methods of this thesis. However, the results are not flawless compared to the sheets obtained with the development dataset. In most rooms in the test point cloud the doors are opened, and therefore partly occlude the wall behind them. No wall sheet can thus be grown in this wall, which could lead to those specific doors not being detected. An example is shown in Figure 6.2.

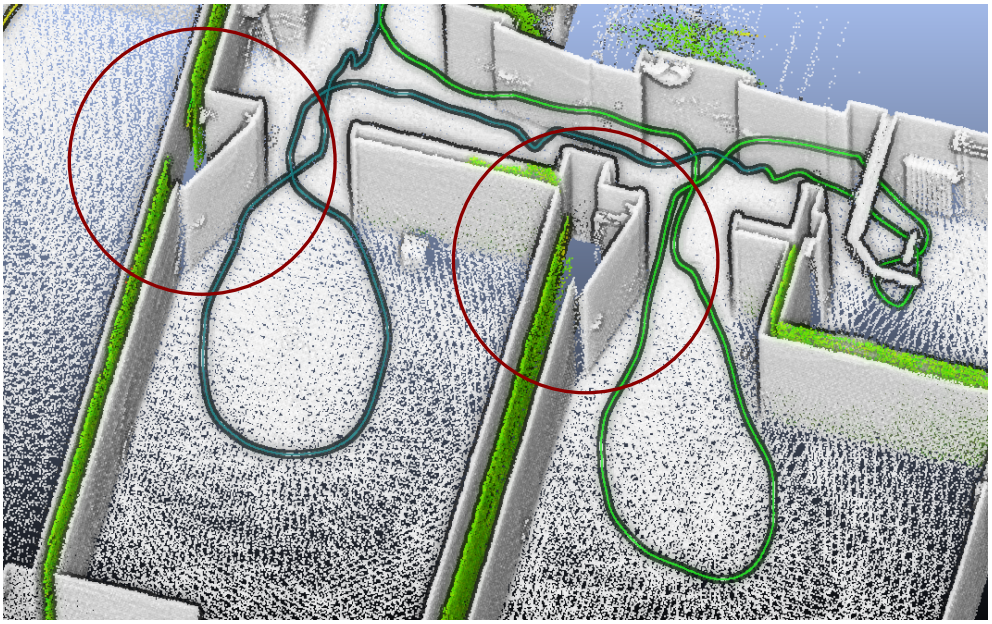


Figure 6.2: Wall sheets are not grown in walls that are blocked from view by a door

Besides this there is another situation in which medial sheets cannot be formed, shown in Figure 6.3. In the hallways on both floors there is a wall with a door in it subdividing the hall into two parts. These walls are so thin that medial balls cannot be formed inside them, and thus doors are not detected on that location. However, because the corridor is subspaced in a later step, the fact that these two doors are not detected is not crucial to the network. In both cases the door is connecting one part of the corridor to another, which could be seen as one space altogether anyway.

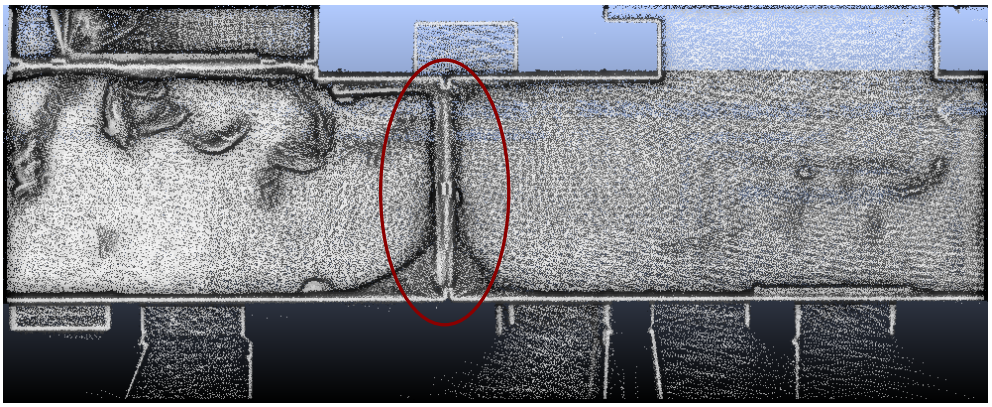
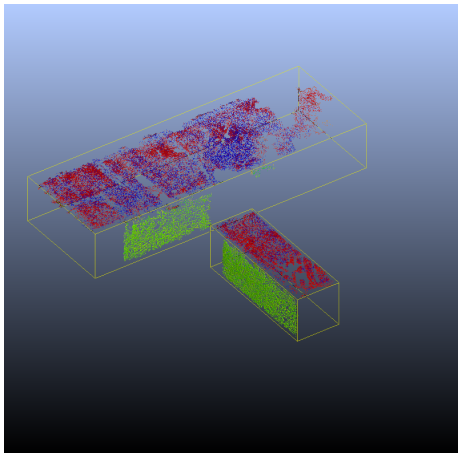
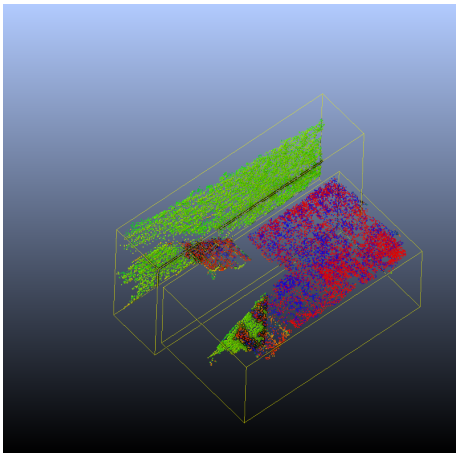


Figure 6.3: Wall sheets cannot be grown in a wall that is very thin

Another new phenomenon occurring is that some sheets are grown in walls continue to grow in an adjacent floor, or even staircase, causing the effects shown in Figure 6.4.



(a) Sheets that are partially grown in a wall, and partially in a floor

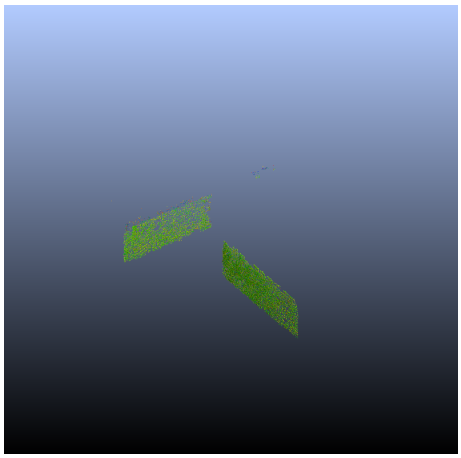


(b) Sheets that are partially grown in a wall, and partially in stairs

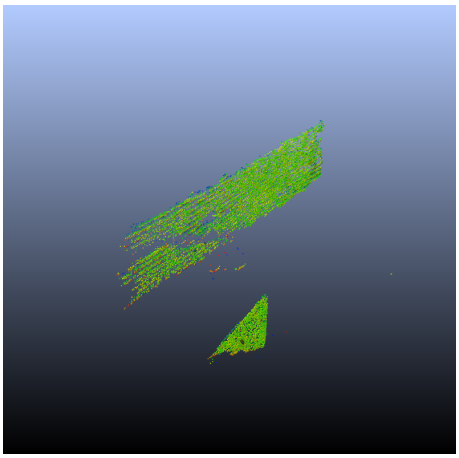
Figure 6.4: MAT sheets where filtering proposed in methodology does not work

One of the sheets in Figure 6.4b is filtered out in the initial results, because the biggest part is a floor and stair, whereas the other sheet, which is mostly wall, is present. For the examples that grow inside a floor, in Figure 6.4a, it is the same case.

In order to distinguish between walls and floors or stairs in these sheets, the direction of the normal vectors should again be taken into account, but now for individual points. These normal vectors are computed automatically in the medial balls that the points belong to. Making use of the normals, the sheet is not filtered in its entirety on the basis of median n_z, but all points are individually regarded. The results of this are shown in Figure 6.5. As can be seen, not all floor points are filtered, especially not at the edges, but the results are sufficient to continue on to the next step in the methodology; door detection.



