GIMA
Geographical Information Management and Applications

Building an IndoorGML model in (near) real-time

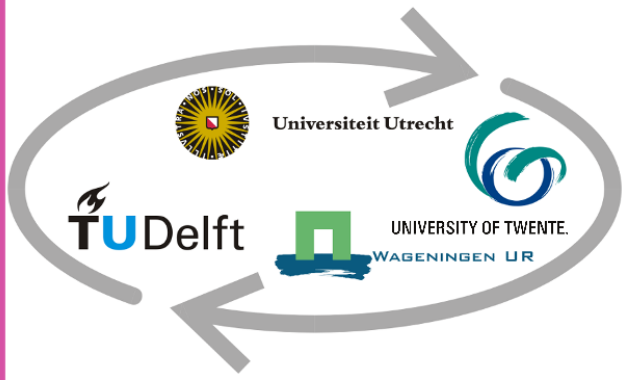
Master's thesis – September 2023

Author:
Danny van Steijn

Supervisors:
ir. R. Voûte
Msc. B. Smit

Responsible professor:
Prof. dr. ir. P.J.M. van Oosterom

University:
Delft University



CGI

Preface

Completing this Master's thesis has been an interesting journey that has allowed me to learn many things in my field. I would like to express my gratitude to the individuals and organizations who have played a role in shaping this work.

I would like to thank my university supervisor, Robert Voûte. His sharp insights have brought this work further.

I would also like to express my gratitude to Bart-Peter Smit, my company supervisor, and CGI, the organization that provided me with the opportunity to undertake this research. His constructive feedback helped me to push the work farther.

Furthermore, I would like to express my appreciation to Lucía Díaz-Vilariño for her assistance in obtaining the dataset utilized in this thesis, and to Abdullah Alattas and Sisi Zlatanova for helping me to understand the standard of IndoorGML better.

Lastly, I would like to acknowledge my fellow interns, whose camaraderie have made this journey a fun one.

Definition of terms

This thesis makes use of several terms that are key to understand the thesis. These terms are defined in the text. Below, these terms are explained in a structured manner for the reader to consult while reading this thesis. It is important to keep in mind that the definitions below are within the context of this research.

(near) real-time	A computing operation (i.e., the generation of a IndoorGML model) that is executed within a set time constraint (i.e., at the same time as collecting the indoor data). It is **near** real-time, when the timing operation does not go completely according to the time constraint, or as in this case the process occurs in steps.
Accuracy	Geometric quality criteria that tests the closeness of the elements in the source models to those of the reference models.
Completeness	Geometric quality criterion that tests to what extent the elements of the reference models are found in the source models.
Correcteness	Geometric quality criteria that tests to what extent the elements of the source models are found in the reference models.
Dual space	Part of the IndoorGML standard. Representation of the indoor space in which the spaces are represented as a graph of nodes (0D) and edges (1D). Dual space can be reconstructed from the primal space.
Dynamic model	See *dynamically generated model*
Dynamically generated model	An indoor model that is generated in (near) real-time, or while the scanning process is still ongoing.
Indoor map	Indoor model that allows for navigation.
Indoor model	A representation of the of the indoor space of a building.
Indoor space reconstruction	The process of modelling certain elements of the indoor environment, such as a wall or door (geometry) or the connectivity between the elements (topology).
Modelling	Creating a representation of real life using raw input. In this case creating a model of the indoor space using a point cloud of the interiors of a building.
Primal space	Part of the IndoorGML standard. Representation of the indoor space in which the spaces are represented as solids (3D) or surfaces (2D).
Reference model	Model that is manually generated using a digitization method on a point cloud. It is used as a reference to which the dynamic model is tested.
Situation awareness	The perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future.

Abstract

Navigating indoor spaces requires indoor models. Depending on the use, such indoor models may need to be created quickly, such as in an emergency. Creating these models in real-time could help emergency responders to better interpret the situation by applying location-based services on the models that are created on-the-fly, such as navigation and spatial measurements. These services also requires a sufficient quality of the models. This work presents a methodology for automatically creating an IndoorGML model from a benchmark dataset in near real-time using a simulation of the scanning process. The timestamps of the dataset was used to simulate the scanning of the building. The rooms were identified and reconstructed by detecting the doorways and extracting the points scanned prior to detection. The results demonstrate the models' completeness but highlight challenges such as limitation in accuracy of the geometry and a delayed reconstruction of the doorways and navigation graph. This work contributes to science in a unique way by exploring how to create an indoor spatial model in real-time, and highlighting the challenges involved. At the same time, it also considers how to evaluate such a model.

Contents

Preface 1
Definition of terms 2
Abstract 3
1 Introduction 7
1.1 Background – mapping the indoors 7
1.2 Problem – creating indoor spatial models in (near) real-time 8
2 Research question 11
2.1 Scope 11
3 Theoretical background 13
3.1 Indoor spatial models 13
3.1.1 Volumetric and Surface-based representation 13
3.2 IndoorGML 13
3.2.1 Cellular space 13
3.2.2 Duality 14
3.2.3 Semantic representation of space 15
3.2.4 Navigation Extension Module 15
3.2.5 Current state of the standard 15
3.3 Indoor spatial reconstruction 15
3.3.1 Methods of spatial indoor reconstruction 16
3.3.2 Real-time indoor reconstruction 16
3.4 Emergency response and situation awareness 17
4 Indoor spatial model framework 19
4.1 Geometric quality criteria 19
4.1.1 Completeness 19
4.1.2 Correctness 20
4.1.3 Accuracy 20
4.2 Service-oriented modelling 20
5 Methodology 23
5.1 General approach 23
5.2 Data and software 24
5.2.1 Data 24
5.2.2 Software 25

5.3 The IndoorGML standard 26
5.3.1 PrimalSpace 27
5.3.2 DualSpace 27
5.4 Reference model 28
5.5 Dynamic model 31
5.5.1 Doorway detection 31
5.5.2 Extracting rooms 32
5.5.3 Geometric reconstruction 33
5.5.4 Topological reconstruction 34
5.5.5 Performance issues 34
5.6 Building an IndoorGML file 35
6 Results: comparing the reference and dynamic IndoorGML models 37
6.1 Reference model 37
6.2 Dynamic model 37
6.2.1 Modelling GeneralSpaces 37
6.2.2 Modelling TransferSpaces 39
6.2.3 The resulting IndoorGML model 40
7 Conclusion 45
7.1 First sub-question 45
7.2 Second sub-question 45
7.3 Third sub-question 46
7.4 Main question 46
8 Discussion 49
8.1 Strengths and limitations 49
8.1.1 Strengths 49
8.1.2 Limitations 49
8.2 Future work 49
8.2.1 Evaluation 49
8.2.2 Real life implementation 49
9 References 50
10 Appendix A: IndoorGML Core 53
11 Appendix B: Navigation Extension Module 54
12 Appendix C: Visualizations of the reference model 57

13 Appendix D: Step-wise visualization of the dynamic model 59
14 Appendix E: IndoorGML of dynamic model 60
15 Appendix F: IndoorGML of GeneralSpace 61
16 Appendix G: IndoorGML of Node and Edge 65

1 Introduction

1.1 Background – mapping the indoors

The use of navigation systems, particularly navigation apps on mobile devices, has become increasingly popular in recent years.[1] Mobile devices use built-in GPS and available maps to navigate outdoors. Position determination in indoor spaces has also become possible through the widespread use of smartphones and other mobile devices, which allow position determination in indoor spaces by receiving Wi-Fi signals or other types of signals (Retscher, 2022). However, what is often missing for the navigation in indoor spaces is accurate information on the indoor space, such as a map. An indoor model is crucial for effective navigation, especially in complex buildings (A. A. Khan et al., 2015; Sithole & Zlatanova, 2016). Such models allow for the linking of a user's position to the digital representation of the indoor space, enabling navigation. Indoor models that enable navigation can also be called indoor maps. A commercial example of creating and applying indoor maps is the *Indoor Maps Program*[2]. Here Apple gives clients industry standard tools (Apple & Open Geospatial Consortium, 2021) to build indoor maps of large public spaces like airports and malls, and implement these in map applications for navigation. Digital indoor models can also be a useful tool in cases of emergency situations, existing information could be missing or outdated, or there have been major changes in the space on which the existing model is based. For example, a ceiling could have collapsed, blocking the entrance to part of the building. This makes existing information unreliable for navigation. The challenge then is to map the right information in a timely manner so that emergency responders can more safely navigate interior spaces.

A digital model with the necessary information such as connectivity and positional information of the user can help find the right routes even in situations such as poor visibility (Rueppel & Stuebbe, 2008). Prior work has been done to form new theories, data models, database management systems, and applications for indoor space (Afyouni et al., 2012; H.-K. Kang & Li, 2017; Li, 2008). Important within an indoor spatial theory and relevant for this thesis is the collection of indoor spatial data and its exchange in data models (H.-K. Kang & Li, 2017). In many projects (Becker et al., 2015; Lehtola et al., 2021; Li et al., n.d.; Nikoohemat et al., 2020), the collection of data and its transformation into an indoor spatial model are done in two separate steps. First the data is captured as point clouds using capturing technologies like LiDAR or RGBD cameras using photogrammetry (Lehtola et al., 2021). Secondly, the point clouds can be reconstructed into an indoor spatial model in a manual or (semi)-automatic manner (Li et al., n.d.; Nikoohemat et al., 2020). These are two separate steps in succession. This therefore assumes you have sufficient time to first walk through the indoor space in the first step, and then later in time reconstruct this gathered dataset into an indoor model. Nevertheless, certain emergency scenarios exist wherein there is either a lack of available or incomplete data on the interior, or circumstances arising from the emergency itself have altered the data (e.g., obstructing an entrance). Within an emergency, time is of the essence. It would be useful then to have these steps performed simultaneously: the data of the interior

[1] https://www.insiderintelligence.com/content/people-continue-to-rely-on-maps-and-navigational-apps-emarketer-forecasts-show

[2] https://register.apple.com/indoor

space is acquired by a mobile device, and at the same time a digital model of it is reconstructed. Here the indoor spatial model is created in real-time.

1.2 Problem – creating indoor spatial models in (near) real-time

The developments being made in the area of (automatically) reconstructing indoor spatial models from point clouds are valuable for many navigation or BIM applications (Khoshelham et al., 2021). However, in an emergency the layout of a space can change suddenly with the appearance or disappearance of new objects/obstacles, such as a blocked path due to a collapsed ceiling, making the indoor model outdated and unfit for use. For use cases like first response and emergency operations, it is critical that information of the changed environment is quickly incorporated in an updated indoor spatial model, as they rely on up-to-date information for decision making (Nikoohemat et al., 2020). For these cases it could help if the indoor spatial model is created in real-time. Here the model is made on the fly, meaning that the reconstruction is performed at the same time as scanning the environment (Smit et al., 2021; Zhu et al., 2022). Using mobile devices, such as a Mobile Laser Scanner (MLS) or a head-mounted camera, to collect data, helps in covering larger areas cost-efficiently, and the trajectory of mobile devices can help in the reconstruction of the models (Nikoohemat et al., 2018).

In this research data is used that was collected by a Mobile Laser Scanner. An MLS is a mobile device that uses LIDAR (Light Detection and Ranging) to gather data of the surface of the environment or objects. This data comes with timestamps indicating exactly when a portion of the data was collected. Using these timestamps, a model is dynamically generated by simulating a real-time situation from the timestamps. The term dynamically generated models is used here to refer to indoor spatial models that are generated in (near) real-time, or while the scanning process is still ongoing. This is done by querying part of the point cloud and to automatically reconstruct these points into an element of the indoor spatial model. This step is repeated to update the spatial model with new parts of the building each time a queried part is reconstructed. The updating is done in steps. A step is made once there is enough data to reconstruct an element of the indoor spatial model. So even if there is newer data that is gathered in real-time, this is not yet be incorporated in the indoor spatial model until that condition is met. This is why this method is called **near** real-time.

This thesis focuses on the implementation of the IndoorGML standard as an indoor modelling standard for the creation of an indoor spatial model in near real-time. IndoorGML[3] is an open standard from the Open Geospatial Consortium that is focused on indoor navigation. The choice for the IndoorGML standard was made because it is an open standard exchanging indoor spatial data. Having an open standard is important because it facilitates the sharing of information. It should not happen in an emergency operation that, for example, the police can use the model and the fire department cannot. In addition, the ultimate goal of an indoor spatial model in an emergency operation is that it can be used for safer navigation in indoor space. The structured space model of IndoorGML allows for a flexibility that other standards do not have, as this model gives the possibility to create a topological model for navigation and a geometrical model for other spatial queries like geometrical measurements. There is

[3] OGC® IndoorGML | OGC

also a gap in the literature of the implementation of IndoorGML in the (near) real-time creation of indoor spatial models.

2 Research question

The objective of this research is to explore the fit for purpose of the IndoorGML standard in creating an indoor spatial model in real-time.

Research question – To what extent is IndoorGML usable as an indoor modelling standard in the creation of an indoor spatial model in near real-time.

The main research question is divided into two steps, each with their own sub-questions. The first step involves exploring a framework for evaluating a dynamic indoor model. For the second step a method was designed and applied to create a dynamic indoor model.

Step 1

The first step is a more theoretical one and is covered in chapter 4 called *Indoor spatial model framework*. The objective of this step is to create an evaluation framework for the dynamic IndoorGML created for this thesis. Consideration is given to what an indoor spatial model should look like, and what requirements it should meet, as well as taking into account possible use cases of the model (e.g., emergency response). This step involves looking at different existing frameworks from literature on evaluation of spatial models. A reference model in IndoorGML is created manually from a benchmark point cloud dataset. The first sub-question is defined as following:

1. **How to evaluate a dynamically generated IndoorGML model?**

Step 2

After exploring the IndoorGML standard, an attempt is made to create an IndoorGML model in near real-time by using spatial mapping methods. The goal is to create a method that functions as *Proof of Concept*. The sub-question asked here is:

2. **What are the necessary steps to implement IndoorGML in near real time?**

Then finally the dynamically generated IndoorGML model is compared to the reference model, to try and evaluate the methodology for creating a dynamically generated indoor spatial model. The final sub-question is:

3. **How does the in near real-time created IndoorGML model compare against a reference model?**

2.1 Scope

Outside of this scope is:

1. Going into the details of surface reconstruction from point clouds;

The aim of this thesis is not to conduct a study in reconstructing point clouds to surfaces. It merely uses these methods to achieve a proof of concept for the application of IndoorGML in near real-time modelling of interior spaces. The application of the methods will be described, but the underlying mathematics of these methods will not.

2. Applying spatial analysis and routing on the models created in this research.

Applying a spatial analysis and routing could provide an insight into whether an IndoorGML is suitable for use (e.g., for navigation). However this was left out of scope due to reasons of time. Instead, to validate the usability of the IndoorGML models, the files are compared with examples from the documentation to see whether they are correctly constructed. If this is the case, then it can be assumed that it can be used, as this has been demonstrated in the literature (Alattas et al., 2020). Comparing the dynamically generated models to the reference models and describing this comparison also helps to evaluate the near real-time method of indoor modelling. Nevertheless, the concept of location-based services (e.g., navigation and positioning) is described in section 4.2. The underlying ideas of this concept have been taken into account in establishing the method of dynamically creating indoor spatial models.

3 Theoretical background

3.1 Indoor spatial models

The life cycle of indoor spatial information can be categorized similar to other type of information: 1) data collection, 2) data management, 3) data sharing, and 4) spatial analysis and services (H.-K. Kang & Li, 2017). An indoor spatial model should be supportive for most of this life cycle, meaning that it should facilitate actions such as sharing, updating with new data and allow for a range of different services. (Afyouni et al., 2012) did a review on different spatial indoor models, specifically those that are developed for context-aware navigation services. They make a clear taxonomy of models with a *geometric-based approach*, in which space is seen as continuous, and *symbolic-based approach*, that deals with space in a topological or hierarchical manner. In this paper they test the models for two types of requirements: 1) *service-oriented* requirements, and 2) *efficiency* requirements. The first type of requirements handles various types of applications and services that should be supported by an indoor spatial model, such as positioning and navigation. The efficiency requirements looks at the efficiency in handling such services, as well as the ease of creating the indoor spatial model. Several standards for 2D and 3D indoor spatial models exist, but many of these standards fit only a few of the requirements (H.-K. Kang & Li, 2017). BIM standards such as IFC[4] or CityGML[5] focus on the features within space, but are not fit for navigational purposes, because of the lack of information about adjacency and connectivity.

3.1.1 Volumetric and Surface-based representation

The geometry of an indoor spatial model can have either a volumetric representation or a surface-based one. In a volumetric representation, all the geometric elements consists of volumetric solids. Meaning that all elements (e.g., rooms and walls) are bounded by surfaces, including those that are not visible from the inside (Tran et al., 2019). A surface-based representation on the other hand, models these elements only by the surfaces that are visible. This way of representation is more common when modelling from point clouds, because the points only represent visible surfaces. The indoor models in this study use a surface-based representation.

