UNIVERSITÀ DEGLI STUDI DELL'AQUILA

DIPARTIMENTO DI INGEGNERIA E SCIENZE DELL'INFORMAZIONE E MATEMATICA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA – AUTOMATICA

AUTOMATIC EXTRACTION OF IMPROVED GEOMETRICAL NETWORK MODEL FROM CITYGML FOR INDOOR NAVIGATION

BY

FILIPPO MORTARI

(MAT. 204129)

ANNO ACCADEMICO 2012 – 2013

RELATORE	Dr. Dipl.-Ing. Eliseo Clementini	UNIVAQ
CORRELATORE	Dr. Dipl.-Ing. Sisi Zlatanova	TUDELFT, THE NETHERLANDS

IN COLLABORATION WITH

TU Delft Delft University of Technology

Filippo Mortari: *Automatic Extraction of Improved Geometrical Network Model from CityGML for Indoor Navigation,* A thesis submitted to University of L'Aquila in partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Automation Engineering , © December 2013

Dedicated to my family.

ABSTRACT

Over the past few years Personal Navigation Systems have become an established tool for route planning, but they are mainly designed for outdoor environments. Indoor navigation is still a challenging research area for several reasons: positioning is not very accurate, users can freely move between the interior boundaries of buildings, path network construction process may not be easy and straightforward due to complexity of indoor space configurations. Public buildings are getting bigger and more complex, so it is likely that even people who are familiar with these environments could possibly find difficult to reach some specific room or place. The creation of a good network and data model is essential for deriving overall connectivity of a building and for representing position of objects within the environment. Thus, the purpose of this research is the automatic derivation of the connectivity graph of a building for indoor navigation since from current literature it has not been found any approach that leads to automatic derivation of 2D and 3D Geometric Network models out of 3D digital models of complex buildings like CityGML or IFC.

ABSTRACT (ITALIAN)

Nel corso degli ultimi anni i Sistemi di Navigazione Personale (PNS), sono divenuti uno strumento essenziale per la pianificazione di itinerari, purtuttavia essi sono sviluppati principalmente per ambienti outdoor. La navigazione indoor resta ancora un'area di ricerca stimolante per una serie di motivazioni: il posizionamento non è molto accurato, gli utenti possono muoversi liberamente entro il perimetro degli edifici, la determinazione di un percorso e la conseguente generazione di un network potrebbero rivelarsi di difficile realizzazione a causa della complessità delle configurazioni degli spazi indoor. Gli edifici pubblici di giorno in giorno crescono in complessità nonchè in dimensione, tanto che perfino coloro che hanno dimestichezza con questi ambienti potrebbero avere difficoltà nell'individuare il percorso necessario per raggiungere un determinato luogo o una determinata stanza. Pertanto, la creazione di un buon network e di un data model risulta essenziale per derivare la connettività completa di un edificio e per rappresentare la posizione di oggetti o soggetti all'interno dello stesso. Lo scopo di questo lavoro di tesi consiste, quindi, nella derivazione automatica di un grafo di connettività di un edificio come soluzione al problema della navigazione indoor, dal momento che allo stato dell'arte non è stato individuato un approccio che consenta di generare automaticamente modelli di network 2D / 3D da modelli informativi digitali di edifici 3D come IFC o CityGML.

"The wisest men follow their own direction."

— Euripides

ACKNOWLEDGEMENTS

This research project is the result of an Erasmus Placement Internship Programme conducted at OTB Institute for the Build Environment - TUDelft (http://www.otb.tudelft.nl/). Foremost, I gratefully acknowledge the Heads of GISTechnology section Peter Van Oosterom and Sisi Zlatanova for having accepted me as a guest researcher. Thank you, for having given me the opportunity to work in your wonderful research group and for allowing me enough freedom to develop and pursue my ideas. Peter, the conversations with you during coffee breaks have always been constructive and genuine, food for thought.

I would like to express my sincere gratitude to my supervisor in TUDelft, Prof. Sisi Zlatanova whose expertise, understanding, motivation, enthusiasm, immense knowledge and patience, added considerably to my graduate experience. Her guidance helped me in all the time of research and made me feel always motivated, also in the darkest moments.

I am indebted to my many of my GISTechnology colleagues to support me in OTB: Ken Arroyo Ohori for the weekly meetings and his very useful advice, Liu Liu for the conversations which helped me out in steering my research to the right direction, Junqiao (John) Zhao for his comments and feedback, Elfriede Fendel for the administrative and technical staff, Zhiyong Wang for being a wonderful office roommate.