(a) Points in wall/floor sheets individually filtered



(b) Points in wall/stair sheets individually filtered

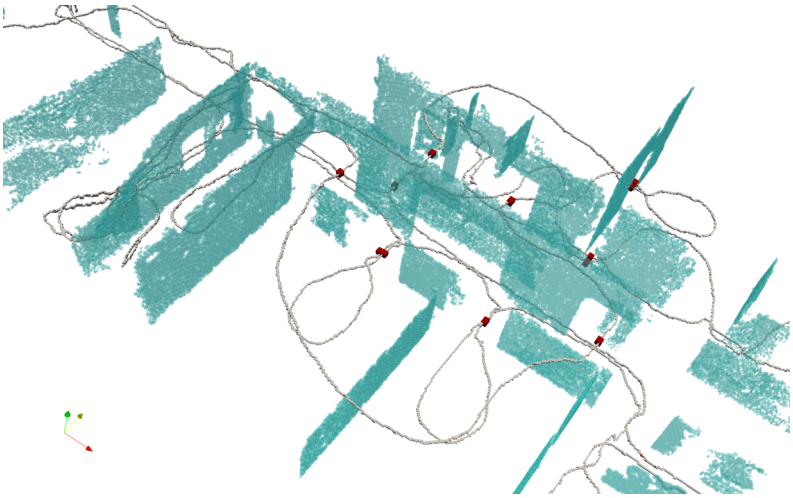
Figure 6.5: Individual points in wall, floor and stair sheets filtered based on n_z

6.2 DOOR DETECTION

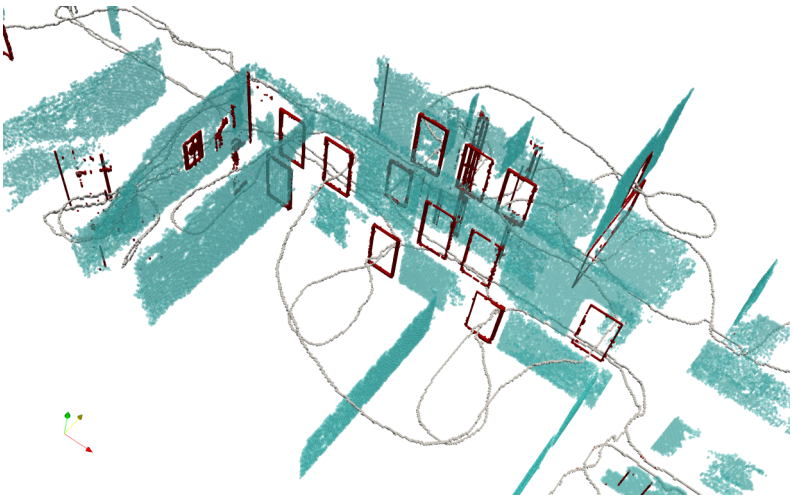
Using the results obtained in Section 6.1 both the upwards-looking and sideways-looking door detection methods are implemented. The results are evaluated based on the locations of doors that were manually defined in Section 5.3.1.

6.2.1 Upwards-looking door detection applied on wall sheet voxels

The location of doors in the trajectory are found by selecting all trajectory voxels that are directly below wall sheet voxels. The results of this are shown in Figure 6.6a. Besides this the voxels in the original point cloud that are below wall sheet voxels are marked as unfit for use in region growing, to create a gap in the floor space between two different rooms (Figure 6.6b).



(a) The locations of doors based on trajectory voxels in red

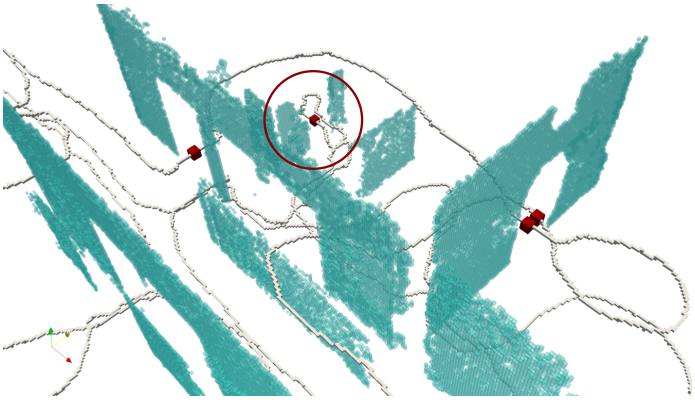


(b) The voxels of the original point cloud that lie inside the doorframe in red

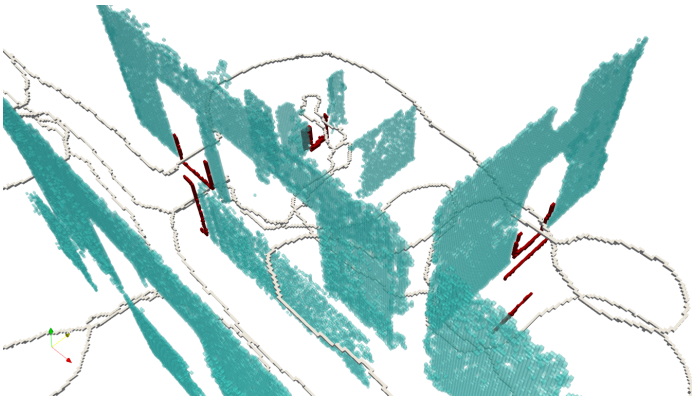
Figure 6.6: The results of upwards-looking door detection

6.2.2 Sideways-looking door detection applied on wall sheet voxels

Some results of the sideways-looking method are shown in Figure 6.7. The doorway marked in Figure 6.7a would not have been found without the sideways-looking method, because it does not have a wall above it.



(a) The locations of doors based on trajectory voxels in red



(b) The voxels of the original point cloud that lie inside the door-frame in red

Figure 6.7: Some results of sideways-looking door detection

Although it works for some doors, the sideways-looking door detection algorithm does not work as well as in the development dataset. This is caused by some wall sheets not forming in walls that are blocked by doors in the point cloud. This is also discussed in Section 6.1. Examples of doors that are left undetected are shown in Figure 6.8.

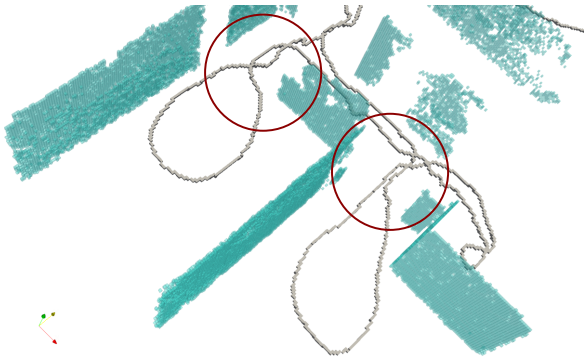
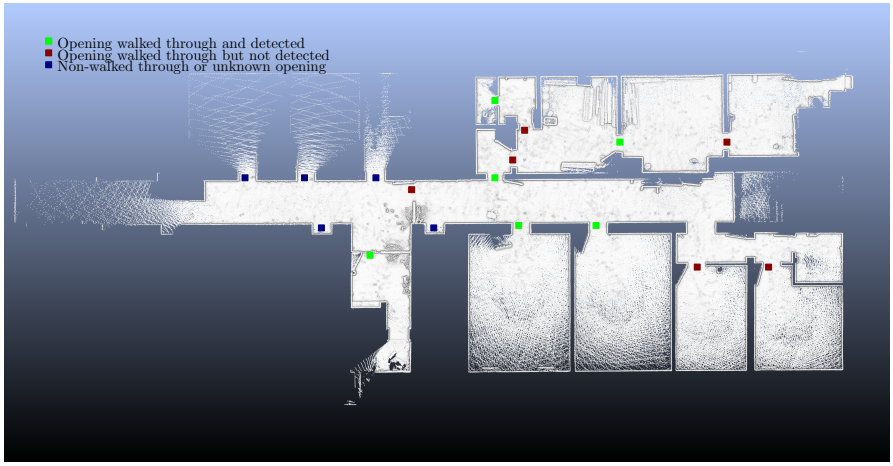


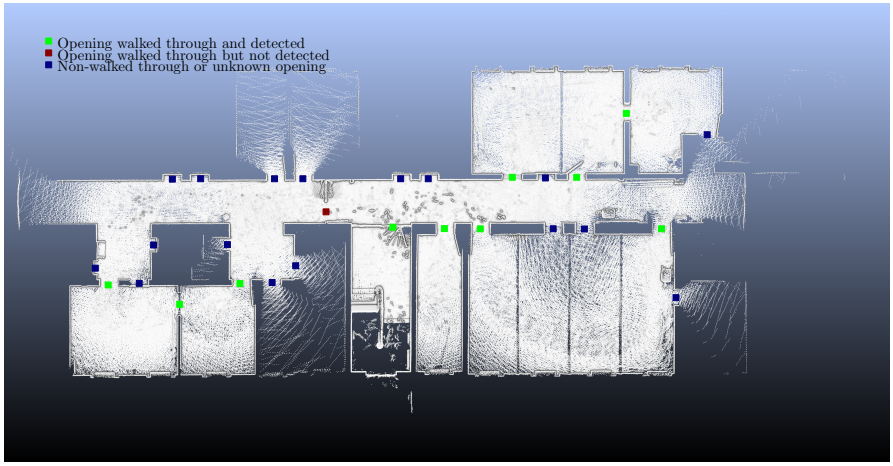
Figure 6.8: Doors that are left undetected when applying the methods proposed

6.2.3 Evaluating doors detected

A summary of which doors have been detected, and which doors have not is shown in Figure 6.9. It can be seen that the results on the second floor are better, which is caused by the fact that most of the doors on this floor had a wall sheet above them, except the one door in the thin wall in the corridor. On the first floor, where walls were often blocked by opened doors, not all doors could be detected. This issue is described in Section 6.1. There were no doors detected at locations where there are no doors.



(a) Doors detected in 1st floor of test point cloud



(b) Doors detected in 2nd floor of test point cloud

Figure 6.9: All doors detected by the algorithms in the test point cloud dataset

In Table 6.1 a summary is given of these parameters. 70% of all doors that were walked through are detected.