3.2 IndoorGML

IndoorGML is a modelling standard created for indoor navigation. However, it is very extensible and can be applied in different situations (H.-K. Kang & Li, 2017), including emergency response (Alattas et al., 2020; Nikoohemat et al., 2020). IndoorGML approaches space in three different ways: geometrical, semantical and topological. This is reflected in its concepts, which are described in this section. For a full overview of the model, see the UML models in appendices A and B.

3.2.1 Cellular space

In IndoorGML space is represented by cellular space. This is a set of cells each with its own unique ID that represent different types of spaces, such as rooms, walls or virtual spaces (e.g., wi-fi coverage), depending on the theme of the layer. The cell can have a geometry (solid or

[4] Industry Foundation Classes (IFC) - buildingSMART Technical
[5] CityGML | OGC

surfaces) described in the IndoorGML document itself or externally referenced from another model. This is called the primal space. The primal space can also be omitted completely. The cells can also be represented by a topological model that shows the relationship between a space and its boundaries. An IndoorGML can have several cellular spaces of different themes. Cell spaces within a cellular space are not allowed to overlap geometrically, but gaps may exist.

3.2.2 Duality

IndoorGML also has a method to represent the relationship between the cells, called *dual space*. The geometrical representation in primal space can be converted to a topological representation in dual space using the concept of Poincare duality (figure 3.1) (Munkres, 1984), in which three-dimensional space is converted to a zero-dimensional node, and two-dimensional space (e.g., walls) is converted to edges in one-dimension.

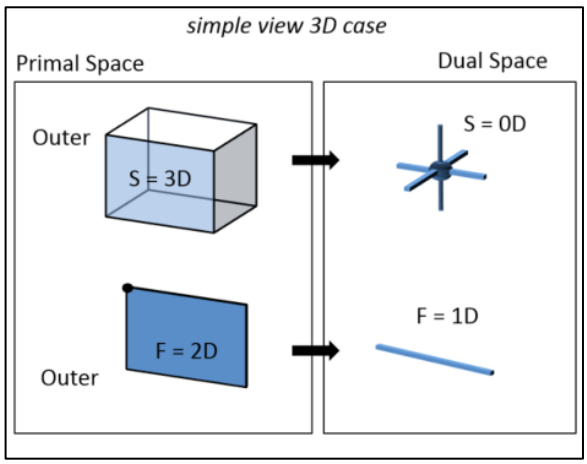


Figure 3.1 - Concept of duality in IndoorGML (applied from the IndoorGML standard documentation)

In this dual representation, space is represented as a graph of nodes (cells) connected by edges (topological relation between cells), called a Node Relation Graph (NRG). This makes IndoorGML a hybrid model, which allows for a geometric-based approach and a symbolic-based approach (Afyouni et al., 2012). By having two simultaneous but different representation of space, IndoorGML is not limited in its applications, such as is the case with standards that only use one type of representation (Afyouni et al., 2012; H.-K. Kang & Li, 2017). The geometrical representation would be more useful for applications that require a more fine granularity, for example in spatially measuring the doorway to investigate whether a room is accessible for a stretcher in an emergency response case. The topological representation is useful for navigation. The overarching model that contains both the geometrical and topological space of the primal and dual space, is called the *structured space model* (figure 3.2).

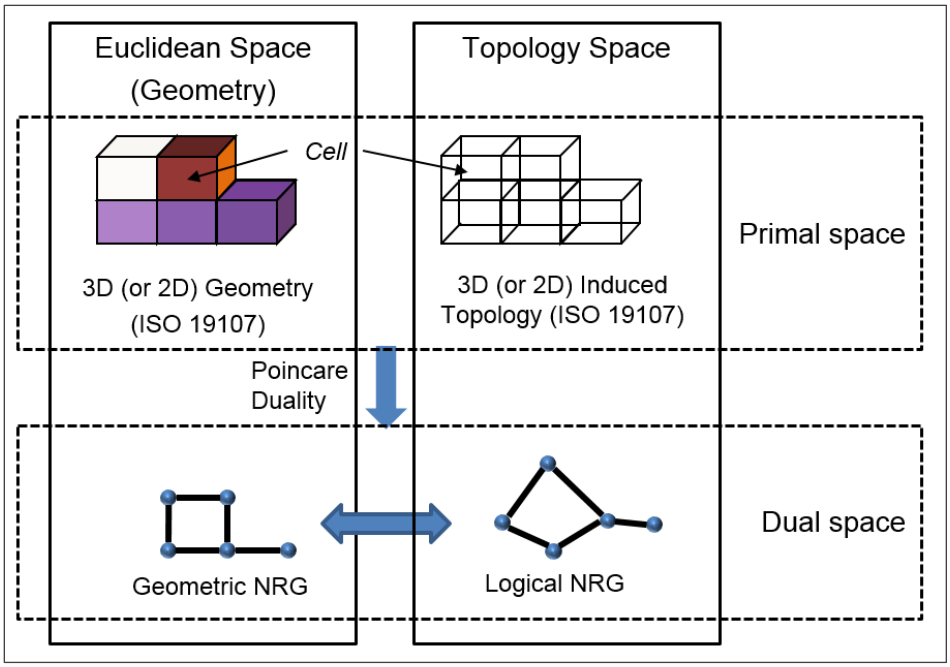


Figure 3.2 – Structured space model, different types of spatial representation in IndoorGML (applied from the IndoorGML standard documentation)

3.2.3 Semantic representation of space

The core module offers basic semantics: name, level and Point of Interest (PoI). Other extension modules may add more advanced semantics. For example, the NavigableSpace class of the navigation module, may denote the type of locomotion that is allowed. This can be flying, rolling, walking or unspecified.

3.2.4 Navigation Extension Module

Several extension modules exist for IndoorGML. One of these is the navigation extension module. This module offers a new set of classes that build upon classes of the core module.

3.2.5 Current state of the standard

IndoorGML however is a fairly new standard which results in it not having much use cases, especially not outside the academic world (A. Diakité et al., 2021). There exist a few projects that aim to further develop IndoorGML in a useful standard for different sectors, and to promote its use for these sectors. In their paper (A. A. Diakité et al., 2020) explore a second version of IndoorGML in which they discuss proposed changes to the standard and in which they argue for the implementation of the Flexible Space Subdivision (FSS) framework (A. A. Diakité & Zlatanova, 2018).

3.3 Indoor spatial reconstruction

(Z. Kang et al., 2020) did an extensive review on techniques for 3D reconstruction of indoor environments. They categorize the techniques into three types of modelling: 1) geometry modelling, 2) semantic modelling, and 3) topological modelling. Geometry modelling is focused on reconstructing the 3D or 2D geometry of the indoor environment by computing the polygonal structural elements such as walls, floors, doorways, etcetera. Objects in a room can cause problems in calculating an accurate geometry for the structural elements. With

room layout estimation this is mitigated by estimating the general layout of a room using its corners or edges. Finally, (Z. Kang et al., 2020) mention the integration of indoor and outdoor models as a topic of indoor modelling.

Semantic modelling aims to model the meaning of a space or the objects within that space, by labelling it. For example, whether part of a space is a floor, wall or ceiling, but also, whether an object within a room is a chair or a table (S. H. Khan et al., 2014). But it may also relate to the functionality of a space (e.g., a classroom, or a kitchen), or what floor the space is on. In IndoorGML the semantics mostly refer to the navigation of the indoor space, for example, whether a space is navigable or non-navigable, or whether the space is a transfer space (e.g., a door) or a general space that may have a purpose other than transferring through.

Finally, topological modelling is concerned with getting the topological relationships between spatial units such as rooms and hallways (Z. Kang et al., 2020). This can result in a subdivision method model, in which the topological relationships are extracted from subdivided spaces. Spaces can be subdivided based on their boundaries, or by making a Voronoi diagram. Topological models based on subdivision methods are highly efficient in querying paths. Another type of model is the grid model, which includes the whole of walkable space in the indoor environment, to offer the user several paths. The query efficiency of the grid model is lower, but it takes into account spatial location information.

3.3.1 Methods of spatial indoor reconstruction

Methods for reconstructing the geometry of structural elements in an indoor environment can be subdivided into *structural element extraction* and *space decomposition-and-reconstruction* methods (Z. Kang et al., 2020). *Structural element extraction* methods often include some type of regression analysis, such as least square-based methods or random sample consensus (RANSAC), to extract planar primitives from the data. It is based on a strong Manhattan assumption, meaning that it works better in environments that are square. Under this assumption, it looks for a set of planar primitives. *Space decomposition-and-reconstruction methods* first partition the indoor spaces or rooms into separate units, and then reconstruct each unit separately. The position or trajectory of the scanning device can be of use to compute the geometry of the different rooms.

Many reconstruction methodology start out by pre-processing the point cloud using a down sampling method (Nikoohemat et al., 2020; Shi et al., 2018). Other pre-processing methods are focused on getting local surface properties of individual point clouds, such as the normal and curvature, by using local nearest neighbors (Shi et al., 2018). Based on these properties, points can be segmented into different groups with similar properties. Then a surface can be fitted to it using some sort of regression analysis on the points belonging to a surface.

3.3.2 Real-time indoor reconstruction

This section focuses only on real-time reconstruction of data collected by mobile devices. (Smit et al., 2021) used a Microsoft HoloLens to map the indoor environment in real-time. Using environment and depth cameras, the HoloLens captures a point cloud of the indoor environment which is then transformed into a spatial mesh which can be transferred in real-time. The spatial mesh can be visualized in several ways, including an object visualization that

interprets the spatial mesh and separates the floors, walls and objects, allowing also for navigation (Smit et al., 2021). Applying reconstruction methods in real-time can present a number of difficulties. (Zhu et al., 2022) state that real-time reconstruction methods often assume slow sensor-motion, scenes without dynamic objects and invariant lightning. In their work they create a fast motion reconstruction method for indoor environments. Fast sensor motions can cause problems such as motion blur, which degrades the color information of the images, and thus leading to inaccuracies because of failing feature-based tracking and loop-closing that make use of the color information between frames (Zhu et al., 2022). Real-time mapping of indoor spaces using LiDAR has been demonstrated in the work of (Zhang & Singh, 2017), in which they apply a low-drift and real-time method using a laser scanner.

3.4 Emergency response and situation awareness

Emergency response can be assisted by informed decision making, meaning that decisions are supported by relevant information, or situation awareness (SA). (Endsley, 1988) defines situation awareness as "*the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future.*". Within this definition, the levels of perception, comprehension and projection can be understood. The perception level requires the availability of descriptive data. The second level of SA, comprehension, is to transform raw data into information that is easy to interpret, making it more comprehensible for the user. The third level is projection, which goes beyond interpretation of the current situation by predicting future scenarios.

Indoor spatial models may help to achieve SA in an emergency situation if the models contain up-to-date information of the indoor environment. Floor plans may exist, but recent changes in the indoor space might make these outdated (Nikoohemat et al., 2020), even for regularly updated floor plans. A new model would need to be made from information collected by the emergency responders themselves (Dilo & Zlatanova, 2011). This data can be used on its own or in combination with existing "static" data (e.g., floorplans) to attain situation awareness. (Smit et al., 2021) uses a SLAM-method to gather 3D indoor spatial data in real-time and create situation awareness by interpreting geometric features of the data. SLAM (simultaneous localization and mapping) solves the problem of placing an agent with a camera or sensor at a position with unknown coordinates and having it gradually map its environment while moving. A system has been developed to offer a solution to this problem, by using the combined estimated position of mapped landmarks and of the agent's trajectory to improve the results (Durrant-Whyte & Bailey, 2006). SLAM methods may encounter problems in terms of accurate geometric data. Positioning may deviate from reality (i.e., drift), especially when such a method is used on a long-term basis. Certain algorithms like loop-closing can solve these errors in accuracy (Karam et al., 2021).

4 Indoor spatial model framework

The ISPRS working group IV/5[6] created a framework for evaluating automatically created indoor spatial models. They provide several point cloud datasets of different buildings, of which they manually make reference models using BIM software. Based on these reference models they discuss an evaluation framework containing a set of criteria for the quality of an indoor spatial model (Khoshelham et al., 2017, 2018). The evaluation is qualitative and quantitative and is performed on models that are automatically made from the benchmark point cloud datasets. In an early paper of the project (Khoshelham et al., 2017), it is reported that the qualitative evaluation focuses on semantics and topological relations of the model and is carried out by a panel of experts. However, they have shifted their focus on only evaluating the geometric quality of wall elements because they did not have access to the details for the qualitative evaluation of semantics and topology (Khoshelham et al., 2021). The quantitative evaluation was made for the geometric representation of the model, in which the automatically reconstructed models (called source models) are compared to the reference models. In the quantitative evaluation they focus on three criteria: completeness, correctness and accuracy. The methodology for comparing different models is laid out in the work of (Tran et al., 2019). This methodology takes into account several challenges that occur in comparing 3D indoor models. These challenges are: 1) the existence of two standards of representation of 3D indoor models (volumetric and surface, described in section 3.1.1); 2) surfaces in volumetric models that are reconstructed by interpretation due to lack of data; 3) lack of one-to-one matches between source and reference elements; 4) interdependence between the different quality aspects. For more details on these challenges, see the publication of (Tran et al., 2019).

The automatically reconstructed models evaluated in (Khoshelham et al., 2017) were not created in real-time. However, when a real-time element is included, it is necessary to evaluate the model at different times, not just the last model. If a real-time modeling method is needed, such as for obtaining situation awareness, it is necessary that the model can provide the necessary quality and information at any time. So the evaluation of the dynamically generated model in this research should be done in its different stages. Evaluating a model says something not only about its quality or functions, but also about the method that created the model. If the quality is substandard, you need to change the method so that a better model can be generated.

4.1 Geometric quality criteria

4.1.1 Completeness

Completeness is the criterion that tests to what extent the elements of the reference models are found in the source models. A complete model is a model in which all the elements of the reference model are to be found in the source model. Completeness as a score is the proportion of the reference model that overlaps with the source model. This is calculated by taking the sum of the area of the union of the orthogonal projection of each source surface that intersect with a reference surface. This is then divided by the total area of the reference surfaces. A buffer is given to the reference surface to make sure there is a match between the

[6] WG IV/5 (isprs.org)

corresponding reference and source surfaces. In the work of (Tran et al., 2019) a threshold angle of 10 degrees is also used to get only the source surfaces that are parallel up to this threshold to the reference surface.

4.1.2 Correctness

Correctness is the criterion that tests to what extent the elements of the source models are found in the reference models. A correct model is a model in which all the elements of the source models are to be found in the reference model.

4.1.3 Accuracy

Accuracy is the criterion that tests the closeness of the elements in the source models to those of the reference models. An accurate model is a model in which the elements of the source model correspond in location and orientation to their counterparts in the reference model. For measuring the accuracy, (Tran et al., 2019) apply a uniform space sampling on the reference surfaces to get a point cloud of the reference model. The Euclidean distance between the points of the sampled point cloud and the closest source model surface is the basis for measuring the accuracy.

4.2 Service-oriented modelling

Another method for the evaluation of an indoor spatial model, is to look at its functionality. This is applied from the service-based requirements of indoor spatial model of Afyouni's framework (Afyouni et al., 2012). These requirements approach the needs of indoor spatial models in location based services including positioning, navigation and location-aware communication. As the duality concept of the IndoorGML standard allows for both geometrical and topological information, some queries could be applied that tests both. In dual space, the connectivity information on the nodes and edges of the graph can be used to apply a shortest-path algorithm such as Dijkstra's, to get the most optimal route from one room to another. Edges can also contain weights that could be used for path finding algorithms. Other information added to the model can also be of use for navigation. For example, in (Alattas et al., 2018) an IndoorGML-LADM (Land Administration Domain Model) database was created that lists different access levels in a building. In this way, different routes could be generated depending on whether the actor is a student or staff.

Geometrical measurements can be made on the model in primal space. For example, the spatial dimensions of doorways can be measured to check whether an object can pass through it. Geometric measurements can again be related to route determination. Because dual space and primal space are linked, conditions for routes can be set based on the geometry of the spaces.

IndoorGML allows the services mentioned above to be performed on an indoor spatial model. This will be taken into account when creating the model and deciding which elements will be chosen (see more on this in chapter 5.3). In the dynamic or reference models of this study, location-based services are not applied to as this is out of scope (see the section on scope in Chapter 2). However, the possibility of applying location-based services is considered when analysing the result. This will be done by comparing the dynamically generated model with reality as presented in the reference model. Important for the evaluation of a dynamically

generated model, is to evaluate not only the possibility of applying location-based services on final model, but throughout the different stages of creation.

5 Methodology

This chapter explains the method used to create an IndoorGML. First, the outline of the method is explained. It then addresses the dataset and software/technology used. Finally, the method is discussed in detail.

5.1 General approach

A benchmark dataset of (Khoshelham et al., 2021) is used to create an IndoorGML. This is done twice: in a manual and an automatic way done dynamically. The manual IndoorGML, from here on referred to the *reference model*, is created by using Cloud Compare to extract the different floors of the point cloud. This is done by taking samples of different Z-ranges of the floor and ceilings for each floor. This creates several slices that contain the different levels of the building. These slices are then imported in GIS software to digitize the walls of the spaces. The digitized walls are used to create the IndoorGML. Figure 5.3 shows a workflow of the steps involved in making the reference model. The spaces that are defined in the IndoorGML model are classified into the classes of GeneralSpace (e.g., hallways, rooms, toilets) and TransferSpace (i.e., doorways). More information about the different classes of the IndoorGML standard can be found in (5.3).