I also owe a great debt of gratitude to Prof. Eliseo Clementini, my italian graduation professor, for his unconditional support, his precious advice and his guidance through the days. He was the one who made me develop a passion for Geographic Information Systems and Computational Geometrty thanks to his authenticity and his devotion to duty.

None of this work would have been possible without the love and patience of my family, to whom this dissertation is dedicated to. Thank you for having been a constant source of love, concern, support and strenght through my entire life.

Special thanks to the friends of a lifetime that have been present, in the good and the bad moments: Daniela, who supported me in everything, Matteo (Maβacesi), Chiara, Fabrizio (Brozio), Matteo (ju Dottò), Alessio, Giada, Luigi (Gasby), Angelica, Federico, Roberto, Francesco, Jacopo, Gian Luca, Elisabetta, Luigi (Tiberi), Elisa, Simone, Antonio, Antonello.

I cannot skip the University friends, in the past few years I spent more time with you rather than with my parents: Luca (JPanel), Davide, Alessio, Luca (ju Toro), Maurizio, Piero, Daniele, Nicola, Emilio, Antonio.

Yet thanks to the wonderful people I had the opportunity to meet in Delft during my internship: I am grateful to Valentina, for having been at my side when I needed it more, then Stefano, Irene, Eugenio, Carlo, Greg, Aldo, Fabio, David, Dena, Christa, Dorina, Leon, Elias, Sina, Bardia, Pirouz.

A final thank to all the people I encountered in these years that contributed to my personal growth. You are too many to fit in such a small space, but all your names reside in my heart.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

BIM	Building Information Modeling
CAD	Computer Aided Design
CGAL	Computational Geometry Algorithms Library
CDT	Constrained Delaunay Triangulation
CityGML	City Geography Markup Language
GIS	Geographic Information Systems
GML	Geography Markup Language
GNM	Geometric Network Model
GPS	Global Positioning System
IFC	Industry Foundation Classes
IGNM	Improved Geometric Network Model
ISO	International Organization for Standardization
LOD	Level of Detail
MAT	Medial Axis Transformation
NRS	Node Relation Structure
OGC	Open Geospatial Consortium
OSM	OpenStreetMap
S-MAT	Straight - Medial Axis Transformation
UML	Unified Modeling Language
VG	Visibility Graph

XML	Extensible Markup Language

1 INTRODUCTION

Over the past few years Personal Navigation Systems have become an established tool for route planning, but they are mainly designed for outdoor environments. A motivation to this can be deducted by some evident factors: in outdoor environments the positioning problem has been easily fixed thanks to global positioning techniques like GPS and also the acquisition of huge amount of datasets for road network generation has been possible using remote sensing and photogrammetry techniques.

Indoor navigation is still a challenging research area for several reasons: positioning is not very accurate, users can freely move between the interior boundaries of buildings, path network construction process may not be easy and straightforward due to complexity of indoor space configurations. Public buildings like shopping malls, airports, concert halls are getting bigger and more complex, so it is likely that even people who are familiar with these environments could possibly find it difficult to reach some specific room or place.

More generally speaking, a navigation system comprises 1) the determination of the position of a user, 2) the calculation of a best path to some destination (presumably the shortest one, the cheapest one, the fastest one), 3) guidance along the path. Since best path algorithms quite often refer to graph theory, the creation of a good network and data model is essential for deriving the overall connectivity of a building and for representing position of objects within the environment. Thus, the purpose of this research is the automatic derivation of the connectivity graph of a building from 3D building models like CityGML.

As shown in [1], regarding indoor modelling approaches, two main classes are inferred: symbolic and geometric spatial models. The last ones model space as continuous or discrete and basically comprise cell-based or boundary-based representations and they rely on metrics and angles (*"the distance from place x to place y is z meters"*), while the first ones model space using topological relationships, graphs by capturing the connectivity and reachability between spatial units (*"the place x is adjacent to place y"*). Examples of geometric spatial models could be regular tessellations (see Figure 1), irregular tessellations based on triangulations or trapezoidalizations or Voronoi Diagrams (see Figure 2). Examples of symbolic models could be Place-based sets or graph-based models. The first ones model an indoor space by creating sets and subsets of place identifiers based on architectural properties of space. A typical example considers places of a building such that each floor is contained within a building, and each room is contained within at most one floor. A superset is likely to be defined as the set of floor numbers, and at a lower level, a subset related to each floor that aggregates all room numbers has to be created [1]. Graph-based models are the most commonly used approaches in several contexts and application areas ranging from emergency management and safety control in micro-scale environments, to indoor navigation services [1]. They are very appreciated because they can lead to the distinction between so called "primal space" (geometrical space in $\mathbb{R}^3$) and "dual space" (logical network), in order to simplify complex spatial relationships between objects in 3D environment like spatial adjacency. Examples of 2D complex relationships have been documented in [12].