Opening	Number
Walked through detected	16
Walked through not detected	7
Not walked through	12
Unknown whether opening or inaccessible	10
Doors falsely detected	0

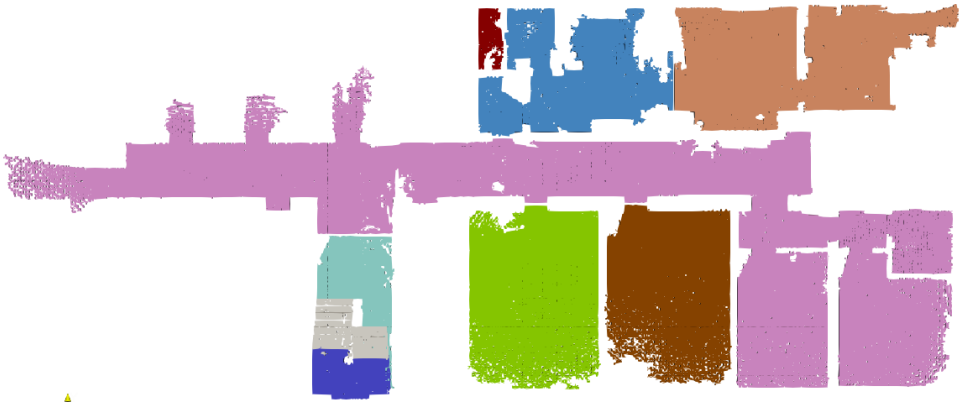
Table 6.1: Results of door detection

6.3 NETWORK CREATION

The connectivity network created for the test point is shown in this Section. First the non-subspaced connectivity Node-Relation Graph (NRG) is shown in Section 6.3.1, and in Section 6.3.2 the subspaced graph is presented. In Section 6.3.3 the retrieval of accessibility information is discussed.

6.3.1 Connectivity NRG

After applying the region growing of walkable space with the voxels inside door frames taken out (see Section 4.2), the different rooms and corridors in the voxelised model are detected. Together with the information on the location of doors, the connectivity NRG is created. There are two floors in the test point cloud, both consisting of a corridor with rooms adjacent to them. The result of colouring all voxels according to the id of their rooms is shown in Figure 6.10 for both floors. In the bottom centre of both floors grey-coloured regions, connected by a blue region, can be seen. These are the stairs that connect the two floors.



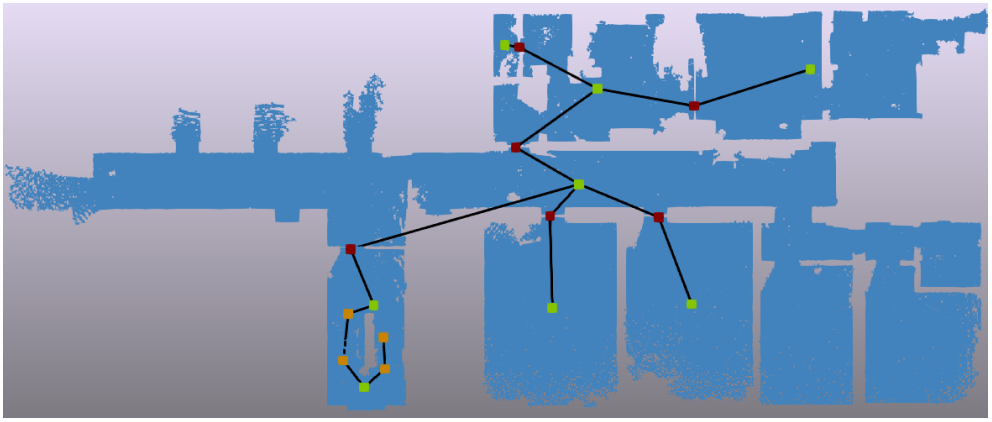
(a) Walkable voxels of floor 1 classified into different rooms, indicated with colours

(b) Walkable voxels of floor 2 classified into different rooms, indicated with colours

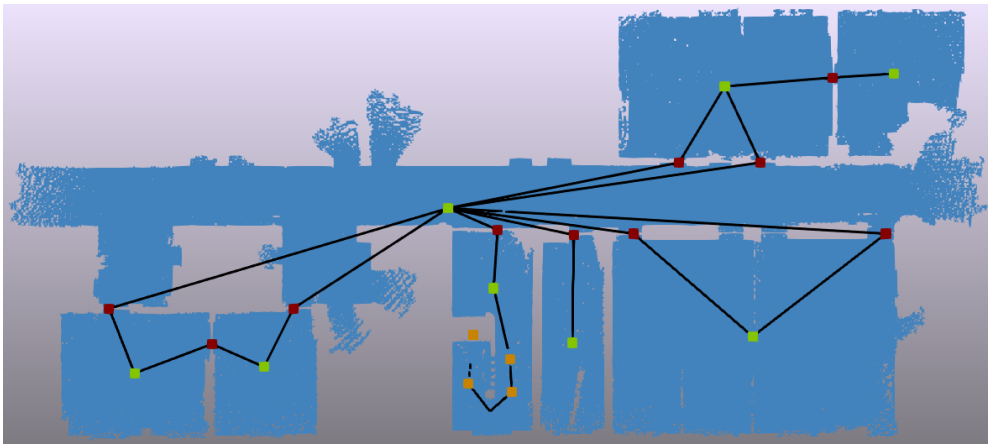
Figure 6.10: The walkable voxel space divided into different rooms

In Figure 6.11 the resulting connectivity network is shown. The orange nodes are generated at the top and bottom of the two stairs which are connected by a flat platform. The spaces are shown with a green node, and doors with a red node. As can be seen, the network is incomplete the first floor because not all doors were detected. Both rooms on the bottom right of Figure 6.11a are made part of the corridor, because doors could not be detected. The connectivity NRG of the second

floor is more complete, although the resulting network is not suitable for routing purposes yet, which is why subspacing is applied.



(a) Connectivity network resulting from the walkable space and known doors of floor 1

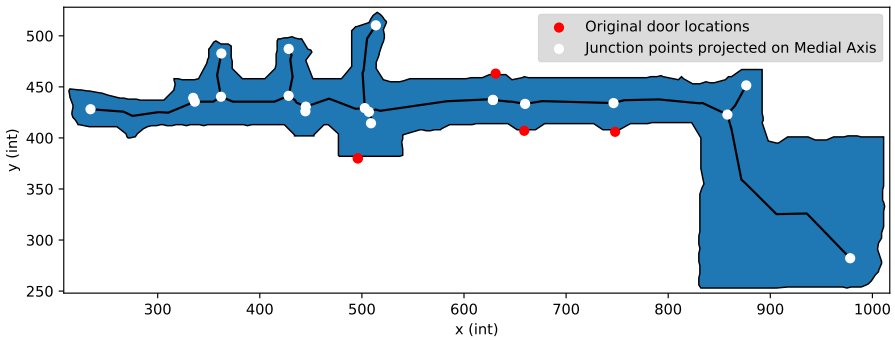


(b) Connectivity network resulting from the walkable space and known doors of floor 2

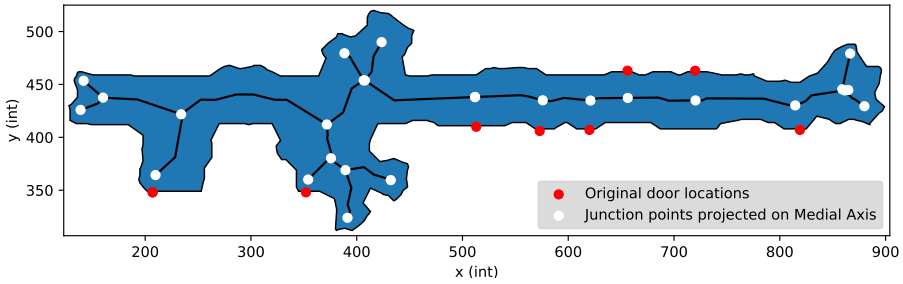
Figure 6.11: Connectivity network of the test point cloud (green = room/corridor, orange = stair, red = doorway)

6.3.2 Subspacing the connectivity NRG

In order to subspace the corridors on both the first and second floor an α-shape is created for the voxels belonging to these regions. The MAT of the α-shape is extracted and, based on the MAT and the location of doors, subspace nodes are defined. The result generating the α-shape and MAT for both corridors is shown in Figure 6.12. It can be noted that at some locations in the subspaced graph nodes are too close to one another, which does not provide added value for navigation. In a next step these nodes are merged with one another, to provide a cleaner navigation graph. The values found for α and the Douglas-Peucker filtering in Section 5.2.6 do not create the best looking corridor polygons for the test point cloud. However the resulting graph from the 2D MAT does provide useful additional nodes in sharp turns and close to doors, which are both beneficial additions for the connectivity NRG. The MAT even produces additional nodes at convenient locations, such as in the top left of Figure 6.12a, where nodes are created in rooms that were walked by, but not visited, because they are protruding from the corridor α-shape. Nonetheless these nodes will not be seen as separate rooms, but regarded as subspaces of the corridor.



(a) Subspacing based on the α-shape and MAT in the corridor in floor 1



(b) Subspacing based on the α-shape and MAT in the corridor in floor 2

Figure 6.12: Result of subspacing the corridors of both floors in the test point cloud

A 3D overview of the subspaced connectivity NRG created by the methods is shown in Figure 6.13. It can be seen that the stairs connect the two floors in the network, and the distance between two rooms in this building could be estimated reasonably accurate. In Figure 6.14 the graph is shown for both floors separately. The extra nodes created in the rooms that were passed by can be seen more clearly here.