For the second step, an IndoorGML model is dynamically generated from a sample of the same benchmark dataset as the first step. A method is defined to create an IndoorGML model in near real-time. From here on the dynamically generated IndoorGML models are referred to as *dynamic model*. In figure 5.1 a diagram can be seen that shows the used method for dynamically generating an IndoorGML model. The timestamps of the trajectory and point cloud are used to simulate the scanning process. The trajectory points are iterated over, and the corresponding indoor points are retrieved from the timestamps. Doorways in the dataset are used to identify separate rooms for the indoor spatial model. Doorways are detected by searching for points within a certain radius of the trajectory point. Each time the trajectory passes through a doorway, a new room is modelled using the points that are scanned in between doorways, or in between the current doorway and the start of the trajectory. These points are reconstructed into the surfaces of that room. The method for reconstruction is a RANSAC analysis. The spaces of the doorways are also defined in the dynamic model, but the method for this is different: doorway spaces are built-up from the points that are detected from the radius around the trajectory points that pass through the doorways.

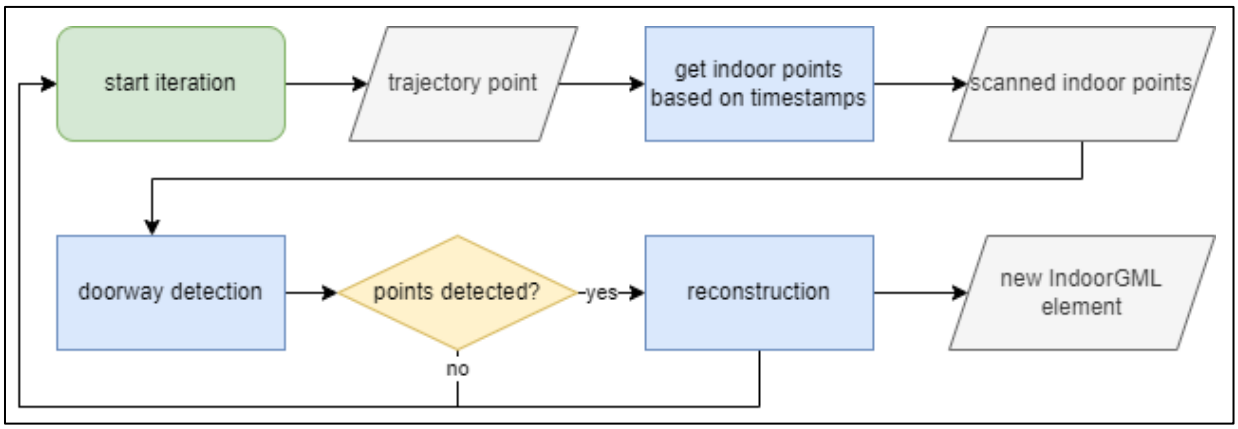


Figure 5.1 - Diagram of dynamically generating an IndoorGML model.

5.2 Data and software

5.2.1 Data

The dataset used in this thesis is the TUB2 dataset from the indoor model evaluation project of the ISPRS (Khoshelham et al., 2017). This dataset contains three files: 1) a scanned point cloud of one of the buildings from the Technical University of Braunschweig, 2) the trajectory of the scan, and 3) an IFC model created manually from the point cloud. The building is a two-story building. Figure 5.3A shows a visualization of the point cloud. The reason for choosing this particular dataset, is that it contains a trajectory, plenty of rooms and that the scan has low clutter. The level of clutter will have an effect on modeling. The more clutter, the more difficult it will be to detect the planes by a RANSAC method. The fact that the scanned building has two stories was also taking into account, but the dynamic modelling of different floors eventually fell out of scope due to limited time. The point cloud of the building was acquired by a ZEB REVO sensor. More information on the specification of the dataset and the sensor can be seen in table 5.1 and 5.2 respectively.

Sensor	ZEB REVO
Number of points	21.6×10^6
mean point spacing (m)	0.008
Color	No
Trajectory	Yes
Clutter	Low

Table 5.1 – Specifications of TUB2 Dataset (from Khoshelham et al., 2017*)*

Max range	30 m
Speed (point/sec)	43×10^3
Horizontal Angular Resolution (deg)	0.625
Vertical Angular Resolution (deg)	1.8
Angular FOV (deg)	270×360
Relative Accuracy	2 – 3 cm
Absolute Accuracy	3 – 30 cm

Table 5.2 – Specifications of ZEB REVO sensor (from Khoshelham et al., 2017)

For the creation of a dynamic model, a sample was taken from the original point cloud. A visualization of this sample and its trajectory can be seen in figure 5.2. The point cloud in figure 5.2 is sliced in order to remove the ceiling and floor for easier visualization. The sample data consists of a point cloud with points belonging to two rooms and part of the hallway. The sample is a selection of a part of the space, which was done by filtering the dataset (i.e., point cloud and trajectory) by a box defined by two sets of coordinates. These coordinates are: minimum XYZ (5.9, -16, -0.9) and maximum XYZ (16.7, -6.2, 1.5). The trajectory starts (A) and ends (B) at the left part of the hallway. The goal of this sample is to create an IndoorGML containing five CellSpaces: 3 GeneralSpaces (the two rooms and the corridor) and 3 TransferSpaces (the three doorways). The reason for using a sample was that it is quicker to test the methodology over part of the data than over the whole point cloud. However, later in the study, the choice was made to make many of the parameters and functions in the method such that they are at least working for the sample, to show that in theory it could also work for the entire point cloud (see the limitations section for more information). The sample dataset contains only rectangular spaces whose walls are aligned to the direction of the X and Y axes. The reconstruction methods are set to these spaces. Although the full dataset follows a Manhattan world assumption, meaning that all surfaces are aligned with the directions of the X, Y and Z dimensions, the method created would not work, because the full dataset contains rooms that are L and T-shaped and thus not rectangular. Literature proves that

reconstruction of more complex rooms (i.e., not following Manhattan world assumption) is possible. However, these reconstruction methods were not applied, and instead a more simple method was applied to the sample dataset to cut time. For future works with different datasets, a more complex reconstruction method should be applied.

Figure 5.2 – Visualization of the sample data with ceiling and floor removed for better visualization. A and B denotes the start and end of the trajectory line. The colours indicate the Z-value.

5.2.2 Software

CloudCompare and QGIS is used to prepare the TUB2 dataset for the manual modelling. CloudCompare[7] is an open-source 3D-software for processing point cloud data. QGIS[8] is an open-source Geographical Information System (GIS).

The Python programming language is used to create a script for the geometrical and topological reconstruction, and for assembling the IndoorGML model. Various modules and packages are used:

- The Open3D library[9] is used to access a wide range of methods used for reconstruction, such as RANSAC analysis. Open3D is also used for its visualization function.
- Geopandas[10] is used to manage the point cloud dataset in Python, by using Geodataframes, the Pandas equivalent of tables but with functionality for spatial data.
- The built-in XML ElementTree module is used to create the XML structure of the IndoorGML models.
- Shapely[11] is used to perform spatial analyses for the detection of doorways.

[7] https://www.cloudcompare.org/main.html
[8] https://qgis.org/en/site/
[9] http://www.open3d.org/
[10] https://geopandas.org/en/stable/
[11] https://shapely.readthedocs.io/en/stable/manual.html

An IndoorGML viewer from an open GitHub repository[12] is used to visualize the IndoorGML files. This viewer is made in Python and visualizes the IndoorGML by plotting the geometry of the solids found in the IndoorGML file. The viewer was edited so that it could also visualize the Edges and Nodes of the topological representation.

5.3 The IndoorGML standard

For this study, not all elements available in the IndoorGML 2.0 standard were included. When making decisions about which elements to include or exclude from the model, it is important to look from the user perspective. In this work, a study of the needs of emergency responders was beyond the scope. A study in which IndoorGML elements are related to needs in emergency response would be a good future work. For now, the intent is to create a simple model that can be used for navigation and spatial measurements. The choices of what elements were to be included in the result were based on these functional requirements. Any elements that are not strictly necessary are therefore excluded. The way you can fulfill these functional requirements may also lie in the data you use for the model. For example, in some cases it may be useful to apply the ObjectSpace class when there are many objects that can cause an obstruction to navigation. The fact that a small sample of the entire dataset was used in this study does not take away from the functionality in terms of navigation and spatial measurements. This is because the spaces of the sample match the characteristics of the entire dataset, namely that the data is clutter-free and there are no objects such as furniture. Other more technical reasons are explained in sections (5.3.1 and 5.3.2). Appendix A and B show two UML's of the standard (of the core and navigation module). Marked with a green checkmark are the elements that are used for the models created in this research. Marked with a red cross are the elements left out of the research. An overview is given below in three tables of the different element and classes of the standard. The tables show the names of the element and a description. Two classes that are included in every IndoorGML file are: *IndoorFeautures* and *ThematicLayer*. Table 5.3 shows an overview of the mandatory classes.

Class name	Description
IndoorFeatures	This is a mandatory class. It is the uppermost abstract layer that contains one or several layers of the ThematicLayer class.
ThematicLayer	Mandatory class that contains the primal and/or dual representation of a certain space. There can be defined more than one thematic layer of space. For example, a layer of the topographic space, and a layer of the Wi-Fi-space that defines the range of the Wi-Fi network.

Table 5.3 – Overview of mandatory IndoorGML classes

The *InterLayerConnection* class connects different thematic layers. This element is excluded because only one thematic layer is modelled in this study. Three other elements that are excluded are: *ExternalSpaceObject*, *ExternalBoundaryObject* and *ExternalObject*. These elements are used when a reference to an external document is made, such as a BIM model. This is not the case for this study and thus were excluded.

[12] https://github.com/assankhanov/IndoorGMLViewer

5.3.1 PrimalSpace

Both primal space and dual space are defined for this research. The primal space contains the Euclidean geometries of the cellular spaces. The following classes are used: *PrimalSpaceLayer*, *CellSpace* and its children from the navigation module (*NavigableSpace*, *GeneralSpace* and *TransferSpace*). In table 5.4 an overview is given of the different element and classes belonging to the primal space representation.

Class name	**Description**
PrimalSpaceLayer	Abstract layer containing all the elements belonging to the primal representation.
CellSpace	Defines the indoor space in terms of cellular space. In this study the geometry of the CellSpace is defined by the surfaces that bound this space, as is inherent in a surface-based representation of indoor spaces (see chapter 3.1.1)
NavigableSpace	Child class of CellSpace, contains the children General- and TransferSpace
GeneralSpace	Child of NavigableSpace used for larger spaces, such as rooms and hallways.
TransferSpace	Child of NavigableSpace used for the spaces between GeneralSpaces.

Table 5.4 – Overview of the classes belonging to the primal space.

The *CellBoundary* class represents the boundaries of CellSpace. This class and all its child classes are excluded because not all surfaces belonging to the walls are visible from the dataset. For this reason a surface-based representation was chosen that defines the surfaces within the geometry in the CellSpace class. The cell boundaries are also not necessary for the purpose of navigation or positioning. The *NonNavigableSpace* and its child class *ObjectSpace* are also excluded from this research. These two classes define spaces that are non-navigable, such as furniture objects that may occupy the general space. In the dataset, there are few objects that are non-navigable. Also, these classes are not strictly necessary for creating a dynamic IndoorGML model. For these reasons, they were not included in the study.

5.3.2 DualSpace

The dual space contains the indoor network that is derived from the primal space. The following classes are used: *DualSpaceLayer*, *Node* and *Edge*. In table 5.5 an overview is given of the different element and classes belonging to the dual space representation.

Class name	**Description**
DualSpaceLayer	Abstract layer that contains the network graph of the indoor spaces. The network is defined by nodes and edges
Node	Represents the cellular space in the network. The geometry of a node is defined here by taking the center of its CellSpace.
Edge	Represents the connectivity between the CellSpaces. An edge geometry is made up by the coordinates of the two nodes.

Table 5.5 – Overview of the classes belonging to the dual space.

The *Route* class is a subset of the Nodes and Edges that comes from a path query. It is not within scope to run a path query on the model, so it is left out.

5.4 Reference model

In figure 5.3 a visualization can be seen of the different steps involved in creating the reference model. To create the reference model, the different floors were separated using the slice tool in CloudCompare (see image A). Both floors were then sliced in a way that only the walls are left visible by slicing out the points belonging to the ceiling and the floor (image B). These slices were then exported as a .las file and imported in QGIS to digitize the walls (image C). Looking at the original point cloud, the Z values of the ceiling, floor and doorways were determined by point picking and were added as tabular data to the digitized walls layer in QGIS. A python script was then used to create a floor and ceiling polygon for every CellSpace. This was done by looping through every neighbouring pair of vertices, and then connecting them with their floor or ceiling counterpart to create the exterior linear ring of the walls. All polygons together then forms a polyhedron consisting of floor, ceiling and walls (image D). Using the built-in xml library of Python, the polyhedron was then defined as a solid geometry in GML. The other parts of the IndoorGML were also written into an XML through Python. A workaround was attempted by transforming the IFC benchmark models from (Khoshelham et al., 2021) to IndoorGML using the ifc2indoorgml tool. However, the working of this tool requires the IFC model to contain *ifcSpace* elements, which is lacking in the benchmark model. [13]

[13] GitHub - grid-unsw/ifc2indoorgml: A tool allowing to generate IndoorGML files from IFC input models

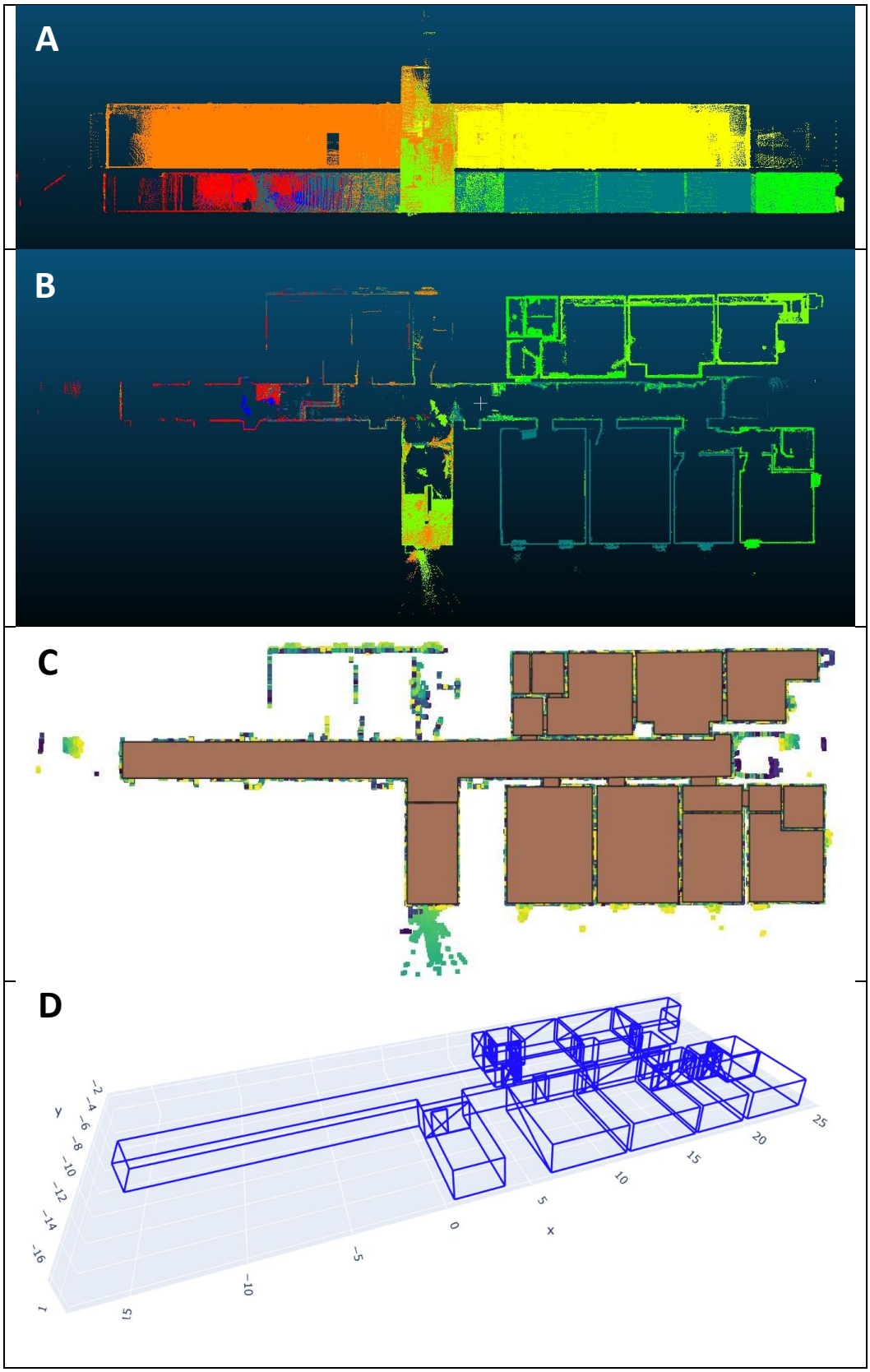


Figure 5.3 – steps involved in making the reference model. A – Side view of the TUB2 building point cloud in CloudCompare; B – Point cloud of the ground floor of the TUB2 building after slicing; C – Digitization of the ground floor in QGIS; D – Final reference model.