Lee [22] introduced a topological data model called Node Relation Structure which is a dual graph representing connectivity relationships between 3D objects. This representation relies on Poincaré Duality: 3D solids in primal space are mapped into vertices in dual space, and linking surfaces between solid objects in primal spaces

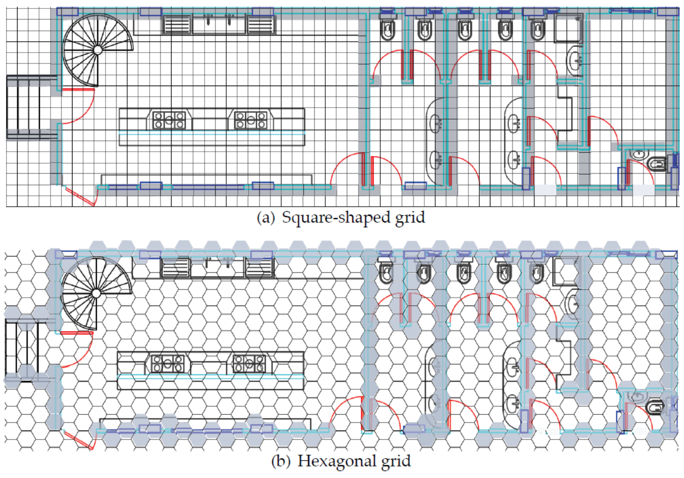


Figure 1: Regular tessellations (image taken from [1])

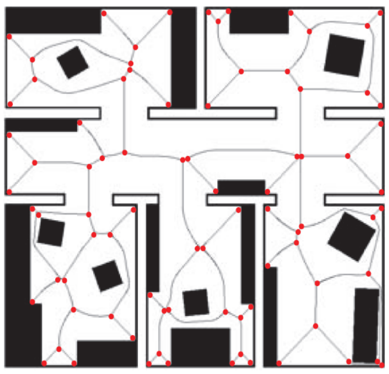


Figure 2: Generalized Voronoi Diagram of an indoor space (image taken from [1])

(e.g. doors) are mapped into connecting edges in dual space. Figure 3 displays the transformation.

Node Relation Structure is merely a topological representation, and so it does not involve metrics. Basically it consists in a graph – a logical network – in which nodes represent spaces and edges represent connectivity between those spaces. Since topology studies properties that are preserved under continuous deformations – like connectivity – geometrical properties like distance are ignored. Metrics are a crucial information for shortest path computations or more generally speaking path finding algorithms. Is it possible to combine these two

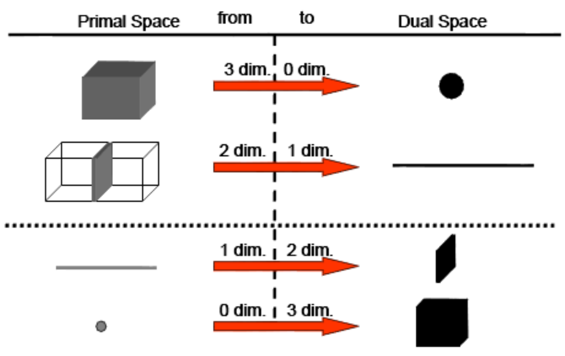


Figure 3: Lee's NRS based on Poincarè Duality

aspects in a model capable of providing both topological and metric information?

Lee extended NRS to the so called "Geometrical Network Model", which is a connectivity graph (topological model) plus geometric information such as coordinates of the nodes and link lengths for cost (the shortest/optimum path) computation. As Figure 4. displays, the geometry, adding metrics and angles to the connectivity provided by the topological representation, makes the "Geometric Network Model" a 3D model itself.