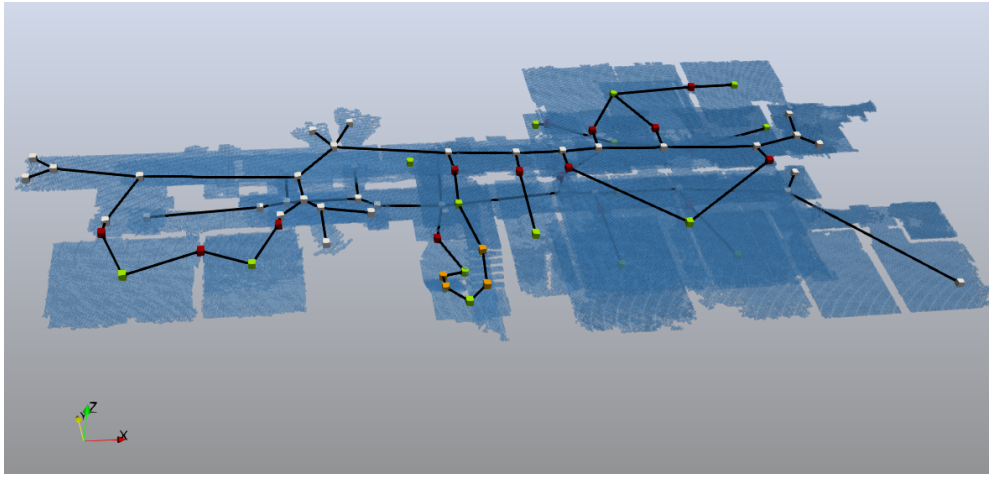
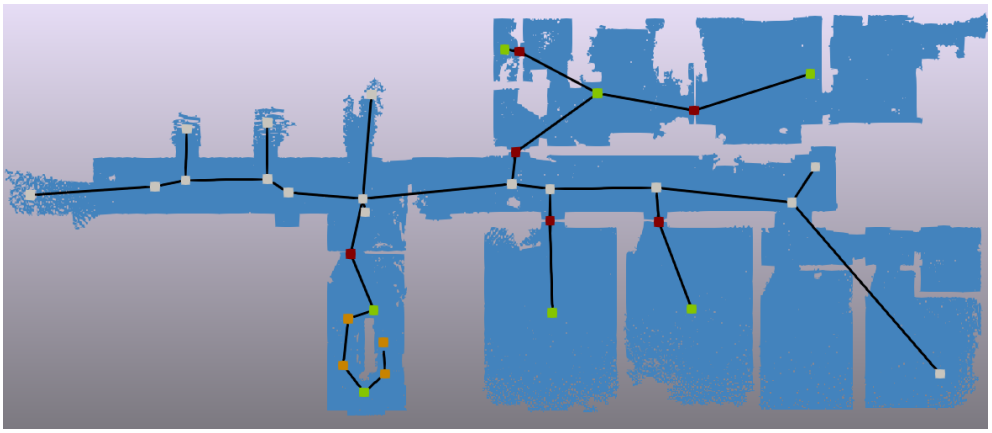
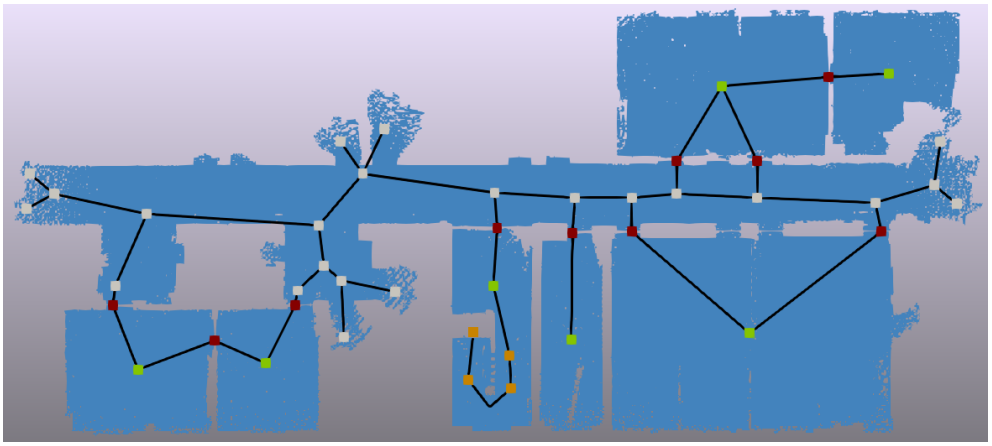


Figure 6.13: A 3D view of the subspaced connectivity graph generated for the test point cloud (green = room/corridor, orange = stair, red = doorway, white = subspaced node)



(a) Subspaced connectivity network of floor 1



(b) Subspaced connectivity network of floor 2

Figure 6.14: Subspaced connectivity network of test point cloud (green = room/corridor, orange = stair, red = doorway, white = subspaced node)

6.3.3 Accessibility NRG

For all detected doors in the test point cloud the width and height are calculated and saved in the database. For most doors (13 out of 16) the height is determined to be around 2.0 m, and the width varies between 0.9 and 1.2 m, which are normal values for doors. However, there are three doors that are calculated to be 1.8 m high, which is not a standard size for a door. An example of one of these doors is shown in Figure 6.15. It can be seen that the voxels in the top and bottom of the door are correctly detected, and there is no noise apparent in the door frame. This leads to two possible explanations: 1) the doors in question are really 1.8 m high, or 2) the low height of the doors is caused by measurement error. The second statement is plausible, because the ZEB-REVO positioning accuracy can decrease to 30 cm after 10 minutes of scanning (see Section 5.1.2). The Simultaneous Localization and Mapping (SLAM) algorithm could have processed the point cloud with this low positioning accuracy, leading to the door height decreasing with 20 cm.

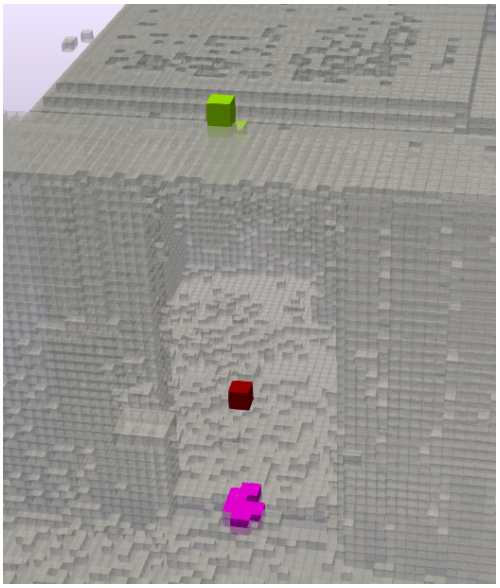


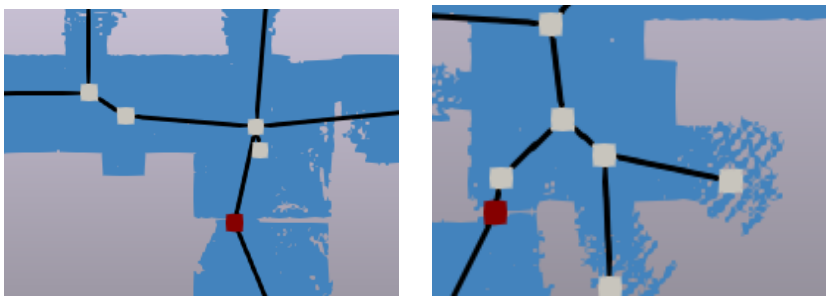
Figure 6.15: A door with a calculated height of 1.8 m (red = door centre node, pink = voxels on floor of door, green = voxels in top of door)

6.3.4 Evaluating the network

The final subspaced connectivity network is compared with the hand-drawn networks found in Appendix A. One thing that should be noted is that the second floor of the test point cloud contains two rooms which seem to be subdivided by vertical lines in the floor plan generated from the point cloud (bottom right corner, and second space from top right corner). These rooms can probably be subdivided into smaller rooms with wall panels, which was not the case during the gathering of the point cloud. The algorithm did thus not create separate spaces for these rooms, but two of the test persons thought they were separate.

The networks drawn by all three test persons show some sort of subdivision of the hallway into different nodes. The graphs of test persons 1 and 2 show a subdivision similar to the one generated by the 2D MAT, as in the methods of this thesis. Test person 3 also based the network on a centre line in the corridor, but did not draw a node at the closest location to every door. Instead of this they created one node to connect multiple doors. This could however lead to routes being drawn inaccurately.

When comparing the drawn networks to the ones created in this thesis, it can be seen that the automatic method sometimes produces too many nodes, often close together (see Figure 6.16).