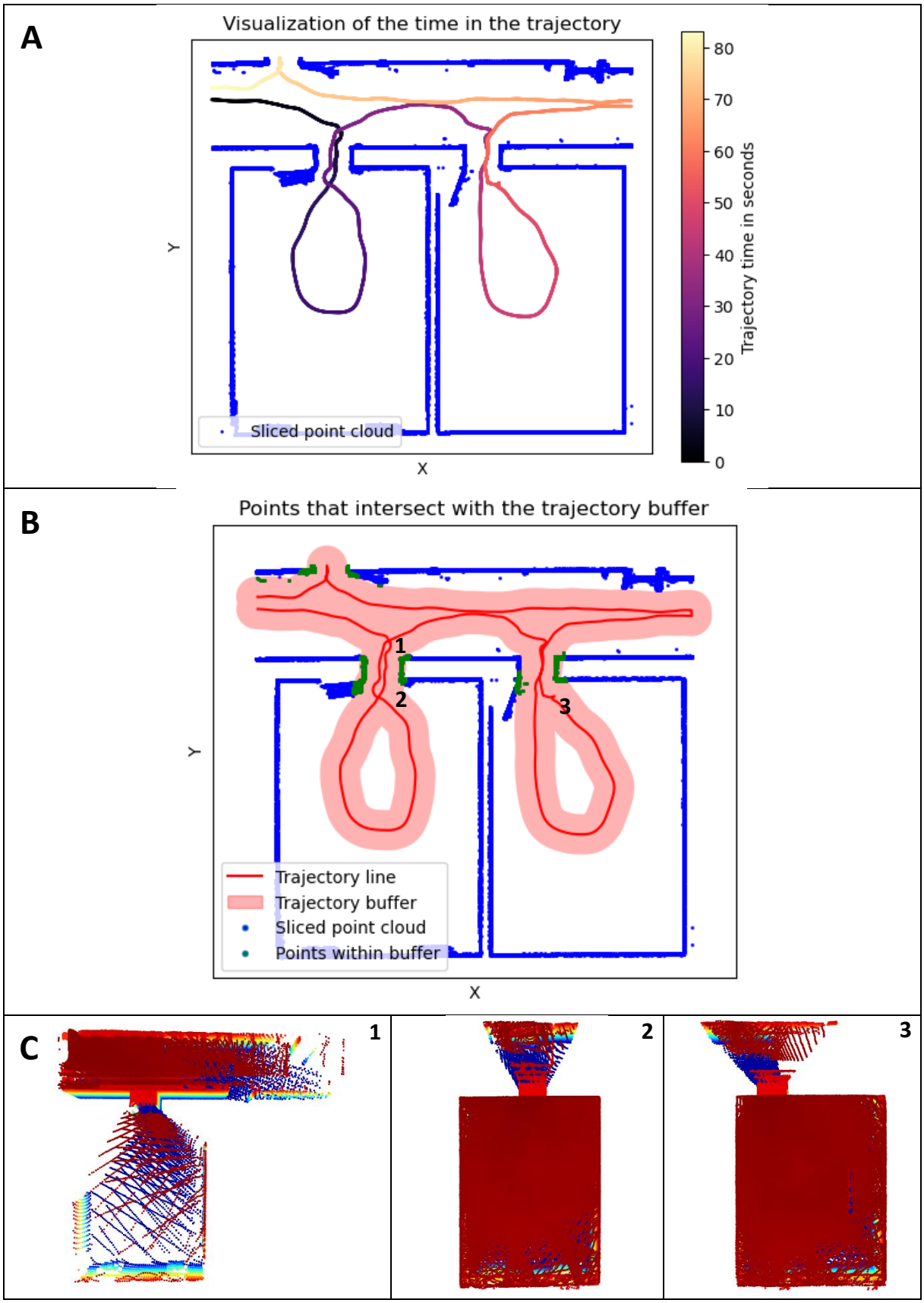


Figure 5.4 – Workflow of the dynamic process of creating an IndoorGML model: A – Trajectory points colored by the timestamps of the trajectory points; B – Trajectory points including a buffer of 0.5 meter around all the trajectory points; C – Points queried at the numbers 1, 2 and 3 of the numbers (see figure B).

5.5 Dynamic model

Figure 5.4 shows a workflow to better understand the method used to make the dynamic model. To make the dynamic model, the time stamps of the trajectory points and point cloud points are used as a way to build the model incrementally. Both the points of the point cloud and the trajectory are read into geodataframes. Each point of the point cloud is connected to a point from the trajectory from which it was collected, based on the timestamps of the point cloud and the trajectory. This means that every point in the point cloud has the ID of its trajectory point. To simulate a real-time situation, the trajectory points are iterated over. In image A of figure 5.4 it shows the time of the trajectory. Each iteration of the trajectory points picks out and add the points of the point cloud with an ID up to the current iteration. This way the point cloud is revealed incrementally.

5.5.1 Doorway detection

The trajectory is used to detect doorways in the point cloud, a method that has been used with success in other works (Nikoohemat et al., 2018; Staats et al., 2019). Detected doorways are used to determine when to query part of the point cloud for reconstruction. Door detection helps to build the IndoorGML model incrementally in near real-time, and the use of door detection makes it easier to divide the spaces in the building into separate cell spaces. The method of detection is done by applying a neighborhood search on the trajectory points to search for nearby points from the point cloud, similarly to the moving windows method often used in analyzing raster data (Hagen-Zanker, 2016). A cylinder with a 50 centimeter radius and a Z-range of 20 centimeter is used with the trajectory point at its center. These parameters were chosen after trial-and-error. Figure 5.4B shows all the cylinders around the trajectory points plotted as a 2D image. A new geodataframe is created and filled with the 'detected' points (i.e., those that are located within the cylinder), as well as the points that are detected in successive iterations. The points in this geodataframe may form a potential doorway. When the iteration arrives at a section in the trajectory that does not detect any potential doorway points, a new geodataframe is created for the next potential doorway to be detected.

As the same doorway can be detected several times, it should be taken into account that several geodataframes of potential doorways may contain points belonging to the same doorway. This is done by checking whether the points in the successive potential-doorway-geodataframes overlap with each other. It is checked whether a point with average XY values of the current potential-doorway-geodataframe lies inside a 1-metre buffer around the bounds of the previous potential-doorway-geodataframe. Figure 5.5 shows an example of this. If this is the case, then the two geodataframes are merged, and otherwise the previous geodataframe is used to create a TransferSpace for the IndoorGML using geometric reconstruction (see chapter 5.5.3). The lacuna in this method, is that it cannot take into account a trajectory passing through the same doorway at a later time (i.e., not sequentially) when the doorway is already stored as a real-doorway. If this happens using this methodology, it would then be stored twice in the model. However, it does not occur in the dataset of this study that the trajectory passes through a doorway two or more times in the way mentioned above.

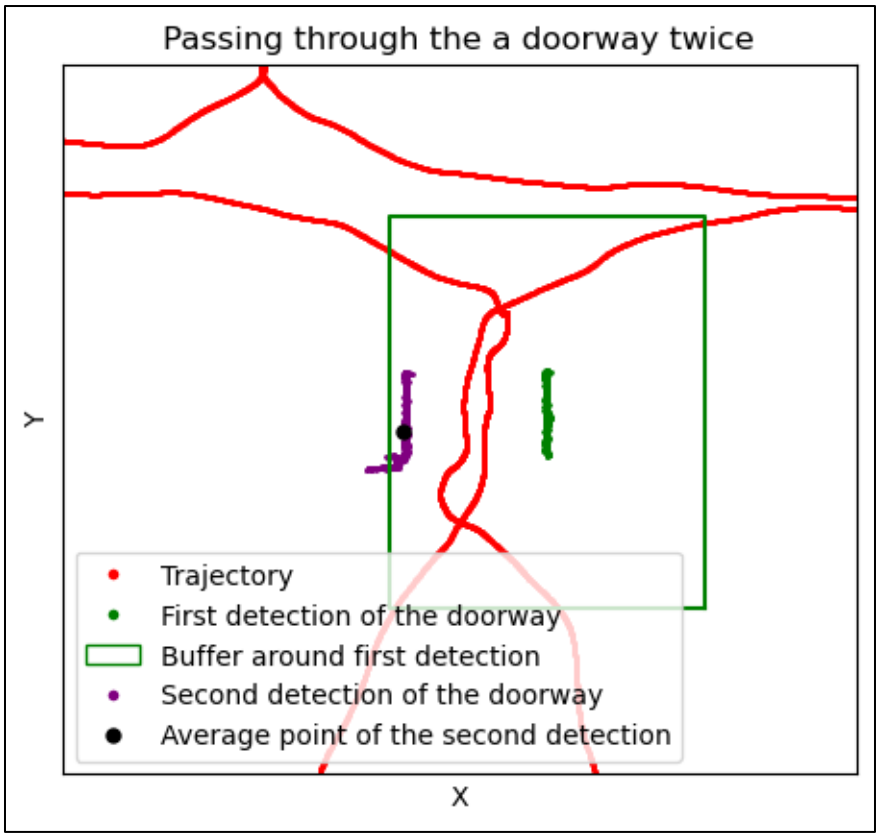


Figure 5.5 – Door detection of the first doorway of the sample dataset.

Because the neighborhood search method only takes a slice of 20 centimeters from the trajectory point, the points of the doorway that lie above or below the slice are not included. These are needed to reconstruct the doorways for the IndoorGML model. To get these anyway, the following method is used. A square is created around the detected doorway points. The square is used to clip all points that belong to the doorway. The result is a geodataframe that contains all the points of the doorway.

5.5.2 Extracting rooms

Detecting doorways is also essential for reconstructing the rooms. When a new doorway is detected at a trajectory point, the point cloud points scanned from all previous trajectory points in between two moments of doorway detection are added to a GeneralSpace-geodataframe. Figure 5.4C shows the visualization of three GeneralSpace-geodataframes. The GeneralSpace-geodataframes are linked to another dataframe (from here on called the GeneralSpace collection) that contains information about the GeneralSpaces, such as which doorways this GeneralSpace is connected to, or which other GeneralSpace it is similar to. Again, there is a possibility that this space, in this case a room, has already been scanned a previous time. As with the doorways, a buffer is created to check whether the rooms are duplicates. For the GeneralSpace this is a buffer of 3 meter around the average point of the just stored GeneralSpace. Again, a check is made to see if it contains the average point of the previous scanned GeneralSpaces. If so, a match is indicated in the GeneralSpace collection in a column of arrays. Each time a space is scanned for the first time, it is designated as the real-GeneralSpace in the GeneralSpace collection. The points of the subsequent GeneralSpaces

that are equal to the real-GeneralSpace are concatenated to the geodataframe of the real-GeneralSpace, which will be added to the IndoorGML model.

These methods of detecting doorways and extracting points for the GeneralSpaces based on these detection result in a number of geodataframes that contain the points of the general spaces and the transfer spaces. These are then used to apply the reconstruction methods to gain the primal spaces used in the IndoorGML file.

5.5.3 Geometric reconstruction

When a real-GeneralSpace is identified, the following methods are applied to reconstruct a simplified geometry. A horizontal slice is created from the point cloud. The height range of this slice is between 1.5 and 1.6 meter from the floor. To get this range, the z-value of the floor is needed, which is obtained by taking the average z-value of the 1000 lowest points. The advantage of making a slice in this range is that it removes some factors of noise from the point cloud. Noise is defined here as all points that do not contribute to obtaining the geometry of the walls. At the range between 1.5 and 1.6 meters, almost no objects can be found.

A RANSAC analysis is applied twice to the horizontal slice to discern the two walls as planes in the points. After the first application of RANSAC, the points that fit to the model found are removed from the dataset that is used as input in the second application of RANSAC. The RANSAC method here consists the following steps[14]: 1) it randomly selects a subset of the points, 2) then it fits a model (i.e., the wall) to the subset of points, 3) it determines the amount of outliers. 4) These three steps are repeated a set number of times, and the best model (i.e., with the least amount of outliers) is chosen. Because of the element of randomness in the RANSAC method, the result of the reconstruction may differ if you run the method several times.

After discerning the set of points of the two wall planes, the corners of the space is then gathered by extracting the minimal and maximal X and Y values of the planes. The minimal and maximal Z values is gathered from the points of the GeneralSpace geodataframe. This method of getting the corners only works in two condition: that the room is rectangular, and that there is a Manhattan world assumption (i.e., walls cannot be diagonal). The TransferSpaces are reconstructed in the same way as the GeneralSpaces. The result is 8 corner coordinates of a room (of the floor and of the ceiling). These can then be transformed into a solid geometry in a similar way as was done in the reference model (see section 5.4).

The method of statistical outlier removal (SOR) is used to remove noise from point clouds by removing points that are far away from neighboring points. An SOR method comes with the O3D Python library.[15] This SOR method removes the points that are further away than an average distance between neighboring points. It takes two input parameters: 1) **nb_neighbours**, the amount of neighbors that are taken into account for calculating the

[14] https://www.mathworks.com/discovery/ransac.html
[15] Point cloud outlier removal — Open3D latest (664eff5) documentation

average distance, and **2) std_ratio**, a number that sets the threshold level based on the standard deviation of the average distances across the point cloud.

5.5.4 Topological reconstruction

Topological reconstruction is performed using the reconstructed geometries of the doorways and rooms. For the coordinates of a Node, the center of a CellSpace is taken. In this study, there are only rectangular spaces, so the center is always within the space. For concave spaces, a different method should be used to ensure that the node is inside the space. To know how the different nodes are connected, the information contained in the GeneralSpace-collection dataframe is used. This shows which doorway each room is connected to. One or more edges can thus be reconstructed for each room by taking the node-coordinates of the rooms and the connected doorways.

5.5.5 Performance issues

The sample dataset contains more than 2 million points. Functions such as selecting and spatial analysis (i.e., door detection) are applied to the dataset. With a dataset of this size, this can take a lot of computing time. Certain measures were taken to increase the performance of the script.

The method of this study iterates over the trajectory points, executing the door detection function for every point in the trajectory. Because of the time it takes to execute this function every iteration, a choice has been made to only do it every 10 iterations. 10 iteration of the method is on average 0.075 seconds. A run of the sample model without this performance measure set took 9 minutes and 53 seconds, while a run with the measure activated took 29 seconds. The resulting models of the two runs were not different from each other. The total duration of the scanning process for the sample part of the dataset is around 83 seconds. The average distance between two consecutive points in the trajectory is 0.56 centimeters. The standard deviation is 0.22 centimeter. With a z-score threshold of 2 only one outlier was found of 14 centimeter. This means that door detection is performed every 5.6 centimeters. The effect on the results from the sample is minimal, but on other datasets it could be larger, especially when the trajectory points are further apart. It might then be necessary to optimize in other areas to obtain better performance. It is important to note that the distances between subsequent points are not evenly distributed over the trajectory. This mean that the quality of the method is less consistent. A better method to increase the performance, would be to limit the iterations based on a distance threshold. For example, to do an iteration every time that the cumulative distance of the subsequent points in the trajectory reach more than 5 centimeters. Doing this would assure more consistency along the trajectory.

For detecting the points from the trajectory, only those points of the building are used that are from the last 10 trajectory points. A sample dataframe is created from the building points dataframe. Every time a new trajectory point is reached, the sample dataframe is supplemented with the points that are scanned from this trajectory point. When building points from more than 10 trajectory points are in the sample dataframe, the points first in line (i.e., from a trajectory point that is earlier in time) are removed.

5.6 Building an IndoorGML file

The built-in Python module of ElementTree was used to build the IndoorGML file. The file can be seen as a tree, with a root (i.e., IndoorFeatures class) containing several branches. The two main branches are the PrimalSpaceLayer and DualSpaceLayer classes, which contain the geometric and topological representation of the model, respectively. In ElementTree the root is created using the Element() function, which takes the class name as the argument. Each subsequent element is created as a subelement with the Subelement() function, which takes as a first argument the parent element and as a second argument the class name. An element can also define the attributes (e.g., the 'gml:id'), and the text within the element (e.g., the coordinates within a pos element). If the solids of a GeneralSpace or a TransferSpace are defined, a function is executed in the script that creates a GeneralSpace or TransferSpace element from these solids with associated information (i.e., the CellSpace type and id).

The advantage of this tree structure is that it can be easily iterated over several branches as children of a parent element. If you are looking for a GeneralSpace with a specific id, you can find it by passing over each branch of the PrimalSpaceLayer until you find the GeneralSpace with the id you are looking for. This way, a CellSpace will be found in the file if it needs to be updated, for example when the trajectory passes this CellSpace for a second time. The old geometry is then reset, and the geometry of the newly defined solid is then updated to the file.