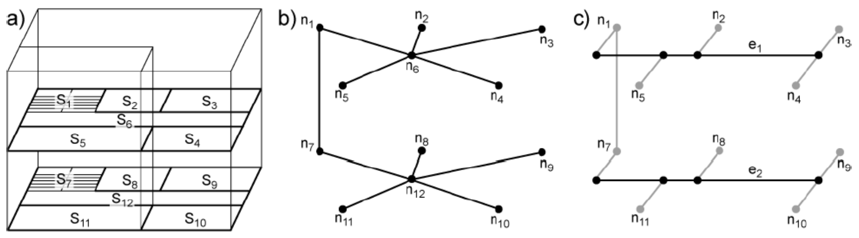


Figure 4: (a) 3D Building Model. (b) Topologic Model . (c) Geometric Network Model (Topology + Geometry)

The Figure 5 summarises Lee's structure of space that leads to a distinction from primal space to dual space on one hand, and from Euclidean space to Topological space on the other hand.

This representation characterized by the separation into two dual graphs is similar to the ISO standard 19107:2003 for describing the spatial characteristics of geographic features [14]

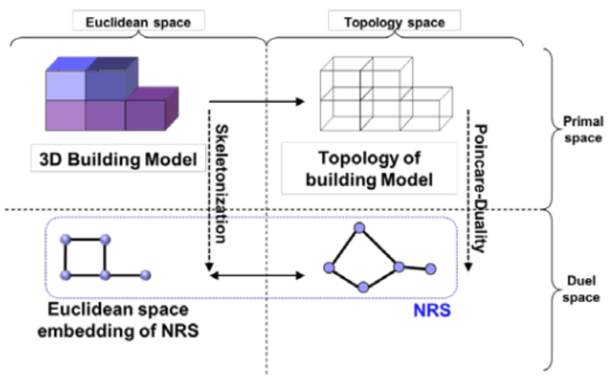


Figure 5: summary of Lee's approach

However, the majority of the indoor models ignore at least one of the following aspects: 1) they support only one type of locomotion. Some of the models are designed only for pedestrian, whereas some others only for wheelchair users. Supporting different types of locomotion means that different indoor space representations have to be generated. 2) They ignore architectural characteristics. Doors, openings, windows are not taken into consideration in several models. For instance, a room could have multiple doors, so different possible paths could be considered for navigation purposes. These kinds of architectural characteristics are very important for indoor navigation since they semantically enrich the data model. Knowing that a room has only one door implies that this room may be considered as an endpoint and not as a transfer space. If the same room has a window that window could be used for emergency response and evacuation of the premises. 3) Granularity is too coarse. Most of the models abstract a room as a single node within the network graph, not considering a detailed partition of a room space into different areas. For instance, it is possible to consider a large concert hall, or the main entrance area of an airport: this kind of spaces should be subdivided into smaller units in order to achieve geometrically/semantically richer models. Modelling these spaces only with a node 4) Obstacles are ignored. These kind of constraints for route planning most of the time are not taken into consideration, due to the fact that most of the frameworks rely on CAD / arbitrary-shaped floor plans as originating datasets.

This last sentence highlights another crucial aspect of the research: what is the input? Becker et al., in [4] claim that *"semantic building models for the representation of topographic 3D objects nowadays become increasingly available in the context of Building Information Modeling (BIM), such as the Industry Foundation Classes (IFC) and in the field of 3D city modeling (referring to CityGML, Ed.)"*. *"The 3D geometry representation of built environment in Euclidean space can be retrieved directly from IFC and CityGML"*. But from current literature it has not been found any approach that leads to automatic derivation of 2D and 3D Geometric Network models out of 3D digital models of complex buildings.

City Geography Markup Language (CityGML) has become an accepted information model throughout the 3D GIS community for its captivating features: *"CityGML is a common semantic information model for the representation of 3D urban objects that can be shared over different applications... CityGML defines the classes and relations for the most relevant topographic objects in cities and regional models with respect to their geometrical, topological, semantical, and appereance properties"* [13]. It supports multi-scale modeling, defining 5 Levels of Detail (see Figure 6), where LOD0 essentially a two and a half digital terrain model, LOD1 is the block model and LOD4 is the most detailed model which specifies architectural characteristics including interior. Therefore, CityGML seems to be a promising starting point for indoor navigation applications. Figure 7 depicts the semantics of CityGML in an indoor context.