(a) Detail of subspaced connectivity graph on floor 1

(b) Detail of subspaced connectivity graph on floor 2

Figure 6.16: Locations in the subspaced navigation graphs showing unnecessary nodes close together

Another point of improvement for the automatically generated graph would be to not only subspace corridors, but also specific rooms that connect more spaces. Adding more nodes, or more connections between existing nodes, is beneficial to the network. A good example of this is shown by test person 2, in the graph on the first floor (see Figure 6.17). If a person can walk a straight line between two doors in a room, these nodes should be connected as well. This idea can also be applied in a corridor, and could be executed by analysing if a straight line can be drawn between the two doors, without it crossing the border of the polygon representing the space.

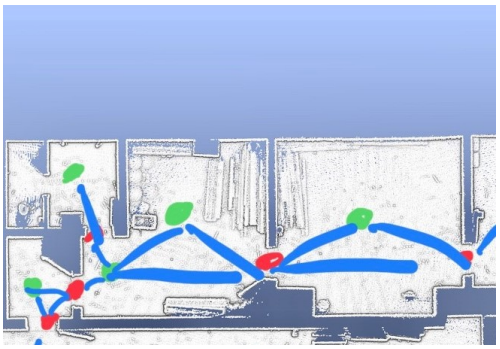


Figure 6.17: Test person 2 subdividing rooms and adding connections between door nodes

7 CONCLUSIONS, REFLECTION AND RECOMMENDATIONS

7.1 CONCLUSIONS

To draw conclusions for this research, first the sub-questions defined in the introduction are answered separately. After this the main research question is answered.

1. What information can be extracted from a point cloud in order to create an IndoorGML ***connectivity*** *and* ***accessibility*** *Node-Relation Graph (NRG)?*

Various research exists on the detection and reconstruction of geometric elements in indoor point clouds, such as walls, floors and furniture. With these methods a point cloud can be converted to a Building Information Modeling (BIM) or CityGML, which can then be used for analysis of the building. Fewer research has been done directly extracting navigation graphs from indoor point clouds. In order to create a navigation graph, it is necessary to know where a person can be in a building. For this the navigable, or walkable, space has to be determined, and doorways have to be detected to split this walkable space into different rooms.

In this research it has been proven that besides information on geometric elements, an indoor point cloud can also provide connectivity and accessibility information. This is done by voxelising the point cloud, and detecting which voxels are walkable, so are part of the floor, stairs or a slope. In combination with door detection in the voxelised point cloud, connectivity relationships between different indoor spaces are found.

For extending the connectivity NRG to an accessibility NRG geometric information can be gathered from the voxelised point cloud. In IndoorGML the accessibility NRG can also contain information on opening hours and restriction rights of certain spaces. This would be impossible to obtain from a point cloud, and would require additional sources to be combined with the network. Some geometric information, such as the occurrence of stairs and slopes, is already present in the connectivity NRG from the walkable voxelspace. A method is proposed to calculate the width and height of all detected doors. However, when applying this method on the test point cloud, some doors were calculated to be 1.8 m high, which is lower than regular doors. It is expected that these door heights are falsely occurring, due to low accuracy of the test point cloud.

2. In what way can ***connecting elements*** *(e.g. doorways and entries) between two indoor spaces be detected?*

In related research the detection of doors, doorways and/or openings in indoor point clouds is often based on ray-casting, edge detection or making use of the trajectory of hand-held laser scanners. The method proposed in this thesis is utilising the idea of the latter, in combination with a new concept of obtaining 3D Medial Axis Transform (MAT) of the indoor point cloud and filtering it to retain medial sheets that are formed inside walls.

When retrieving the 3D MAT of an indoor point cloud, some of the medial sheets will be generated inside walls that were scanned on both sides. These sheets can be distinguished from others by looking at the geometric characteristics of medial balls belonging to points in the sheets. Characteristics of medial balls in wall sheets include a low radius, separation angle close to 180° and low value for z-direction of the normal vector. For this thesis tests are executed that find the optimal values

for these parameters, which results in the selection of wall sheets from the 3D MAT, which are used in the door detection method.

One method of door detection that makes use of the wall sheets relies on modelling a surface from the points, and intersecting this surface in 3D with the trajectory of the laser scanner. However, this method proved to be vulnerable to flaws while trying it on a development dataset, and was therefore discontinued.

The other proposed method is based on voxelisation of the trajectory and MAT wall sheets, whereupon an analysis is done on which trajectory voxels are located inside doorways. In the first part of this method a trajectory voxel is identified to be inside a door when there is a wall sheet voxel above it. Most doors in an indoor environment have some space between the top of the door frame and the ceiling, which results in wall sheet voxels being present above the door frame. However, some doors are of the same height as the ceiling. For this case another method is developed, which looks around the trajectory to find voxels in MAT sheet on both sides of the doorway. This sideways-looking method is based on the idea that when walking through a door, a wall will be both on your left and right side.

The methods were developed on a point cloud, in which 100% of doors walked through could be found. However, in the testing phase with another point cloud, it appeared that these combined methods do not work in all cases. Especially when walls around the door in a point cloud are occluded by large objects, such as opened doors, MAT sheets are not formed inside the wall. In order for this method to work accordingly, the person scanning the indoor environment should walk through all doors that need to be in the network, and also scan the wall behind opened doors.

*3. In what way can separate **indoor spaces** (e.g. rooms and corridors) be detected, and how can these, in combination with the **connecting elements**, be used in modelling the connectivity NRG?*

For the purpose of defining separate indoor spaces in the voxelised point cloud two different elements of the indoor environment are crucial to be detected: all indoor walkable space, and all connecting elements (doorways). Not only the locations of doorways are determined, but also the voxels on the floor inside door frames are identified, for the reason that the walkable voxel finding algorithm grows voxels into regions based on neighbourhood. When detecting voxels on the floor inside a door frame, they create a barrier for the region growing, ensuring that all rooms in the indoor space grow into separate regions.

An α-shape polygon is modelled around voxels belonging to a room or corridor. This method relies on the parameter α, for which the ideal value could not be determined in this research, as it differs per polygon. Inside the boundaries of the alpha-shape a point is chosen as a node for the connectivity NRG. Door nodes are based on the location of doors in the trajectory. Connectivity relationships between doors, stairs, slopes and indoor spaces are found by observing which two spaces are connected by a door, stair or slope.

*4. **In what circumstances** should indoor spaces in the navigation network be subdivided into **subspaces**, and **how** can this be done in a way that is efficient in terms of size of the network?*

IN WHAT CIRCUMSTANCES In related literature the consensus is that indoor spaces, connecting multiple other spaces, often require subspacing in the network. Examples of these are corridors and stairways, which are called *TransitionSpaces*, as defined in IndoorGML. In a connectivity NRG they can be detected by analysing the number of nodes that a node is connected to. These nodes could require subspacing in order to improve the usability of the network. If an indoor space is only connected to one other space, but has a large area and a concave geometry, subspacing could also be preferable.

HOW The method used in this researched is based on the MAT, or skeleton, of a polygon. This polygon is the α-shape modelled for a group of voxels belonging

to an indoor space. The 2D MAT is defined as the middle of a polygon and can thus represent a route that a person would walk through the space, which is a natural basis of subspacing. In case of a corridor connecting multiple rooms, the closest point on the medial axis is found for all connected doors. These locations are used as new nodes, called subspaced nodes. Nodes at junction points, such as in the corners of corridors, or at a T-junction, are also added as nodes. Connectivity between the subspaced nodes is defined by finding out which nodes are directly connected with each other via the medial axis.

*5. How **suitable** is IndoorGML as a **data model** for mapping the navigation network of indoor point clouds?*

IndoorGML is a standard especially developed as a data model for indoor navigation. Having a standard ultimately dedicated to indoor navigation is a great step towards the integration of indoor navigation networks. The navigation module, of which the connectivity and accessibility NRG are a part, offers a data model with a variety of categories for nodes and connections between nodes. Besides navigation graphs for walkable space, IndoorGML also offers modules for other types of space, such as Radio-frequency identification (RFID) and Wireless Local Area Network (WLAN) space, that can be used as an aid for localisation.

However, although the standard provides many possibilities, it also provides some restrictions. The connectivity NRG is seen as a basis for the navigation graph, but this alone is often not suitable for navigation. If the indoor environment includes a corridor that is connecting multiple spaces, the connectivity graph can provide a misrepresentation of routes a human would take in reality. Therefore, subspacing is necessary to make the graph more realistic, but no guidelines for this are given in IndoorGML. The same applies for the inclusion of stairs in the network, for which no guidelines are provided on subspacing either. Based on the results of this thesis it can be stated that the 2D MAT provides a good basis for subspacing a corridor, but for wider open spaces, such as big halls, it does not generate the best results, because the centre line of a large space is often not the only route that can be travelled. For the suspacing of stairs or slopes it is recommendable to add a node at the top and bottom of the stairs.

A deficiency in IndoorGML is the lack of use cases where the standard has been implemented in navigation applications. This kind of research would be beneficial for the standard, in order to improve its usability and applicability. In conclusion, although IndoorGML is an extensive standard for indoor navigation, there are still essential developments to be done in some parts of the data model.

How can a navigation network in IndoorGML format automatically be extracted from a cluttered indoor point cloud and its trajectory?