The building of the DualSpaceLayer is done parallel to the creation of the PrimalSpaceLayer. When a new CellSpace is defined, or an old one is updated, the associated Node will automatically be created or updated. For updated Nodes, the coordinates of the Edges are also automatically updated. The Edges are created parallel to the TransferSpace. When a new TransferSpace is created, an Edge will be defined at the same time.

6 Results: comparing the reference and dynamic IndoorGML models

The models are visualized through a viewer that uses Plotly for plotting graphs (see 5.2.2). It parses the CellSpaces, Nodes and Edges from the IndoorGML file, and then plots the models as a 3D graph. The GeneralSpaces are visualized in blue, the TransferSpaces in yellow, and the topology is given a red color. In the models each CellSpace has one side with a diagonal line. This is an effect of the viewer, and does not reflect the data in the model.

6.1 Reference model

The IndoorGML models of the reference model can be seen in appendix C. The first image of appendix C visualizes the primal space of the reference model, including the ground floor and first floor, while the second image depicts the primal and dual space, including only the ground floor. The reason for excluding the first floor in the second depiction is that including it resulted in a confusing picture from which not much could be made out.

6.2 Dynamic model

6.2.1 Modelling GeneralSpaces

In the figure 6.1 below there is a visualization of the points belonging to a CellSpace. A room is clearly discernible. The points on the left of the image are part of the corridor that is scanned from inside the room. This is noise that is taken care of in order to extract a simplified geometry.

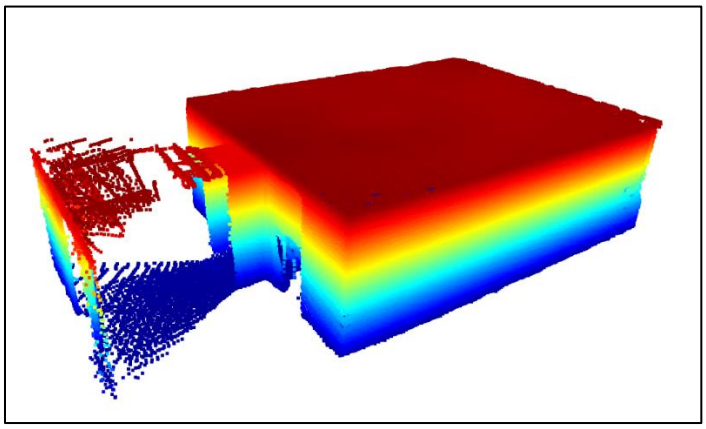


Figure 6.1 – All points queried between two doorway detections.

To remove this noise, a statistical outlier removal was tried. The result can be seen in figure 6.2. The red points are the outliers that are removed, and the grey points are the inliers that remain after the statistical outlier removal. Outliers are points that are too far from their neighbors. The results are mixed. Even though it removes most of the noise on the left, it does not remove everything. It also removes many points on the right. The reason for this is that the surfaces scanned further away are less dense than surfaces that are scanned closer, resulting in many points being located far from their neighbors. Because of the mixed results, the choice was made to leave out the statistical outlier removal method.

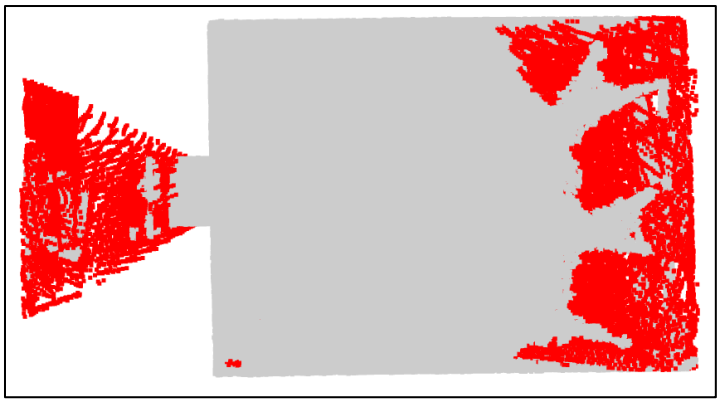


Figure 6.2 – Result of the SOR. The red points are outliers and grey points are inliers.

Figure 6.3 shows the result of slicing the point cloud of the room. Doing this significantly reduces the noise without removing parts of the point cloud needed for the correct geometry.

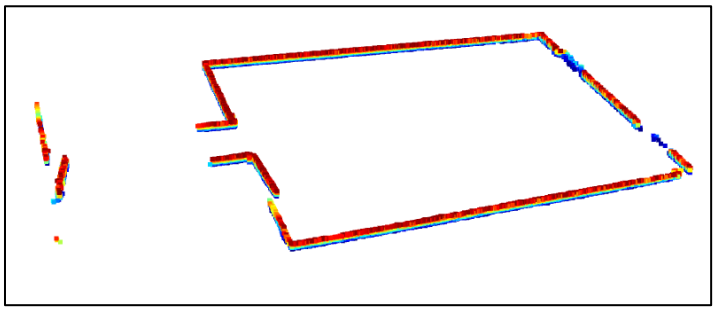


Figure 6.3 – Slice of the GeneralSpace.

In figure 6.4 you see the results of the RANSAC method applied twice to the sliced point cloud from figure 6.3. Two walls opposite of each other are segmented as planes. From these two walls, together with the first point cloud of the room, the necessary coordinates can be obtained for the geometry in IndoorGML. This is done by taking the minimum and maximum of the X and Y from the walls. This method is feasible only for interiors that follow the Manhattan space assumption and are oriented along the right axes. The Z coordinates for the GeneralSpace are taken from the point cloud.

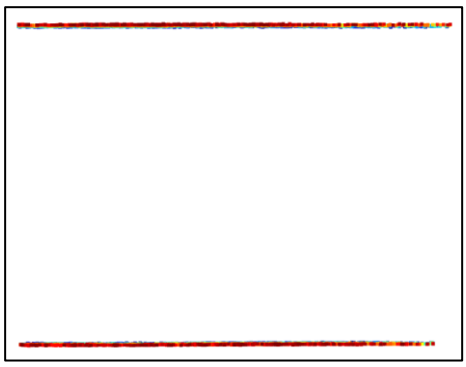


Figure 6.4 – Walls from a GeneralSpace resulting from the RANSAC method.

6.2.2 Modelling TransferSpaces

The initial points of the transfer spaces are those points of the building that are detected by the neighborhood search from the trajectory. Because the trajectory usually passes through an opening or door twice, the points of the transfer spaces that are located in the same space are merged together. This results in a set of points as seen in figure 6.5.

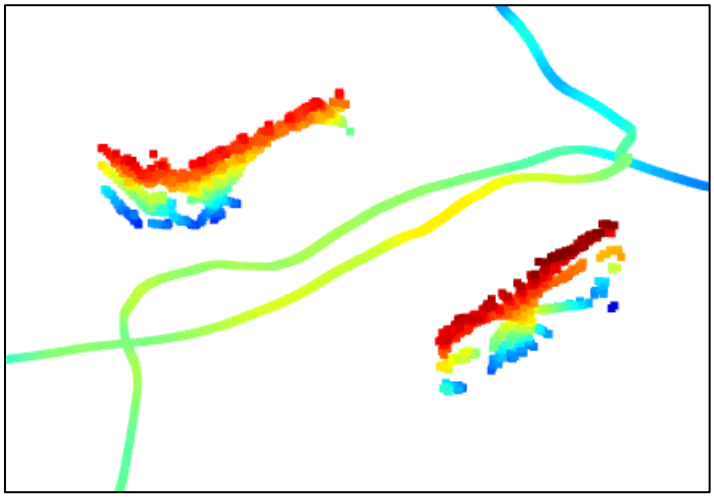


Figure 6.5 – door points as detected from the trajectory (trajectory points are added for orientation purposes).

In figure 6.7 the result can be seen of gathering all points from the doorway. To do this, a square is fitted around the detected doorway points (e.g., those in figure 6.5) and then this square is used to clip the points from the points scanned up till then.

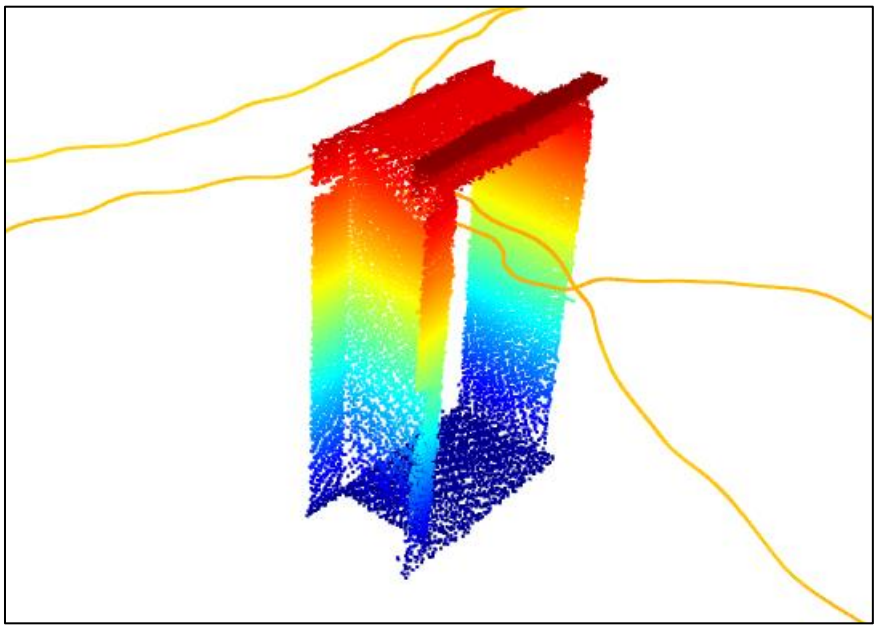


Figure 6.6 – Doorway points after clipping (trajectory points are added for orientation purposes).

In the resulting doorway points, there is still some noise left. Also here the SOR function of o3d is tried to filter out the noise. However, it was not possible to find the right parameters for the SOR function that works well for both doorways. The choice was made to leave in the noise, meaning that the Z-value of the doors are incorrect. As can be seen in figure 6.6 it includes some points on the top that are part of the roof of the GeneralSpaces, as well some

points of the GeneralSpaces walls. This results in that the geometry of the TransferSpace in the final model is larger than it is in reality. However, it can still be seen as a TransferSpace and used for navigation.

6.2.3 The resulting IndoorGML model

The final result of the dynamic model of the sample can be seen in figure 6.7. In figure 6.8 the same area of the sample model is shown in the reference model. The visualization of the dynamic model as it is built incrementally can be seen in Appendix D. In Appendix D, you can see the IndoorGML file of the dynamic model. Here the CellSpace, Node and Edge elements are collapsed. An example of the IndoorGML of these elements can be seen in Appendix F and Appendix G. Almost all elements that exist also in the reference model can be seen in the final dynamic model. This means that the model is high in completeness. Only the doorway as seen at the top of figure 6.1 is not included in the model. The resulting model has the correct nodes and edges, which means that a path query is possible. There does seem to be a problem with the accuracy of the model: the TransferSpace on the left in figure 6.7 is smaller than it should be. This is because only one side of the doorway was scanned. The same TransferSpace also does not touch the ground. The TransferSpaces also seem to overlap with the spaces of the rooms and hallways, which makes the model geometrically incorrect. These problems limit the accuracy of the model and thus limit the possibility of accurate spatial measurements on this indoor model.

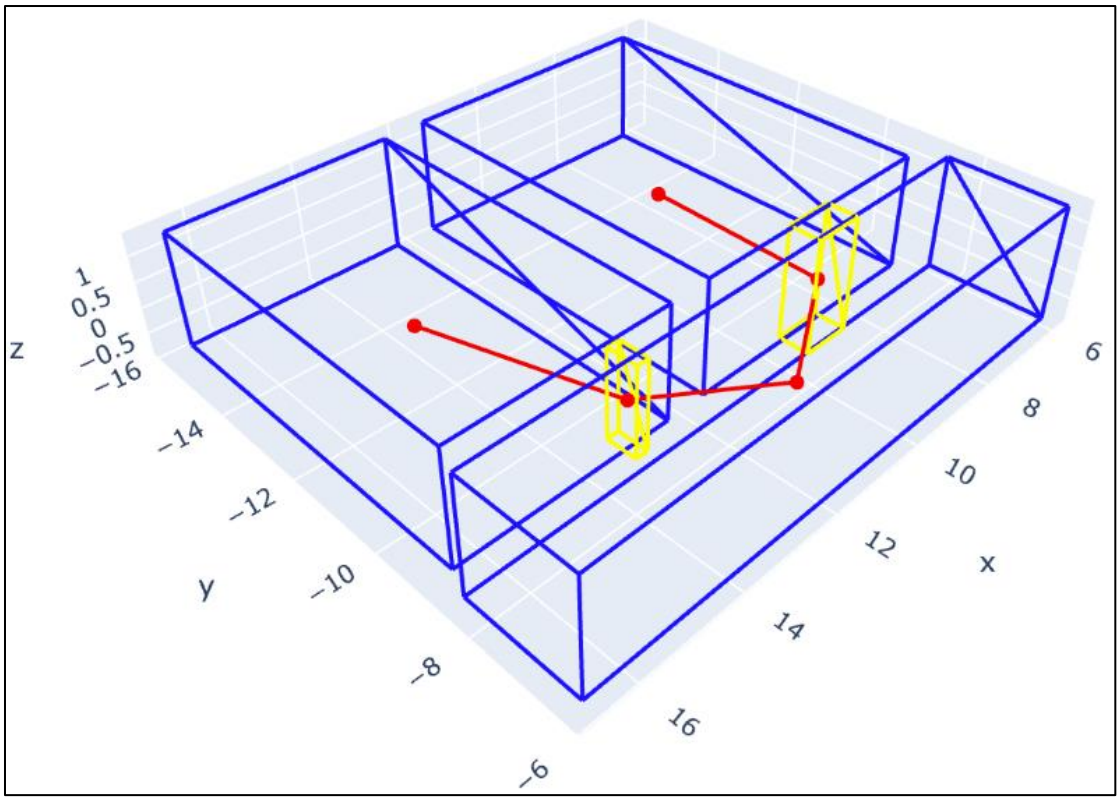


Figure 6.7 – the final dynamic model.

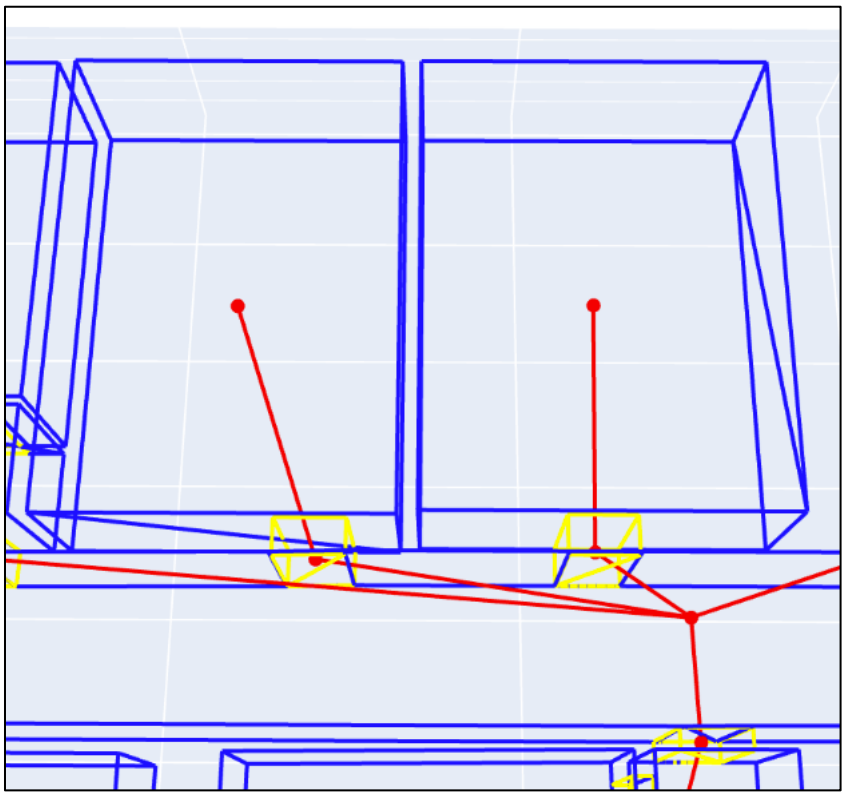


Figure 6.8 – the same spaces included in the sample as shown in the reference model.

The dynamic reconstruction of the interior space can be seen in appendix D. The model was updated every time a GeneralSpace and TransferSpace was identified, the numbers show the order of updates in the model. To get more insight on when a particular update was done during the trajectory, a visualization was made of all the moments in the trajectory that the model was updated. This can be seen in figure 6.9. The green parts of the trajectory are the moments in the trajectory in which an update to the model is made, the numbers in the figure indicate which update. These numbers correspond to the numbers in appendix D.