1.1 RESEARCH OBJECTIVE

As highlighted in the introduction section, existing indoor navigation data models suffer from various problems. Nevertheless, the automatic derivation of connectivity network out of complex semanti-

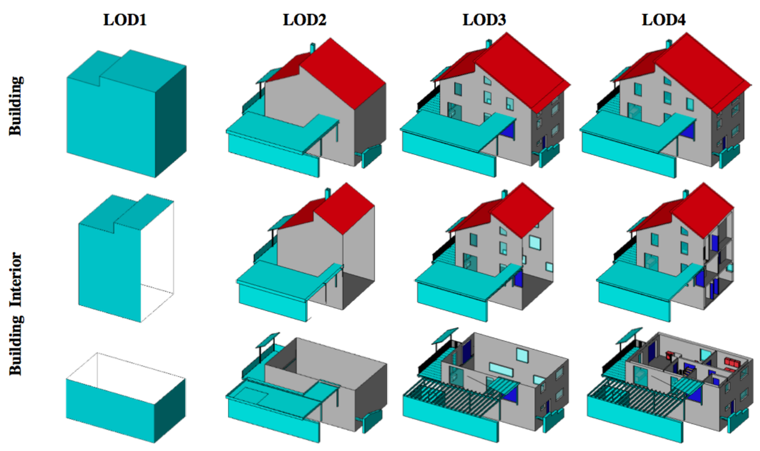


Figure 6: Different LODs in CityGML

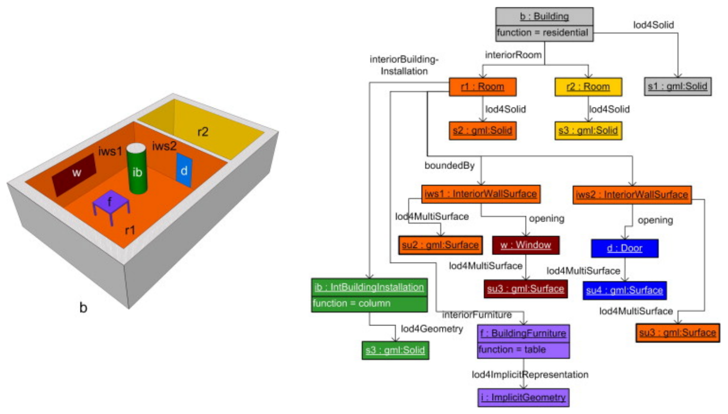


Figure 7: a graphical overview of the semantically-rich CityGML classes

cally rich 3D models like CityGML is still a challenging and not very straightforward task. Therefore, the research question of this thesis is:

Is it possible to automatically extract an Improved Geometrical Network Model from CityGML LOD4 for Indoor Navigation?

From this question several sub-questions have been derived:

- How indoor geometrical properties of buildings can be retrieved from CityGML LOD4 datasets?
- What is an Improved Geometrical Network Model?
  - In other words what are the improvements we want to perform in order to achieve a richer connectivity graph?

  - What are the semantic properties that can be extracted from CityGML that could be meaningful to this improvement?

- Is it possible to implement an automatic network extraction process?
  - What could be the geometrical operations needed to perform network extraction?

1.2 THESIS OUTLINE

FOLLOWING THIS INTRODUCTION The rest of this thesis proceeds as follows: Chapter 2 illustrates related work regarding the existing approaches to the problem of indoor space modelling. In Chapter 3 the methodological concepts that are used in this research and the workflow being followed are described. Chapter 4 displays how CityGML could be beneficial for Indoor Navigation applications, and methodology for extracting semantics, topology and geometry from LOD4 datasets is illustrated. In Chapter 5 a novel Data Model for Indoor Navigation is presented. This model has been designed as an extension of OGC's candidate standard named IndoorGML. In Chapter 6 the strategies for deriving automatic network extraction are discussed. In Chapter 7 some impementation details regarding both derivation of semantics, geometry, topology from CityGML and automatic derivation of Improved Geometric Network Model will be given. In Chapter 8 we will review the research contributions of this dissertation, as well as discuss directions for future research.