In conclusion, a point cloud provides a good source for extracting geometric information for an IndoorGML connectivity and accessibility NRG. It can be used to detect separate indoor spaces, doors, stairs and slopes, and moreover the connectivity between these elements. In this thesis a new door detection method is proposed using wall sheets created by the 3D MAT of the point cloud. In some situations this method needs to be extended, but the first results are promising. In order to make the connectivity NRG more usable for navigation purposes, some spaces should be subspaced. Besides this geometric accessibility information can be extracted by defining the shape (doors) or type (stairs, slopes) of elements. Finally it can be concluded that IndoorGML is still in need of more development, but it provides a good basis for indoor navigation information.

7.2 DISCUSSION

Although the application of the proposed methods on the development point cloud gave promising results, the test point cloud provided insight into some limitations of using MAT wall sheets in door detection. The creation of wall sheets is sensitive to objects preventing walls from being scanned. Especially when doors are left open, thus blocking the wall behind, wall sheets will not be formed close enough to doorways to be useful for door detection.

Moreover it should be noted that many of the parameters found in Chapter 5 are sensitive to changes in point clouds. Especially the values found for filtering parameters of medial sheets to retain wall sheets can differ depending on the point cloud. One parameter states that walls have a high number of points, but this number will change with varying point cloud density.

Another restriction of the method is that every door in the indoor environment needs to be walked through in order to be taken into the network. However, as also discussed in this thesis, it can also be a good way to make sure that no false doors, such as permanently closed doors and glass panes, are detected.

The automatic detection of elevators in a point cloud is also a point of discussion. Especially when gathering data with a hand-held Mobile Laser Scanner (MLS) like the ZEB-REVO, an elevator will cause difficulty in the Simultaneous Localization and Mapping (SLAM) algorithm that makes one point cloud out of the scanned points. Detecting elevators from the outside and climbing stairs to different floors would be the best option to include them in the network.

7.3 CONTRIBUTION TO RESEARCH

This thesis provides some new insights to research on point clouds and indoor navigation graphs. First of all a new way of using the 3D MAT of a point cloud is introduced. By filtering the sheets of points obtained as a result of the MAT, medial sheets inside walls are retained. The results of this method can not only be used for the purposes of door detection in this thesis, but could potentially also assist in the reconstruction of 3D geometric models from indoor point clouds.

Based on the MAT wall sheets a new door detection method is introduced. It is based on existing methods that use the trajectory of the laser scanner and find doors by looking above the trajectory for drops in height, compared to the ceiling. Instead of looking at the complete point cloud, only the wall sheets are taken into account to avoid false doors being detected when walking under low-hanging objects. Seeing that not all doors have a wall above them, the method is extended by looking around the trajectory and detecting locations where a wall sheet surrounds the trajectory on both the left and right side. This method could also be applied for door detection in a voxelised point cloud without making use of the MAT wall sheets, although furniture could cause false doors being detected in that case.

Finally, this thesis introduces a method to retrieve a navigation network directly from a point cloud, without needing existing building models or floor plans. It is beneficial to use point clouds for obtaining information about the indoor environment, because the interior of buildings has often changed since their corresponding models or floor plans were created. The resulting navigation graph is modelled according to the IndoorGML standard, in which format few models exist at the time of writing. This research could thus also be seen as a use case and contribution in the development of IndoorGML.

7.4 RECOMMENDATIONS FOR FUTURE WORK

In this thesis it has been proven that geometric information for an IndoorGML navigation graph can be obtained from a point cloud. However, the methods still have some shortcomings, and will therefore not work in every point cloud. Besides this more information can be extracted to make the network more complete. For those reasons a summary of recommendations for future research is given.

- More research should be done on the filtering parameters to obtain 3D MAT sheets inside walls. Especially the minimum number of points inside a sheet should be examined, because this is dependent on the point density of the original point cloud.
- The door detection methods proposed in this thesis could be extended by analysing the voxelised point cloud. The same sideways-looking method as proposed for the trajectory and 3D wall sheets could be used for the trajectory and the point cloud, although furniture could cause false doors being detected in this case.
- Node generation for stairs and slopes works fine in case when a stair or slope is connecting two spaces at different height levels, but more complex stair structures should also be researched, such as spiral staircases or a stairway with multiple branches.
- A 2D α-shape polygon is modelled around voxels belonging to an indoor space, in this way creating a floor plan of the environment. This could be improved by doing more research on the best value of α, letting it depend on the level of noise in a point cloud, and implementing an algorithm that automatically aligns polygons adjacent to each other to let them have a shared boundary. The height of indoor spaces can be extracted from the point cloud to extrude the spaces to 3D cells.
- The subspacing method is shown to be useful for a corridor, but needs to be improved for large, wide rooms such as conference halls and airport terminals. In these spaces multiple paths can be taken besides the centre line created by the 2D MAT.
- Instead of using only the 2D MAT of polygons for subspacing, the polygons could be analysed to extend the routes from skeletal to 'door-to-door' routes (see Figure 3.5), creating even more realistic routes in the environment. This can also be applied by analysing between which nodes in the network a straight line can be drawn, without the line ever crossing the polygon boundary that it is in.
- The accessibility NRG can be extended with more geometric information, such as the width of corridors. This can be extracted from the 2D MAT at subspaced nodes, because the diameter of the medial circle is equal to the width of the corridor at that location. Also a method for elevator detection in a point cloud should be found, so that routes for people in wheelchairs can be extended to multiple floors.
- The point clouds used in this thesis are gathered in a local Coordinate Reference System (CRS), which is not referenced to a known world point. In order for the indoor navigation graph to be used in a seamless indoor/outdoor navigation application it should be georeferenced to a regional or global CRS, so that it connects with the outdoor world.
- As the processing of point clouds in Python is comparatively slow, the methods developed in this thesis could be sped up by using another programming language, such as C or C++. Moreover parallel computing could be applied to some methods that take up the most time.
- Finally, it would be useful to create a navigation application that uses an IndoorGML network as an input for routing. This would be beneficial to the development of the standard, which is currently missing use cases.

BIBLIOGRAPHY

Adrados, C., Girard, I., Gendner, J.-P., and Janeau, G. (2002). Global positioning system (GPS) location accuracy improvement due to selective availability removal. *Comptes Rendus Biologies*, 325(2):165–170.

Afyouni, I., Ray, C., and Claramunt, C. (2012). Spatial models for context-aware indoor navigation systems: A survey. *Journal of Spatial Information Science*, (4).

Alattas, A., Zlatanova, S., Oosterom, P. V., Chatzinikolaou, E., Lemmen, C., and Li, K.-J. (2017). Supporting indoor navigation using access rights to spaces based on combined use of IndoorGML and LADM models. *ISPRS International Journal of Geo-Information*, 6(12):384.

Ballard, D. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122.

Blum, H. (1967). A transformation for extracting new descriptors of shape. *Models for Perception of Speech and Visual Forms*, pages 362–380.

Boguslawski, P., Gold, C. M., and Ledoux, H. (2011). Modelling and analysing 3d buildings with a primal/dual data structure. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(2):188–197.

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

Bosché, F., Guillemet, A., Turkan, Y., Haas, C. T., and Haas, R. (2014). Tracking the built status of MEP works: Assessing the value of a scan-vs-BIM system. *Journal of Computing in Civil Engineering*, 28(4):05014004.

Broersen, T., Fichtner, F., Heeres, E., De LIefde, I., Rodenberg, O., Meijers, B., Verbree, E., Van der Spek, S., and Ten Napel, D. (2015). Project pointless: Identifying, visualising and pathfinding through empty space in interior point clouds using an octree approach. *Geomatics Synthesis Project 2015/16*.

Broersen, T., Peters, R., and Ledoux, H. (2017). Automatic identification of watercourses in flat and engineered landscapes by computing the skeleton of a LiDAR point cloud. *Computers & Geosciences*, 106:171–180.

Brown, G., Nagel, C., Zlatanova, S., and Kolbe, T. H. (2013). Modelling 3d topographic space against indoor navigation requirements. In *Progress and new trends in 3D geoinformation sciences*, pages 1–22. Springer.

Cole, I. R. (2015). *Modelling CPV*. PhD thesis, Loughborough University.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press.

Deak, G., Curran, K., and Condell, J. (2012). A survey of active and passive indoor localisation systems. *Computer Communications*, 35(16):1939–1954.

Delaunay, B. (1934). Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2.

Diakité, A. A. and Zlatanova, S. (2017). Spatial subdivision of complex indoor environments for 3d indoor navigation. *International Journal of Geographical Information Science*, 32(2):213–235.

Diakité, A. A., Zlatanova, S., and Li, K.-J. (2017). About the Subdivision of Indoor Spaces in IndoorGML. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W5:41–48.

Díaz-Vilariño, L., Boguslawski, P., Khoshelham, K., Lorenzo, H., and Mahdjoubi, L. (2016). Indoor Navigation From Point Clouds: 3D Modelling And Obstacle Detection. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B4:275–281.

Díaz-Vilariño, L., Khoshelham, K., Martínez-Sánchez, J., and Arias, P. (2015). 3d modeling of building indoor spaces and closed doors from imagery and point clouds. *Sensors*, 15(2):3491–3512.

Díaz-Vilariño, L., Martínez-Sánchez, J., Lagüela, S., Armesto, J., and Khoshelham, K. (2014). Door recognition in cluttered building interiors using imagery and lidar data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5:203–209.

Díaz-Vilariño, L., Verbree, E., Zlatanova, S., and Diakité, A. (2017). Indoor Modelling From SLAM-Based Laser Scanner: Door Detection To Envelope Reconstruction. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W7:345–352.