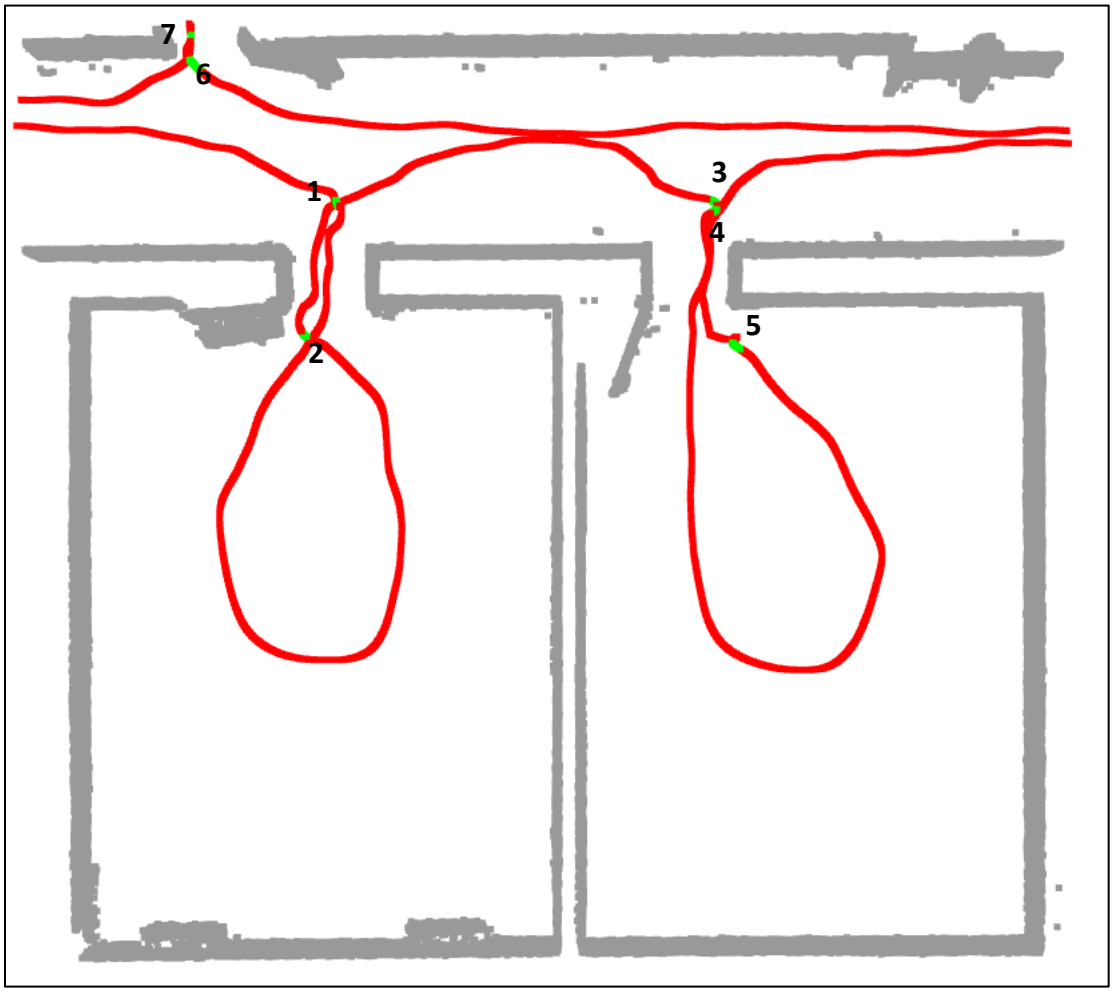
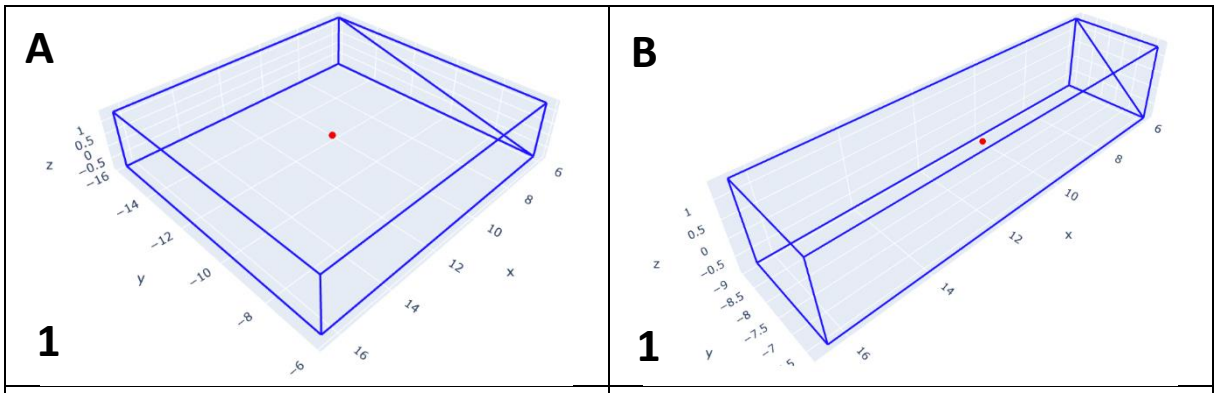


Figure 6.9 – the moments in the trajectory with updates to the dynamic model.

As the trajectory starts in the hallway, this is always the first space that is reconstructed. The script was executed several times to test for the possible effect that the element of randomness from the RANSAC method could have on the model. It shows that in different runs, the geometry of the rooms differ only slightly. For example, the average difference between two runs of the coordinates in the floor plane of the hallway is approximately 0.003 meter. Meaning that every corresponding coordinate only shifts 0.003 meter in different runs. In some runs the difference were shown to be higher, as the geometry of the hallway is sometimes reconstructed highly inaccurate in the first update of the model (see figure 6.11A), something which is corrected in a later update (figure 6.11B). The reason for this inaccuracy is that point cloud used for the first update also includes points from one of the rooms scanned through the door. During reconstruction it accidentally reconstructs the wall of the room, and

the coordinates the method then picks for the simple geometry are those of both the hallway and of the room. The hallway is updated, as the amount of points used for reconstructing the hallway increases incrementally since the trajectory passes through the hallway several times. This allows for the next reconstruction of the space to be more accurate. The inaccuracy of the hallway seems to be a consistent fact in every run, as can be seen in Appendix D, although the degree of inaccuracy varies. In the fifth update of the model it can be seen that the hallway space overlaps with that of the doorway. This is corrected for in the sixth update.



Inaccurate first reconstruction of the hallway	More accurate reconstruction of the hallway

Figure 6.10 – first model update in two different runs.

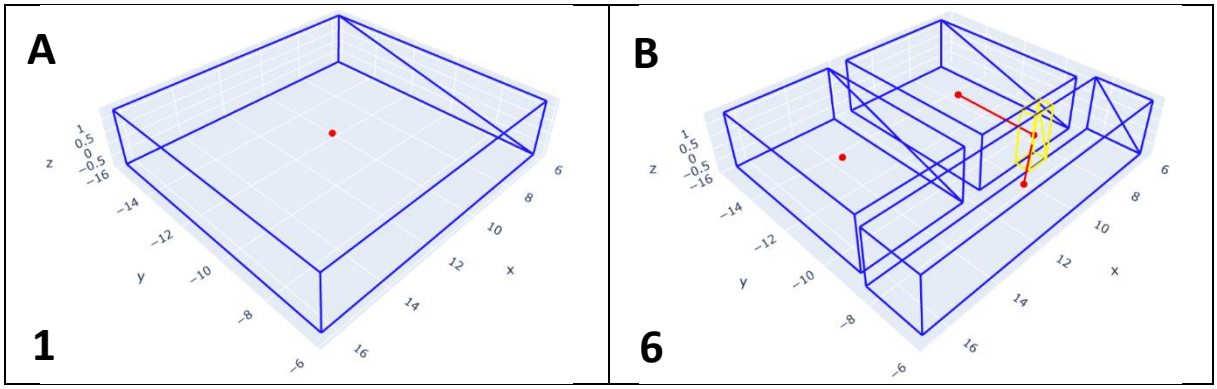


Figure 6.11 – update 1 and 6 of the same run.

In update two and three there seem to be no visible changes made to the model. However, this is a small update of less than half a centimeter of the hallway made when the trajectory passes through the hallway for the second time.

The Nodes are constructed together with every new CellSpace. The Edges are constructed whenever a new TransferSpace is reconstructed. As described in the methodology, a TransferSpace is created when doorway detection is false at a point in the trajectory but is true at the previous point. For space creation, it is the other way around. This is why the TransferSpace at the one doorway is only created at the other doorway: because when a doorway is shown to be at a different location than the previous detected doorway, the previous doorway is modelled into the IndoorGML. The same can be said for the doorway at the number 4 of figure 6.10 that is generated at the doorway near number 7. This method is problematic, as it does not model the last doorway, because after detecting that doorway, there is no other doorway detected. It also limits the Situation Awareness that could be gotten

from the model at any time, as the lass doorway that is walked through is not yet modelled or might not be modelled at all.

7 Conclusion

In this study, an IndoorGML model was generated in near real-time from a point cloud. This point cloud already existed and was created by a research team at Braunschweig University of Technology (Khoshelham et al., 2017). The dataset of the point cloud includes a trajectory of the mobile laser scan that was used to scan the point cloud. The main question of the study was:

To what extent is IndoorGML usable as an indoor modelling standard in the creation of an indoor spatial model in near real-time.

To answer this, we must first arrive at an answer of the sub-questions.

7.1 First sub-question

How to evaluate a dynamically generated IndoorGML model?

Existing literature on evaluating indoor spatial models was reviewed. Two studies were taken in this regard. The first study by (Khoshelham et al., 2017) focuses on evaluating the geometric quality of indoor spatial models by looking at the completeness, correctness and accuracy of a model. These three criteria are measured by comparing an automatically generated model with a benchmark model. Such a benchmark model was also created in this study, and is referred to as the reference model. Completeness is the criterion that tests to what extent the elements of the automatically generated model are found in the benchmark model. Correctness is the criterion that tests to what extent the elements of the benchmark model are found in the automatically generated model. Accuracy is the criterion that tests the closeness of the elements in the benchmark model to those of the benchmark model. The second existing framework deals with the added value of indoor spatial models for performing spatial services, such as navigation and spatial measurements (Afyouni et al., 2012). Evaluating an indoor spatial model involves not only evaluating the geometric quality of the model or its functions, but also evaluating the workability of the method that generated the model. Here it is good to have a benchmark model for comparison. In this study, a model is generated in near real time. It is therefore good here to look not only at the final model, but also at all temporary models generated during the scanning process. The reference model was created manually by digitizing the rooms to a 2D map file. The 2D features of this file were converted to 3D by adding the Z dimension. Then the model was converted to an IndoorGML model and the dual space (i.e., a network graph) was generated from the primal space (i.e., the Euclidean geometry).

7.2 Second sub-question

What are the necessary steps to implement IndoorGML in near real time?

A dynamic (i.e., automatic and in (near) real-time) modelling method was applied on a small sample of the dataset containing 3 GeneralSpaces (i.e., the rooms and hallway) and 3 TransferSpaces (i.e., the doorways). A simulation approach was used on an existing dataset through simulating the scanning process by using the timestamps in the point cloud and trajectory. Through the timestamps the points of the point cloud can be connected to the points in the trajectory. The method iterates over the trajectory points and for every 10

iterations it searches for nearby points from the point cloud. If a point is found, a door is detected and a space is reconstructed from the point cloud points that are scanned up until the detection. For the reconstruction a RANSAC analysis method is used. From the reconstructed space a GeneralSpace element is generated for the IndoorGML file. The doorway spaces are reconstructed when a point is detected that is outside the range of the previous detected point. The network graph is also generated dynamically. A Node element is created every time a CellSpace (i.e., TransferSpace and GeneralSpace) is generated. Edges that connect the nodes are created whenever a new TransferSpace is generated. Everytime a new IndoorGML element was added, a new IndoorGML file was created.

7.3 Third sub-question

How does the in near real-time created IndoorGML model compare against a reference model?

The resulting models were compared to the reference model. Not only the final model, but also the IndoorGML models belonging to earlier steps of the scanning process. In the final model, all spaces but one TransferSpace were reconstructed in an IndoorGML. The TransferSpaces were not geometrically correct and were overlapping with the GeneralSpaces. The result is a model that was limited in accuracy, but high in completeness. This method does quite well in dynamic modeling of the GeneralSpaces, but the TransferSpaces are generated only when the current point in the trajectory is far away from the TransferSpace in question. This also effects the topological space, as an Edge is only created when the TransferSpace to which it is connected to is created. This limits the value of the method, because essential parts of the IndoorGML model are not generated when they are scanned. Still, these elements can be delivered during the scanning process, but with a delay. Some elements of the model changed dimensions along the scanning process. This is because the trajectory goes through some spaces multiple times, and therefore stores more information about that space and thus can generate a more accurate model. An important gap in this method is that it cannot account for a trajectory that passes the same doorway multiple times in a non-successive manner, as it will only correct for duplicated doorways that are detected one after another. The method is far from perfect, but the result does show that it is possible to generate an IndoorGML from a point cloud in a way that is near real-time.

7.4 Main question

To what extent is IndoorGML usable as an indoor modelling standard in the creation of an indoor spatial model in near real-time.

This study proves that to implement an IndoorGML in near real-time is possible in a simulation. The final model contains a primal space with a geometrical representation that allows for spatial measurements of different elements in the model, and a dual space with a network graph that enables navigation. The models belonging to the steps that lead up to the last model were not complete despite the information (i.e., the points already scanned) that could be enough in these steps to manually create a more complete model. Other reconstruction methods could potentially solve this problem. When defining the usability of an indoor modelling standard in terms of its geometric qualities and the functions it can provide (i.e., navigation and spatial measurements), then the study proves that the IndoorGML model is usable as an indoor modelling standard in creating an indoor spatial model in near real-time.

Limitation in the usability of the standard is due more to the method than to the standard itself. The unique contribution of this work is investigating how to create an indoor spatial model in (near) real-time, and how to evaluate such models. The research has highlighted several challenges. More research needs to be done to overcome these challenges and to see whether a dynamic model can be generated in a real life situation.

8 Discussion

8.1 Strengths and limitations

8.1.1 Strengths

This research takes a first step toward dynamically creating an indoor space model. It proves that it is possible to build an IndoorGML dynamically. It brings together several frameworks of evaluating indoor models and proposes a way to apply these on dynamic indoor models by evaluating not only the final model, but also the resulting models in between the start and end of scanning.

8.1.2 Limitations

The method is based on simulating a scanning process. Shortcomings involved in the collection of an indoor point cloud by a mobile device, such as drift, cannot be accounted for in the method by this simulation.

Much of the method built for this study focuses on the sample taken from the dataset. This creates some limitations that make it difficult to apply this method to other datasets whose characteristics differ from those of the sample dataset. These limitations include:

1. The parameters for the detection of doorways are configured to the doorways of the sample dataset;
2. It assumes a Manhattan world system. Meaning that diagonal walls cannot be modeled;
3. As was shown in the last paragraph of 6.2.1, the indoor spaces used for this method needs to be oriented along the right axes. This was done as a shortcut and works for this dataset, but forms a big limitation for other datasets.

However, the method achieves its intended purpose, which is to demonstrate that it is possible to build an IndoorGML model dynamically. More sophisticated methods of reconstruction should be used in future works.

8.2 Future work

8.2.1 Evaluation

The evaluation framework discussed in this paper includes the work done by working group IV/5 of the ISPRS (Khoshelham et al., 2017). This method tests the geometric quality of the model. However, there are several criteria involved in an indoor spatial model that are not included in this thesis. Besides common criteria such as performance that is important in all data models, the service-based requirements of (Afyouni et al., 2012) can also be considered. Future work could apply the location-based services from the evaluation framework of this study. In addition to checking for geometric quality by using a reference, it would also be useful to make an inherent evaluation of the quality. Because in a real application, chances are there is no reference model available. Future work will have to strive for a method that creates a model that is topologically correct. That is, without gaps and overlaps between features.

8.2.2 Real life implementation

Although the method used with the sample data showed a proof of concept to dynamically build an IndoorGML, more sophisticated reconstruction methods exist in the literature. For

example, the method would need to be improved to also allow for non-Manhattan spaces. Future should improve the method to be able to dynamically build more complex datasets, or even to perform it in a real scanning situation.

9 References

Afyouni, I., Ray, C., & Claramunt, C. (2012). Spatial models for context-aware indoor navigation systems: A survey. In *Journal of Spatial Information Science* (Vol. 4, Issue 2012, pp. 85–123). University of Maine. https://doi.org/10.5311/JOSIS.2012.4.73

Alattas, A., Oosterom, P. van, Zlatanova, S., Diakité, A. A., & Yan, J. (2018). Developing a database for the LADM-IndoorGML model. *Proceedings of the 6th International FIG 3D Cadastre Workshop*.