2 RELATED WORK

As already stated in the introduction section, there are several indoor models that have been found in current literature. In this Chapter we will make an overview of the existing approaches to the problem of indoor space modelling. In [20] is presented a model which decom-

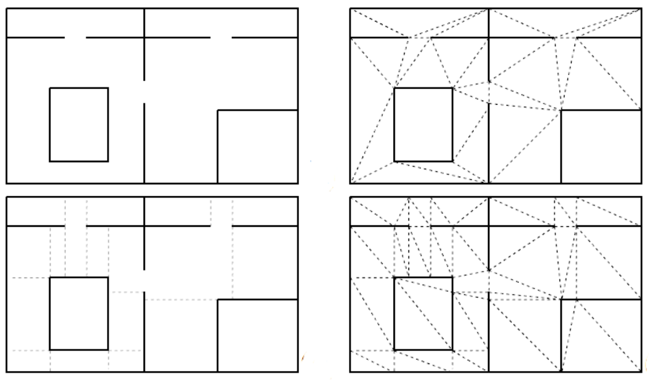


Figure 8: 2D plan-based subdivision (work from [20])

poses the space by using differently shaped cells. The subdivision technique relies on Constrained Delaunay Triangulation and Convex Cell optimisation detecting bottlenecks inside rooms (see Figure 8). A hierarchical path planning is proposed based on the abstraction and generalisation of topological properties extracted from the spatial subdivision. This approach anyway is limited to 2D floor plans and ignores semantics.

A hybrid spatial model for indoor environments which consist of hierarchically structured paths and optional semantic information is presented by Lorenz et al. in [28]. For buildings with simple rooms and corridors, indoor space (2D floor plan) is decomposed into cells to build a graph structure. They propose a direct mapping for small building instances like small rooms and corridors to the nodes in the

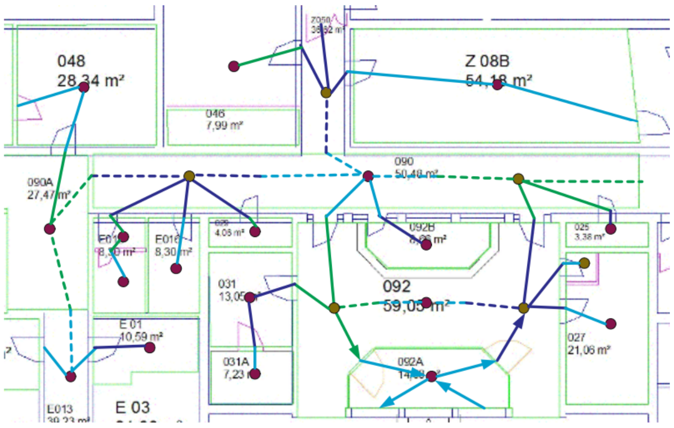


Figure 9: Cell centres and paths overlaid with a floor Plan (work from [28])

graph. The model is hybrid in the sense that nodes and edges can be labelled with qualitative as well as quantitative information. For more complex spaces they propose decomposing cells into several non-overlapping disjoint cells. The necessity of space decomposition is highlighted for several reasons: the sheer size, or the concavity of a room or the semantic functionality of certain areas in a larger open space. But *"unfortunately, there is no obvious way to fully automate the cell decomposition. It has to be designed very carefully, taking into account the purpose of different cells"*. The proposed approach in [28] has been

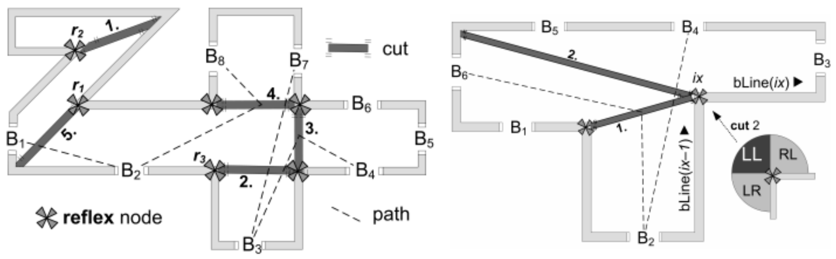


Figure 10: Space decomposition from work of [34])

extended by Stoffel et al. in [34]. This new method consists in partitioning a plan into non-overlapping convex sub-regions, so-called "cells", according to visibility criteria (see Figure 10). Stoffel mentioned the necessity of cellular decomposition also in [33], claiming that for practical navigation tasks the finest resolution of the hierarchy (of leaf regions) can still be too coarse, and subspacing of complex regions may be an essential task for navigation purpose. These regions according to the qualitative classification of spaces by Montello [29] are medium spaces: people can only experience them by locomotion and integration of multiple vantage points. Thus, Stoffel