Dold, C. and Brenner, C. (2006). Registration of terrestrial laser scanning data using planar patches and image data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5):78–83.

Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110.

Edelsbrunner, H., Kirkpatrick, D., and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559.

Eppstein, D. and Erickson, J. (1999). Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete & Computational Geometry*, 22(4):569–592.

Fichtner, F. W., Diakité, A. A., Zlatanova, S., and Voûte, R. (2018). Semantic enrichment of octree structured point clouds for multi-story 3d pathfinding. *Transactions in GIS*, 22(1):233–248.

Gardiner, J. D., Behnsen, J., and Brassey, C. A. (2018). Alpha shapes: Determining 3d shape complexity across morphologically diverse structures.

Ge, S. and Cui, Y. (2002). Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, 13(3):207–222.

GeoSLAM (2017). ZEB-REVO solution. https://geoslam.com/wp-content/uploads/2018/04/GeoSLAM-ZEB-REVO-Solution_v9.pdf. Accessed: 2019-01-23.

Haunert, J.-H. and Sester, M. (2007). Area collapse and road centerlines based on straight skeletons. *GeoInformatica*, 12(2):169–191.

Hichri, N., Stefani, C., De Luca, L., Veron, P., and Hamon, G. (2013). From point cloud to bim: a survey of existing approaches. In *XXIV International CIPA Symposium*, page na. Proceedings of the XXIV International CIPA Symposium.

Hong, S., Jung, J., Kim, S., Cho, H., Lee, J., and Heo, J. (2015). Semi-automated approach to indoor mapping for 3d as-built building information modeling. *Computers, Environment and Urban Systems*, 51:34–46.

Jamali, A., Rahman, A. A., Boguslawski, P., Kumar, P., and Gold, C. M. (2015). An automated 3d modeling of topological indoor navigation network. *GeoJournal*, 82(1):157–170.

Jung, H. and Lee, J. (2015). Indoor Subspacing to Implement IndoorGML for Indoor Navigation. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-2/W4:25–27.

Kang, H.-K. and Li, K.-J. (2017). A standard indoor spatial data model—OGC IndoorGML and implementation approaches. *ISPRS International Journal of Geo-Information*, 6(12):116.

Khoshelham, K., Vilariño, L. D., Peter, M., Kang, Z., and Acharya, D. (2017). The ISPRS Benchmark on Indoor Modelling. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W7:367–372.

Kim, J.-S., Yoo, S.-J., and Li, K.-J. (2014). Integrating IndoorGML and CityGML for indoor space. In *Web and Wireless Geographical Information Systems*, pages 184–196. Springer Berlin Heidelberg.

Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press.

Krūminaitė, M. and Zlatanova, S. (2014). Indoor space subdivision for indoor navigation. In *Proceedings of the Sixth ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness - ISA 14*. ACM Press.

Kwan, M.-P. and Lee, J. (2005). Emergency response after 9/11: the potential of real-time 3d GIS for quick emergency response in micro-spatial environments. *Computers, Environment and Urban Systems*, 29(2):93–113.

Lamarche, F. and Donikian, S. (2004). Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum*, 23(3):509–518.

Lee, J. (2004). A spatial access-oriented implementation of a 3-d GIS topological data model for urban entities. *GeoInformatica*, 8(3):237–264.

Liu, L. and Zlatanova, S. (2011). A "Door-to-Door" Path-finding Approach for Indoor Navigation. *Proceedings Gi4DM 2011: GeoInformation for Disaster Management, Antalya, Turkey, 3-8 May 2011*.

Lobos, C., Payan, Y., and Hitschfeld, N. (2010). Techniques for the generation of 3d finite element meshes of human organs. In *Informatics in Oral Medicine*, pages 126–158. IGI Global.

Ma, J., Bae, S. W., and Choi, S. (2012). 3d medial axis point approximation using nearest neighbors and the normal field. *The Visual Computer*, 28(1):7–19.

Mahdaviparsa, A. and McCuen, T. (2018). Comparison between current methods of indoor network analysis for emergency response through BIM/CAD-GIS integration. In *Advances in Informatics and Computing in Civil and Construction Engineering*, pages 627–635. Springer International Publishing.

Meijers, M., Zlatanova, S., and Pfeifer, N. (2005). 3d geoinformation indoors: structuring for evacuation. In *Proceedings of Next generation 3D city models*, volume 6. Germany Bonn.

Mirza, R., Tehseen, A., and Kumar, A. V. J. (2012). An indoor navigation approach to aid the physically disabled people. In *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*. IEEE.

Munkres, J. R. (1984). *Elements of Algebraic Topology*. Addison-Wesley, Menlo Park, CA.

Nakagawa, M. and Nozaki, R. (2018). Geometrical Network Model Generation Using Point Cloud Data for Indoor Navigation. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4:141–146.

Nikoohemat, S., Peter, M., Elberink, S. O., and Vosselman, G. (2017). Exploiting Indoor Mobile Laser Scanner Trajectories for Semantic Interpretation of Point Clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W4:355–362.

Nourian, P., Gonçalves, R., Zlatanova, S., Ohori, K. A., and Vo, A. V. (2016). Voxelization algorithms for geospatial applications. *MethodsX*, 3:69–86.

OGC (2014). OGC Indoor GML. (14-005r5).

Peters, R. (2018). *Geographical point cloud modelling with the 3D medial axis transform*. PhD thesis, TU Delft.

Pu, S. and Vosselman, G. (2009). Knowledge based reconstruction of building models from terrestrial laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(6):575–584.

Quirk, P., Johnson, T., Skarbez, R., Towles, H., Gyarfas, F., and Fuchs, H. (2006). RANSAC-assisted display model reconstruction for projective display. In *IEEE Virtual Reality Conference (VR 2006)*. IEEE.

Rueppel, U. and Stuebbe, K. M. (2008). BIM-based indoor-emergency-navigation-system for complex buildings. *Tsinghua Science and Technology*, 13(S1):362–367.

Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (PCL). In *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M., and Beetz, M. (2008). Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226.

Sithole, G. (2018). Indoor Space Routing Graphs: Visibility, Encoding, Encryption and Attenuation. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-4:579–585.

Staats, B. R. (2017). Identification of walkable space in a voxel model, derived from a point cloud and its corresponding trajectory. Master's thesis, Delft University of Technology.

Staats, B. R., Diakité, A. A., Voûte, R. L., and Zlatanova, S. (2017). Automatic Generation of Indoor Navigable Space Using a Point Cloud and its Scanner Trajectory. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W4:393–400.

Staats, B. R., Diakité, A. A., Voûte, R. L., and Zlatanova, S. (2018). Detection of doors in a voxel model, derived from a point cloud and its scanner trajectory, to improve the segmentation of the walkable space. *International Journal of Urban Sciences*, pages 1–22.

Tang, P., Huber, D., Akinci, B., Lipman, R., and Lytle, A. (2010). Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. *Automation in Construction*, 19(7):829–843.

Teo, T.-A. and Yu, S.-C. (2017). The Extraction of Indoor Building Information from BIM to OGC IndoorGML. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-4/W2:167–170.

Thomson, C. and Boehm, J. (2015). Automatic geometry generation from point clouds for BIM. *Remote Sensing*, 7(9):11753–11775.

Tran, H., Khoshelham, K., Kealy, A., and Díaz-Vilariño, L. (2017). Extracting Topological Relations between Indoor Spaces from Point Clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W4:401–406.

Volk, R., Stengel, J., and Schultmann, F. (2014). Building information modeling (BIM) for existing buildings — literature review and future needs. *Automation in Construction*, 38:109–127.

Wallgrün, J. O. (2005). Autonomous construction of hierarchical voronoi-based route graph representations. In *Spatial Cognition IV. Reasoning, Action, Interaction*, pages 413–433. Springer Berlin Heidelberg.

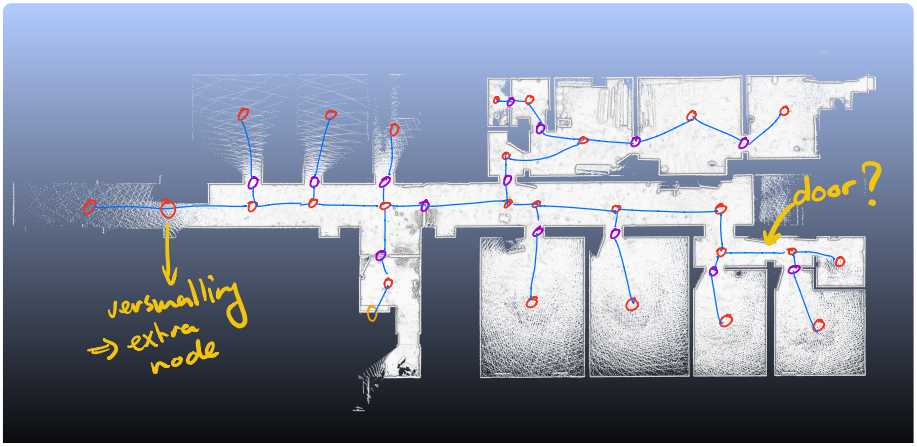
Wang, W., Ai, T., and Gong, C. (2018). Indoor Route Planning under Hexagon Network Considering Multi-constrains. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-4:693–696.