Alattas, A., van Oosterom, P., Zlatanova, S., Hoeneveld, D., & Verbree, E. (2020). LADM-IndoorGML for exploring user movements in evacuation exercise. *Land Use Policy*, *98*. https://doi.org/10.1016/j.landusepol.2019.104219

Apple, & Open Geospatial Consortium. (2021). *Indoor Mapping Data Format, version 1.0.0 (20-094)*. https://docs.ogc.org/cs/20-094/index.html

Becker, S., Peter, M., & Fritsch, D. (2015). Grammar-supported 3D indoor reconstruction from point clouds for "as-Built" BIM. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *2*(3W4), 17–24. https://doi.org/10.5194/isprsannals-II-3-W4-17-2015

Diakité, A. A., & Zlatanova, S. (2018). Spatial subdivision of complex indoor environments for 3D indoor navigation. *International Journal of Geographical Information Science*, *32*(2), 213–235. https://doi.org/10.1080/13658816.2017.1376066

Diakité, A. A., Zlatanova, S., Alattas, A. F. M., & Li, K. J. (2020). Towards indoorgml 2.0: Updates and case study illustrations. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *43*(B4), 337–344. https://doi.org/10.5194/isprs-archives-XLIII-B4-2020-337-2020

Diakité, A., Díaz-Vilariño, L., Biljecki, F., Zlatanova, S., Li, K.-J., Isikdag, ümit, & Simmons, S. (2021). *IFC to IndoorGML: pushing forward the development of software tools for IndoorGML*.

Dilo, A., & Zlatanova, S. (2011). A data model for operational and situational information in emergency response. *Applied Geomatics*, *3*(4), 207–218. https://doi.org/10.1007/s12518-011-0060-2

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

Endsley, M. R. (1988). Design and Evaluation for Situation Awareness Enhancement. *Proceedings of the Human Factors Society Annual Meeting*, *32*(2), 97–101. https://doi.org/10.1177/154193128803200221

Hagen-Zanker, A. (2016). A computational framework for generalized moving windows and its application to landscape pattern analysis. *International Journal of Applied Earth Observation and Geoinformation*, *44*, 205–216. https://doi.org/10.1016/j.jag.2015.09.010

Kang, H.-K., & Li, K.-J. (2017). A Standard Indoor Spatial Data Model—OGC IndoorGML and Implementation Approaches. *ISPRS International Journal of Geo-Information*, *6*(4), 116. https://doi.org/10.3390/ijgi6040116

Kang, Z., Yang, J., Yang, Z., & Cheng, S. (2020). A Review of Techniques for 3D Reconstruction of Indoor Environments. *ISPRS International Journal of Geo-Information*, *9*(5), 330. https://doi.org/10.3390/ijgi9050330

Karam, S., Lehtola, V., & Vosselman, G. (2021). Simple loop closing for continuous 6DOF LIDAR&IMU graph SLAM with planar features for indoor environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, *181*, 413–426. https://doi.org/10.1016/j.isprsjprs.2021.09.020

Khan, A. A., Yao, Z., & Kolbe, T. H. (2015). *Context Aware Indoor Route Planning Using Semantic 3D Building Models with Cloud Computing* (pp. 175–192). https://doi.org/10.1007/978-3-319-12181-9_11

Khan, S. H., Bennamoun, M., Sohel, F., & Togneri, R. (2014). Geometry Driven Semantic Labeling of Indoor Scenes. *European Conference on Computer Vision*.

Khoshelham, K., Díaz Vilariño, L., Peter, M., Kang, Z., & Acharya, D. (2017). THE ISPRS BENCHMARK ON INDOOR MODELLING. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-2/W7*, 367–372. https://doi.org/10.5194/isprs-archives-XLII-2-W7-367-2017

Khoshelham, K., Tran, H., Acharya, D., Vilariño, L. D., Kang, Z., & Dalyot, S. (2021). Results of the ISPRS benchmark on indoor modelling. *ISPRS Open Journal of Photogrammetry and Remote Sensing*, *2*, 100008. https://doi.org/10.1016/j.ophoto.2021.100008

Khoshelham, K., Tran, H., Díaz-Vilariño, L., Peter, M., Kang, Z., & Acharya, D. (2018). AN EVALUATION FRAMEWORK FOR BENCHMARKING INDOOR MODELLING METHODS. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII–4*, 297–302. https://doi.org/10.5194/isprs-archives-XLII-4-297-2018

Lehtola, V. V., Nikoohemat, S., & Nüchter, A. (2021). Indoor 3D: Overview on Scanning and Reconstruction Methods. In *Handbook of Big Geospatial Data* (pp. 55–97). Springer International Publishing. https://doi.org/10.1007/978-3-030-55462-0_3

Li, K.-J. (2008). *Indoor Space: A New Notion of Space* (pp. 1–3). https://doi.org/10.1007/978-3-540-89903-7_1

Li, K.-J., Hintz, D., Doh, N., Kalantari, M., Kim, S.-H., Benson, J., De Lathouwer, B., & Serich, S. (n.d.). *From Point Cloud to IndoorGML*. https://www.danielgm.net/cc/

Munkres, J. R. (1984). *Elements of Algebraic Topology* (1st ed.). CRC Press.

Nikoohemat, S., Diakité, A. A., Zlatanova, S., & Vosselman, G. (2020). Indoor 3D reconstruction from point clouds for optimal routing in complex buildings to support disaster management. *Automation in Construction*, *113*. https://doi.org/10.1016/j.autcon.2020.103109

Nikoohemat, S., Peter, M., Elberink, S. O., & Vosselman, G. (2018). Semantic interpretation of mobile laser scanner point clouds in Indoor Scenes using trajectories. *Remote Sensing*, *10*(11). https://doi.org/10.3390/rs10111754

Retscher, G. (2022). Indoor Navigation—User Requirements, State-of-the-Art and Developments for Smartphone Localization. *Geomatics*, *3*(1), 1–46. https://doi.org/10.3390/geomatics3010001

Rueppel, U., & Stuebbe, K. M. (2008). BIM-Based Indoor-Emergency-Navigation-System for Complex Buildings. *Tsinghua Science & Technology*, *13*(SUPPL. 1), 362–367. https://doi.org/10.1016/S1007-0214(08)70175-5

Shi, W., Ahmed, W., Li, N., Fan, W., Xiang, H., & Wang, M. (2018). Semantic Geometric Modelling of Unstructured Indoor Point Cloud. *ISPRS International Journal of Geo-Information*, *8*(1), 9. https://doi.org/10.3390/ijgi8010009

Sithole, G., & Zlatanova, S. (2016). POSITION, LOCATION, PLACE AND AREA: AN INDOOR PERSPECTIVE. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *III–4*, 89–96. https://doi.org/10.5194/isprs-annals-III-4-89-2016

Smit, B. P., Voûte, R., & Verbree, E. (2021). CREATING 3D INDOOR FIRST RESPONDER SITUATION AWARENESS in REAL-TIME through A HEAD-MOUNTED AR DEVICE. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *5*(4), 209–216. https://doi.org/10.5194/isprs-annals-V-4-2021-209-2021

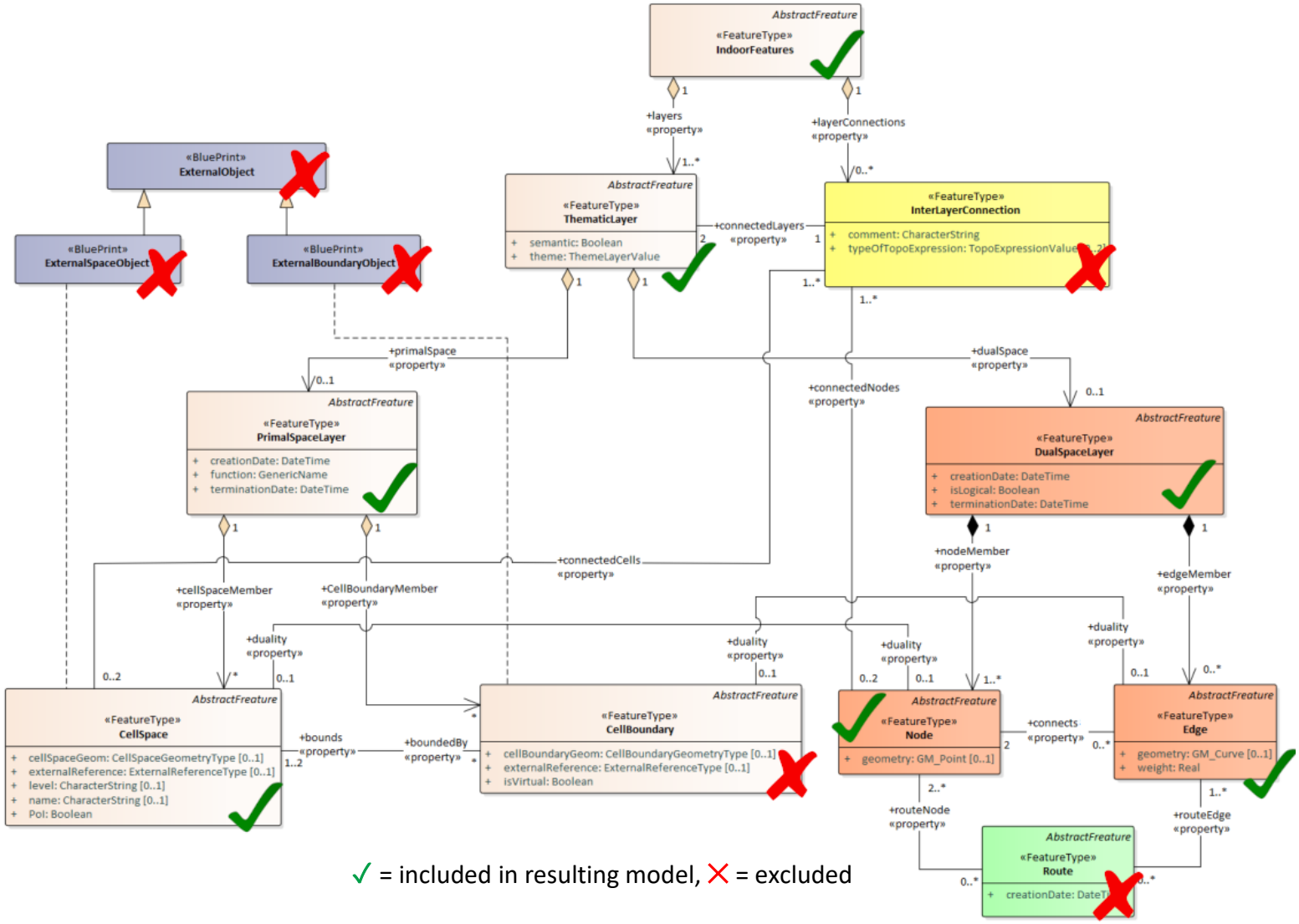
Staats, B. R., Diakité, A. A., Voûte, R. L., & Zlatanova, S. (2019). Detection of doors in a voxel model, derived from a point cloud and its scanner trajectory, to improve the segmentation of the walkable space. *International Journal of Urban Sciences*, *23*(3), 369–390. https://doi.org/10.1080/12265934.2018.1553685

Tran, H., Khoshelham, K., & Kealy, A. (2019). Geometric comparison and quality evaluation of 3D models of indoor environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, *149*, 29–39. https://doi.org/10.1016/J.ISPRSJPRS.2019.01.012

Zhang, J., & Singh, S. (2017). Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, *41*(2), 401–416. https://doi.org/10.1007/s10514-016-9548-2

Zhu, Z., Xu, Z., Chen, R., Wang, T., Wang, C., Yan, C., & Xu, F. (2022). FastFusion: Real-Time Indoor Scene Reconstruction with Fast Sensor Motion. *Remote Sensing*, *14*(15), 3551. https://doi.org/10.3390/rs14153551

10 Appendix A: IndoorGML Core module UML and table with features

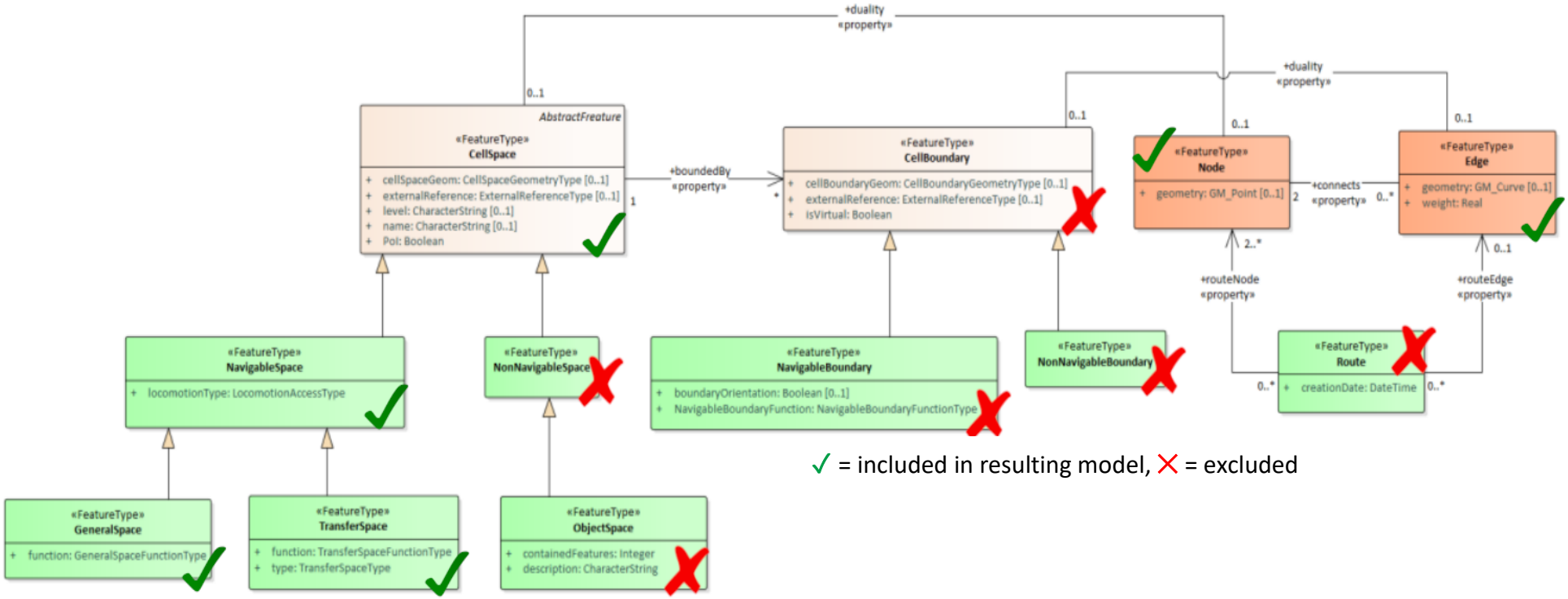


✓ = included in resulting model, ✗ = excluded

Element name	Included/excluded from research
IndoorFeatures	Included
ThematicLayer	Included
InterLayerConnection	Excluded
PrimalSpaceLayer	Included
DualSpaceLayer	Included
ExternalObject	Excluded
ExternalSpaceObject	Excluded
ExternalBoundaryObject	Excluded
CellSpace	Included
CellBoundary	Excluded
Node	Included
Edge	Included
Route	Excluded

Table 10.1 – Elements of the IndoorGML Core module and whether they are included or excluded

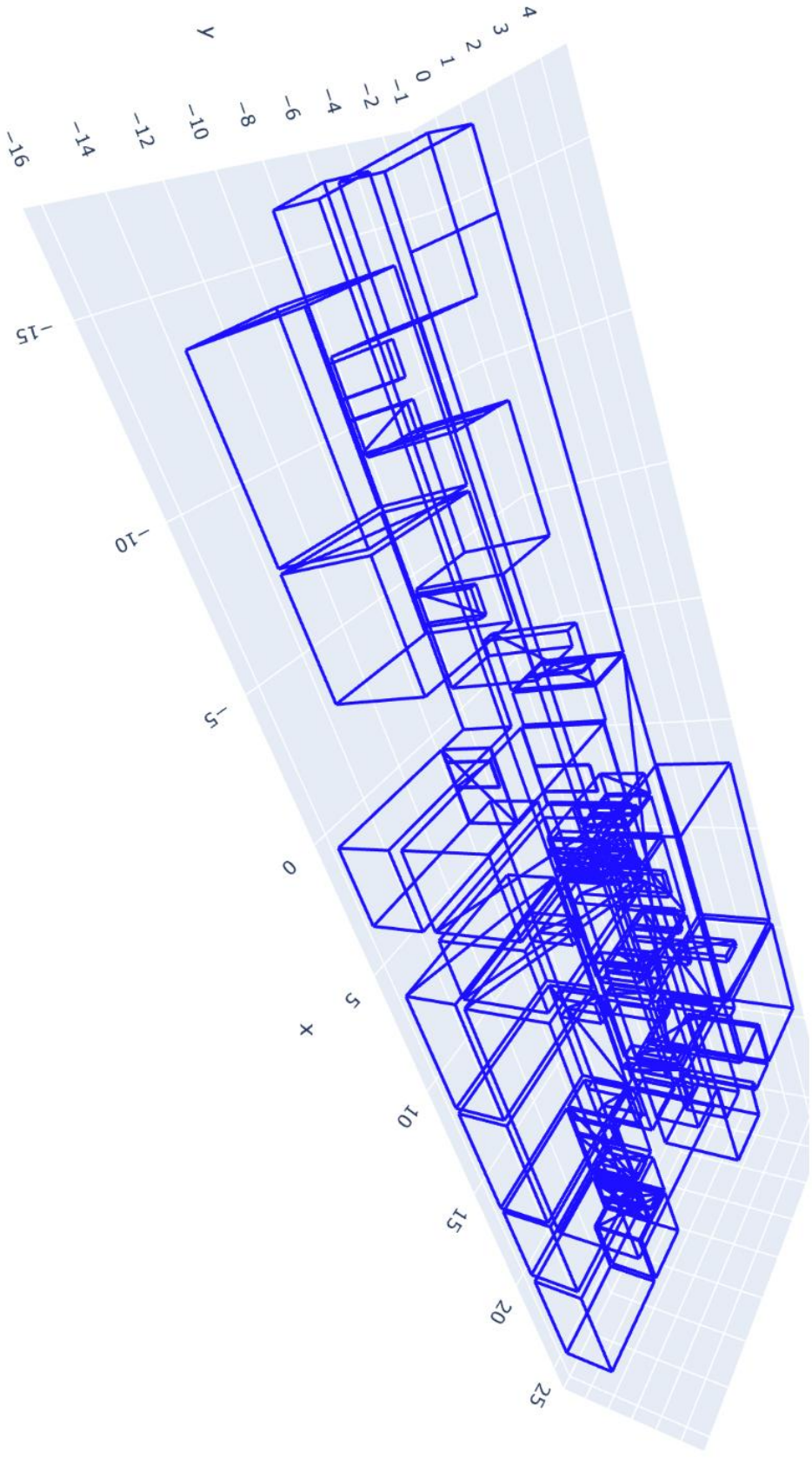
11 Appendix B: Navigation Extension module UML and table with features