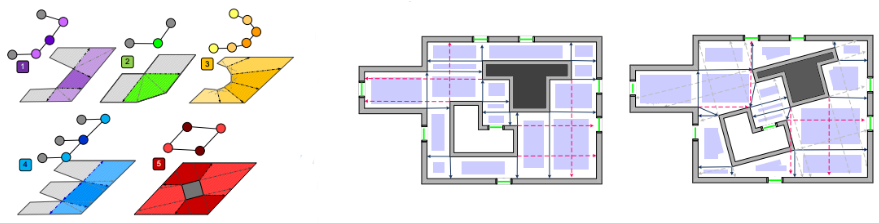


Figure 11: Decomposition of a complex region into convex parts according to the work in [33])

in [33] proposes a decomposition of space based on "Gestalt theory" considering the peculiarities of human perception of spatial objects and shapes, as shown in Figure 11.

The aforementioned approaches have in common the fact that they rely on irregular space partitioning, providing adaptive decomposition of the space that is suitable to exactly represent the complexity of the environment being studied. Another methodology consists in representing the available free space by regular tessellations (rectangles, hexagons, etc..). The union of the generated cells may not represent exactly the available space especially for the boundaries, but, on the other hand the model is quite simple to implement and could be more efficient for path planning. In [24] a node-edge grid graph is used to represent a 2D indoor space (2D floor plan), in which nodes do not represent rooms or vista spaces but cells within an occupancy grid, and connections between cells are materialized by edges. First, an indoor space is separated into cellular units, which are defined and identified based on the floor plan of the space. Second, these cellular units are covered by a grid graph, where nodes and edges are labelled with membership values according to the underlying cellu-

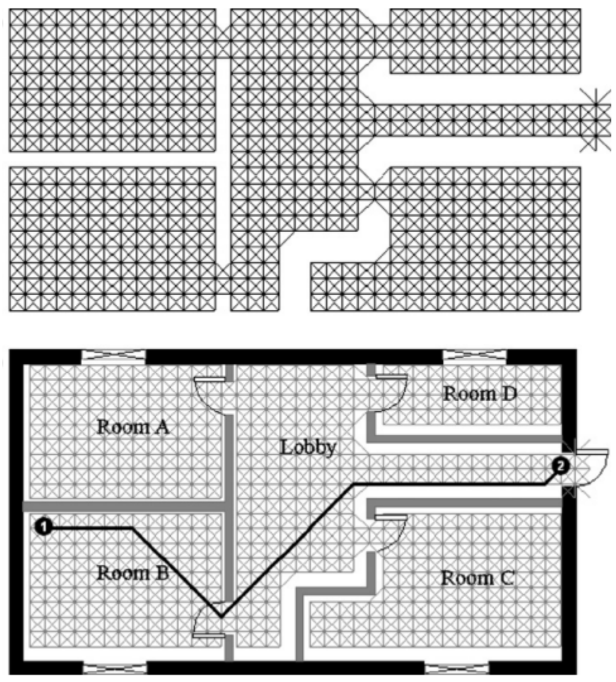


Figure 12: Free space represented by a grid graph and example of shortest path (work from [24])

lar units. Then, indoor space analyses could be conducted based on the resulting grid graph. Shortest path computations can be easily implemented as shown in Figure 12. Another grid-based approach is proposed in [3]. Object space is discretized into a 3D grid of uniform cells and an optimal obstacle-free path is generated between two points as a discrete cell path with consideration of surmountable and insurmountable obstacles. The grid is treated as graph with orthogonal links of uniform cost. Figure 13 displays the generated paths.

In [41] Yuan and Schneider propose a cube-based model called LEGO graph to compute feasible routes for different users according to their widths, heights, and other requirements. The indoor space is first represented by multiple cubes with different types. Then, according to the heights and types of the cubes, possible passages are generated merging cubes into larger blocks. Based on these blocks, feasible routes are computed by checking the availability of the connectors between different blocks. However, the resulting navigation model seems to be too coarse and an algorithm to compute optimal