Yang, L. and Worboys, M. (2015). Generation of navigation graphs for indoor space. *International Journal of Geographical Information Science*, 29(10):1737–1756.

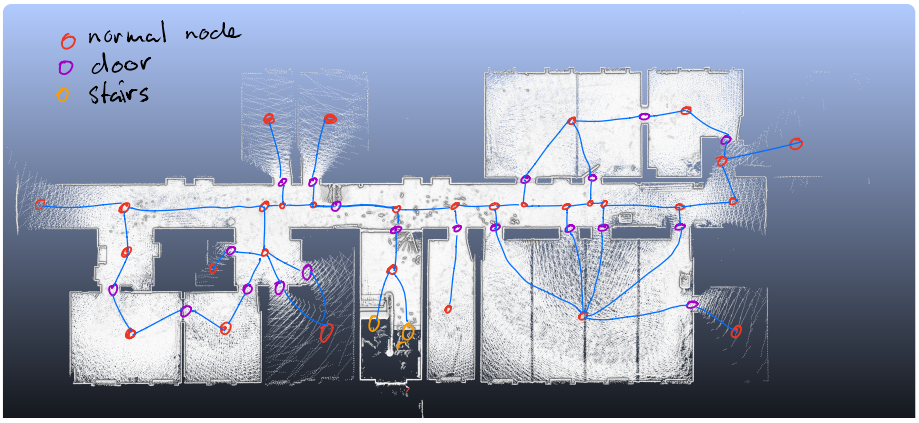
Zhu, Q., Li, Y., Xiong, Q., Zlatanova, S., Ding, Y., Zhang, Y., and Zhou, Y. (2016). Indoor multi-dimensional location GML and its application for ubiquitous indoor location services. *ISPRS International Journal of Geo-Information*, 5(12):220.

A SUBSPACED NETWORK TEST

A.1 TEST PERSON 1



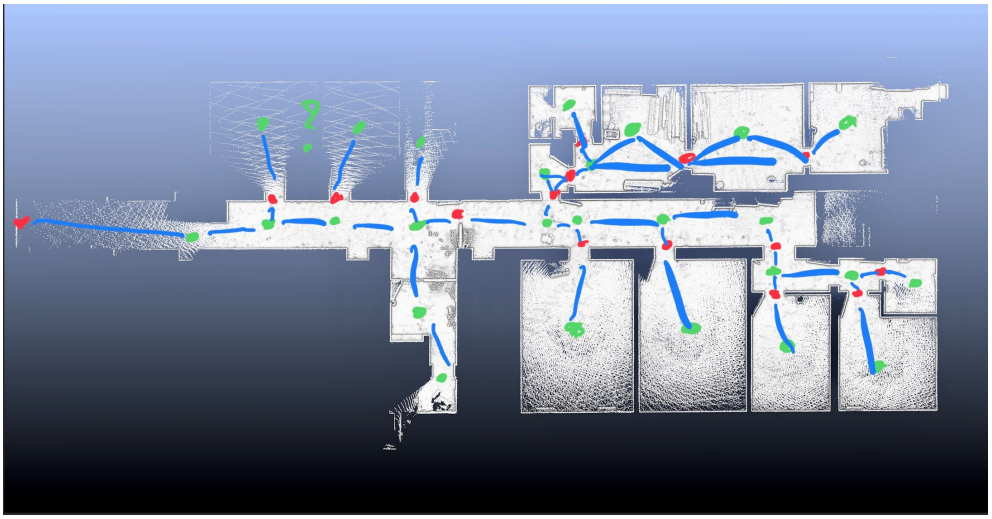
(a) The navigation network of floor 1 of the test point cloud as drawn by test person 1



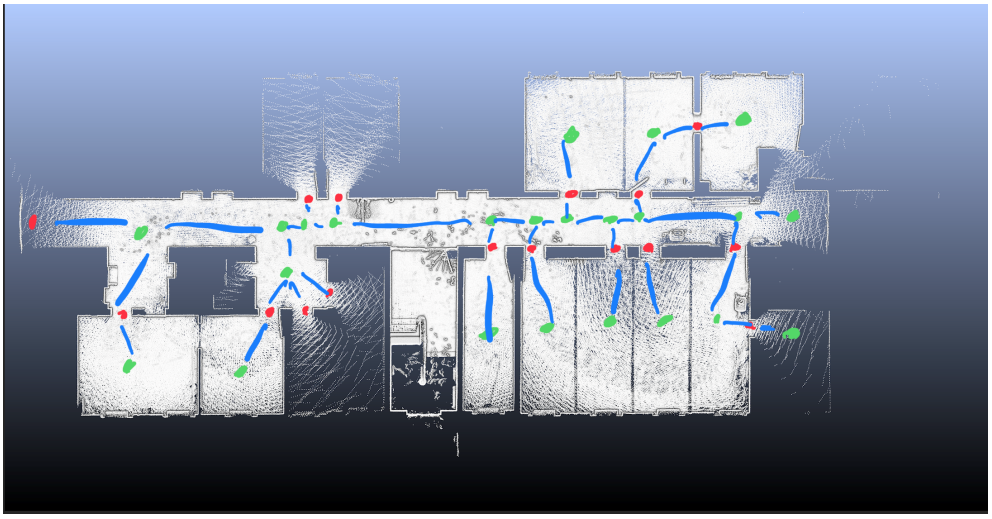
(b) The navigation network of floor 2 of the test point cloud as drawn by test person 1

Figure A.1: The navigation network in the test point cloud as drawn by test person 1

A.2 TEST PERSON 2



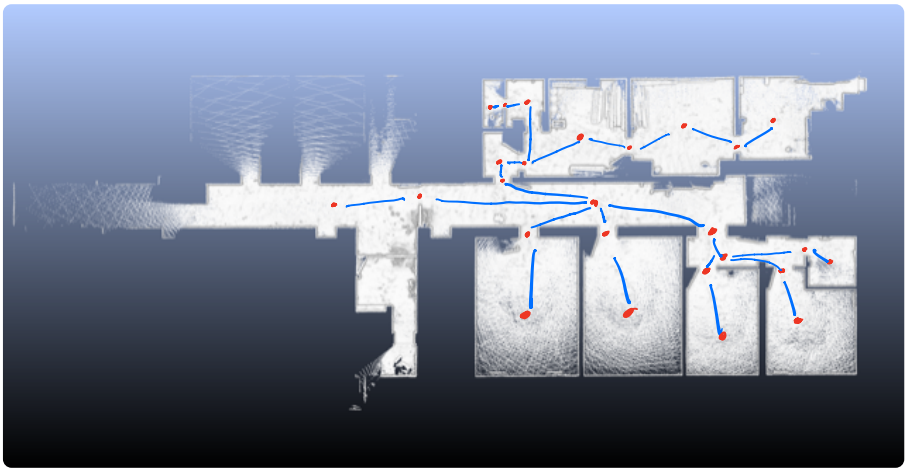
(a) The navigation network of floor 1 of the test point cloud as drawn by test person 2



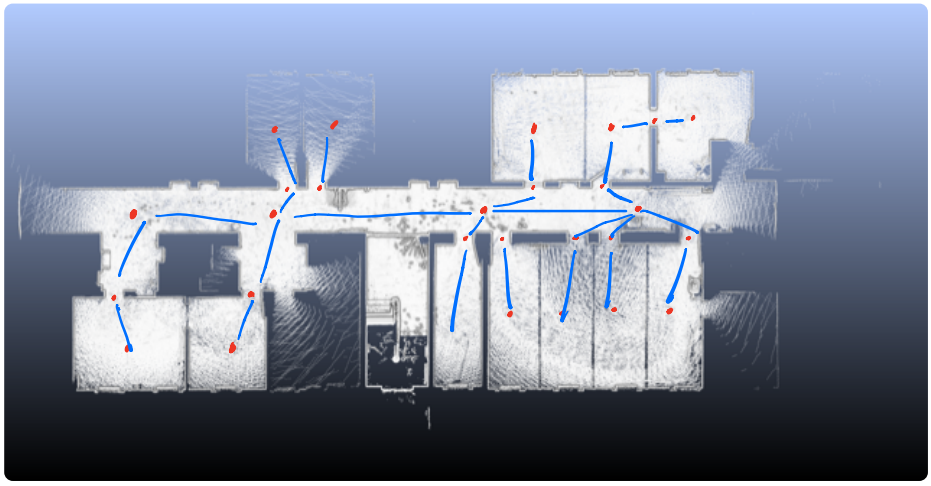
(b) The navigation network of floor 2 of the test point cloud as drawn by test person 2

Figure A.2: The navigation network in the test point cloud as drawn by test person 2

A.3 TEST PERSON 3



(a) The navigation network of floor 1 of the test point cloud as drawn by test person 3



(b) The navigation network of floor 2 of the test point cloud as drawn by test person 3

Figure A.3: The navigation network in the test point cloud as drawn by test person 3

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the `arsclassica` package by Lorenzo Pantieri, which is an adaption of the original `classicthesis` package from André Miede. The figures and diagrams were mostly drawn using IPE. The results including voxels are created with Paraview, and with point clouds in CloudCompare. For some objects on the front page: Vector Design by Vecteezy.com