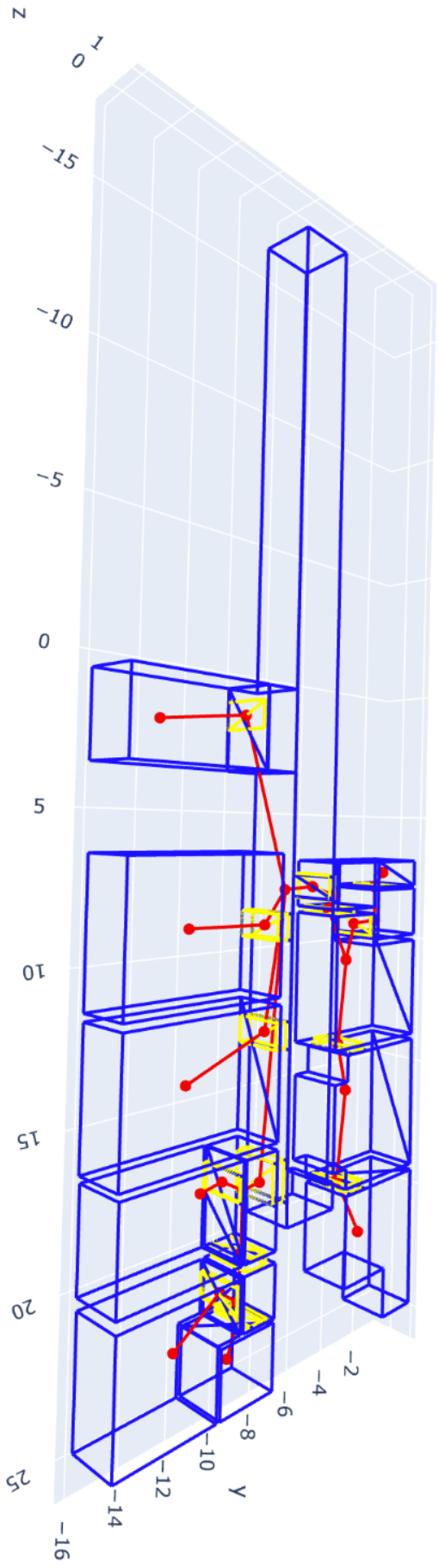


Element name	Included/excluded from research
CellSpace	Included
CellBoundary	Excluded
NavigableSpace	Included
GeneralSpace	Included
TransferSpace	Included
NonNavigableSpace	Excluded
ObjectSpace	Excluded
NavigableBoundary	Excluded
NonNavigableBoundary	Excluded
Node	Included
Edge	Included
Route	Excluded

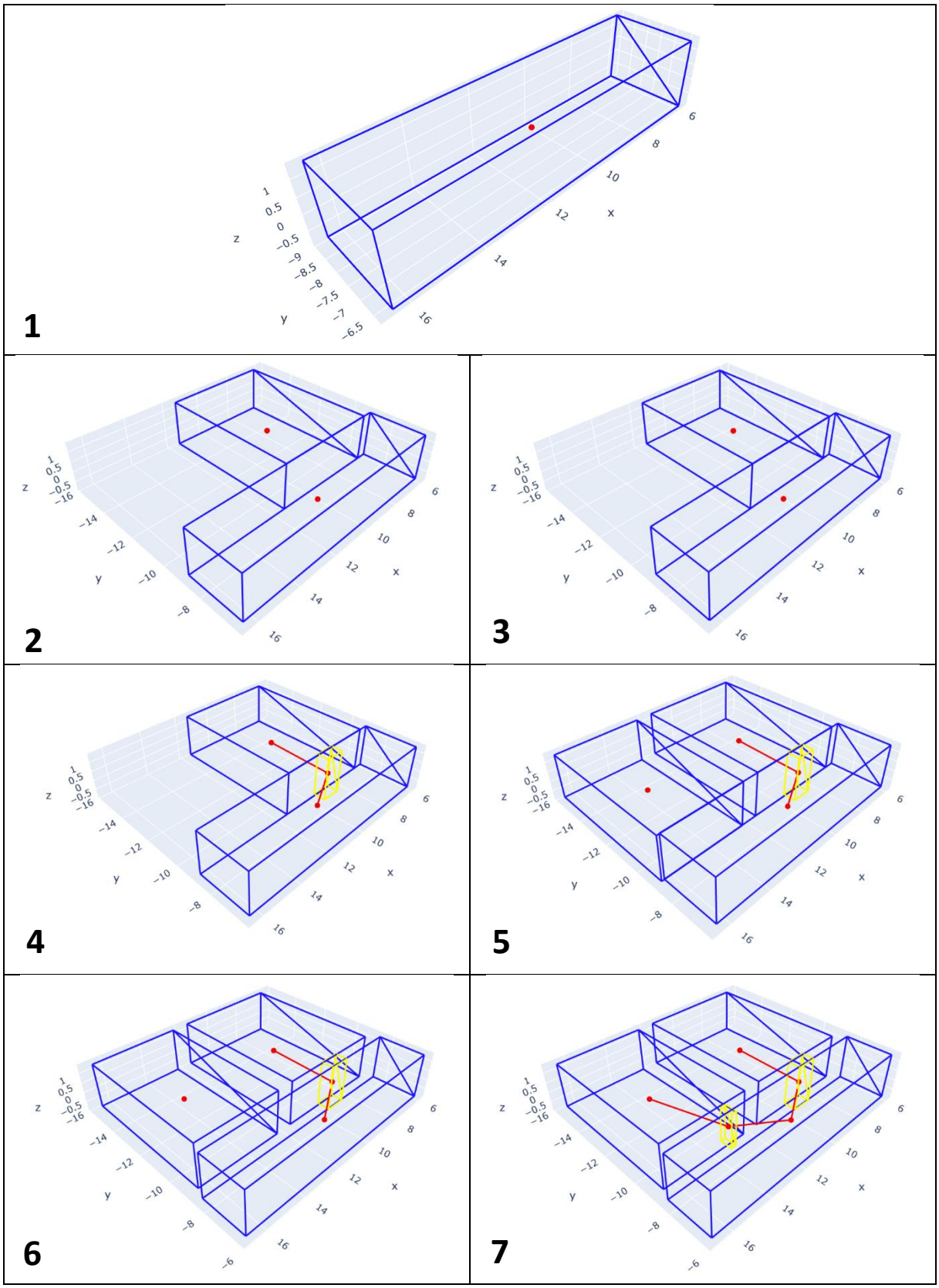
Table 11.1 – Elements of the IndoorGML Navigation Extension module and whether they are included or excluded

12 Appendix C: Visualizations of the reference model




z
1
0
−15
−10
−5
0
5
10
15
20
25
−2
−4
−6
−8
−10
−12
−14
−16
y

13 Appendix D: Step-wise visualization of the dynamic model



14 Appendix E: IndoorGML of dynamic model

```
<?xml version="1.0" encoding="utf-8"?>
<core:IndoorFeatures gml:id="InFt_1"
xmlns:core="http://www.opengis.net/indoorgml/1.0/core"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:navi="http://www.opengis.net/indoorgml/1.0/navigation"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/indoorgml/1.0/core
C:\thesis_env\xsd\indoorgml_2_0_base_module.xsd
http://www.opengis.net/indoorgml/1.0/navigation
C:\thesis_env\xsd\indoorgml_2_0_navigation_module.xsd">
    <core:ThematicLayer>
        <semantic>
            true
        </semantic>
        <Theme>
            Physical
        </Theme>
        <core:PrimalSpaceLayer gml:id="PSL_1">
            <navi:GeneralSpace gml:id="CS_0">
            </navi:GeneralSpace>
            <navi:GeneralSpace gml:id="CS_1">
            </navi:GeneralSpace>
            <navi:TransferSpace gml:id="CS_10">
            </navi:TransferSpace>
            <navi:GeneralSpace gml:id="CS_2">
            </navi:GeneralSpace>
            <navi:TransferSpace gml:id="CS_11">
            </navi:TransferSpace>
        </core:PrimalSpaceLayer>
        <core:DualSpaceLayer gml:id="DSL_1">
            <core:Node gml:id="N_0">
            </core:Node>
            <core:Node gml:id="N_1">
            </core:Node>
            <core:Node gml:id="N_10">
            </core:Node>
            <core:Edge gml:id="E-CS_10-CS_0">
            </core:Edge>
            <core:Edge gml:id="E-CS_10-CS_1">
            </core:Edge>
            <core:Node gml:id="N_2">
            </core:Node>
            <core:Node gml:id="N_11">
            </core:Node>
            <core:Edge gml:id="E-CS_11-CS_0">
            </core:Edge>
            <core:Edge gml:id="E-CS_11-CS_2">
            </core:Edge>
        </core:DualSpaceLayer>
    </core:ThematicLayer>
</core:IndoorFeatures>
```

15 Appendix F: IndoorGML of GeneralSpace

```
<navi:GeneralSpace gml:id="CS_0">
    <core:cellSpaceGeometry>
        groundfloor
        <core:Geometry3D>
            <gml:Solid gml:id="GC-CS_0">
                <gml:exterior>
                    <gml:Shell>
                        <gml:surfaceMember>
                            <gml:Polygon>
                                <gml:exterior>
                                    <gml:LinearRing>
                                        <gml:pos srsDimension="3">
                                            5.90381029129 -8.519130364895
                                            -0.8748379001617518
                                        </gml:pos>
                                        <gml:pos srsDimension="3">
                                            16.6949408741 -8.519130364895
                                            -0.8748379001617518
                                        </gml:pos>
                                        <gml:pos srsDimension="3">
                                            16.6949408741 -6.274986969948
                                            -0.8748379001617518
                                        </gml:pos>
                                        <gml:pos srsDimension="3">
                                            5.90381029129 -6.274986969948
                                            -0.8748379001617518
                                        </gml:pos>
                                        <gml:pos srsDimension="3">
                                            5.90381029129 -8.519130364895
                                            -0.8748379001617518
                                        </gml:pos>
                                    </gml:LinearRing>
                                </gml:exterior>
                            </gml:Polygon>
                        </gml:surfaceMember>
                        <gml:surfaceMember>
                            <gml:Polygon>
                                <gml:exterior>
                                    <gml:LinearRing>
                                        <gml:pos srsDimension="3">
                                            5.90381029129 -8.519130364895
                                            -0.8748379001617518
                                        </gml:pos>
                                        <gml:pos srsDimension="3">
                                            16.6949408741 -8.519130364895
                                            -0.8748379001617518
                                        </gml:pos>
                                        <gml:pos srsDimension="3">
                                            16.6949408741 -8.519130364895
                                            1.3639314014911852
                                        </gml:pos>
                                        <gml:pos srsDimension="3">
                                            5.90381029129 -8.519130364895
                                            1.3639314014911852
                                        </gml:pos>
```

```
                    <gml:pos srsDimension="3">
                        5.90381029129 -8.519130364895
                        -0.8748379001617518
                    </gml:pos>
                </gml:LinearRing>
            </gml:exterior>
        </gml:Polygon>
    </gml:surfaceMember>
    <gml:surfaceMember>
        <gml:Polygon>
            <gml:exterior>
                <gml:LinearRing>
                    <gml:pos srsDimension="3">
                        16.6949408741 -8.519130364895
                        -0.8748379001617518
                    </gml:pos>
                    <gml:pos srsDimension="3">
                        16.6949408741 -6.274986969948
                        -0.8748379001617518
                    </gml:pos>
                    <gml:pos srsDimension="3">
                        16.6949408741 -6.274986969948
                        1.3639314014911852
                    </gml:pos>
                    <gml:pos srsDimension="3">
                        16.6949408741 -8.519130364895
                        1.3639314014911852
                    </gml:pos>
                    <gml:pos srsDimension="3">
                        16.6949408741 -8.519130364895
                        -0.8748379001617518
                    </gml:pos>
                </gml:LinearRing>
            </gml:exterior>
        </gml:Polygon>
    </gml:surfaceMember>
    <gml:surfaceMember>
        <gml:Polygon>
            <gml:exterior>
                <gml:LinearRing>
                    <gml:pos srsDimension="3">
                        16.6949408741 -6.274986969948
                        -0.8748379001617518
                    </gml:pos>
                    <gml:pos srsDimension="3">
                        5.90381029129 -6.274986969948
                        -0.8748379001617518
                    </gml:pos>
                    <gml:pos srsDimension="3">
                        5.90381029129 -6.274986969948
                        1.3639314014911852
                    </gml:pos>
                    <gml:pos srsDimension="3">
                        16.6949408741 -6.274986969948
                        1.3639314014911852
                    </gml:pos>
                    <gml:pos srsDimension="3">
```

```
                    16.6949408741 -6.274986969948
                    -0.8748379001617518
                </gml:pos>
            </gml:LinearRing>
        </gml:exterior>
    </gml:Polygon>
</gml:surfaceMember>
<gml:surfaceMember>
    <gml:Polygon>
        <gml:exterior>
            <gml:LinearRing>
                <gml:pos srsDimension="3">
                    5.90381029129 -6.274986969948
                    -0.8748379001617518
                </gml:pos>
                <gml:pos srsDimension="3">
                    5.90381029129 -8.519130364895
                    -0.8748379001617518
                </gml:pos>
                <gml:pos srsDimension="3">
                    5.90381029129 -8.519130364895
                    1.3639314014911852
                </gml:pos>
                <gml:pos srsDimension="3">
                    5.90381029129 -6.274986969948
                    1.3639314014911852
                </gml:pos>
                <gml:pos srsDimension="3">
                    5.90381029129 -6.274986969948
                    -0.8748379001617518
                </gml:pos>
            </gml:LinearRing>
        </gml:exterior>
    </gml:Polygon>
</gml:surfaceMember>
<gml:surfaceMember>
    <gml:Polygon>
        <gml:exterior>
            <gml:LinearRing>
                <gml:pos srsDimension="3">
                    5.90381029129 -8.519130364895
                    1.3639314014911852
                </gml:pos>
                <gml:pos srsDimension="3">
                    16.6949408741 -8.519130364895
                    1.3639314014911852
                </gml:pos>
                <gml:pos srsDimension="3">
                    16.6949408741 -6.274986969948
                    1.3639314014911852
                </gml:pos>
                <gml:pos srsDimension="3">
                    5.90381029129 -6.274986969948
                    1.3639314014911852
                </gml:pos>
                <gml:pos srsDimension="3">
                    5.90381029129 -8.519130364895
```

```
                                                    1.3639314014911852
                                                </gml:pos>
                                            </gml:LinearRing>
                                        </gml:exterior>
                                    </gml:Polygon>
                                </gml:surfaceMember>
                            </gml:Shell>
                        </gml:exterior>
                    </gml:Solid>
                </core:Geometry3D>
            </core:cellSpaceGeometry>
        </navi:GeneralSpace>
```

16 Appendix G: IndoorGML of Node and Edge

```
<core:Node gml:id="N_0">
    <core:duality xlink:href="CS_0"/>
    <core:geometry>
        <gml:Point>
            <gml:pos srsDimension="3">
                11.299375582695 -7.397058667421501 0.2445467506647167
            </gml:pos>
        </gml:Point>
    </core:geometry>
</core:Node>



<core:Edge gml:id="E-CS_10-CS_0">
    <core:connects xlink:href="N_10"/>
    <core:connects xlink:href="N_0"/>
    <core:geometry>
        <gml:LineString>
            <gml:pos srsDimension="3">
                11.299375582695 -7.397058667421501 0.2445467506647167
            </gml:pos>
            <gml:pos srsDimension="3">
                9.0240556612015 -8.766625227213 0.2618407500982253
            </gml:pos>
        </gml:LineString>
    </core:geometry>
</core:Edge>
```