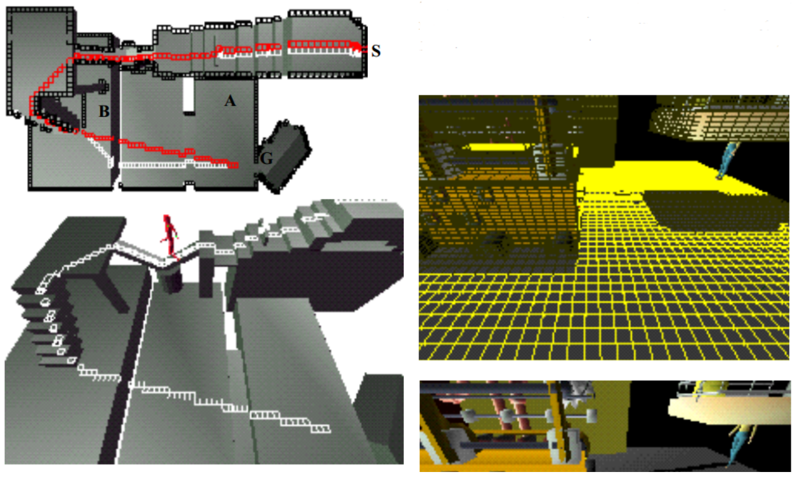


Figure 13: generated paths in a grid-based model (work from [3])

navigation routes is missing.

On the other hand, there are several models that try to convert the architectural structure into path networks. One of them is the aforementioned three-dimensional navigable model proposed by Lee in [22], in which the network is derived by Poincaré duality combined with a graph-theoretic framework and a hierarchical representation. How to technically construct the 3D geometrical network model is described in [23]: a skeleton-abstraction algorithm formally named Straight-Medial Axis Transformation (S-MAT) is proposed, which can abstract linear features from simple polygons, as displayed in Figure 14. The same approach has been found in [21].

Another Medial Axis-based network generation is proposed in [36]. A multi-layered environment is represented by a set of two dimensional layers and a set of connections. Each layer is a collection of two-dimensional polygons that all lie in a single plane and each connection provides a means of moving between layers. First, traditional medial axis is computed of each two-dimensional layer in the environment. The connections are then used to iteratively merge this collection of medial axes into a single data structure (See Figure 15).

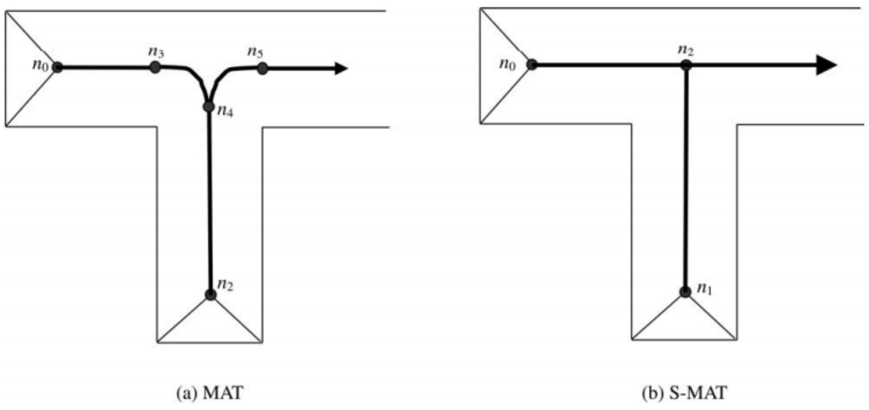


Figure 14: Straight-Medial Axis Transformation proposed by Lee)

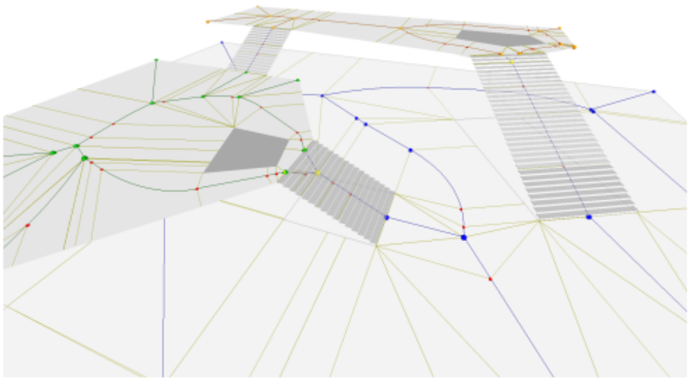


Figure 15: Navigation meshes for multi-layered environments

In [4] Becker et al. propose the so called "Multilayered Space-Event Model" (MLSEM), which consists in a conceptual framework for the modelling of indoor spaces, extending previous work from Lee to a multilayer representation of specific decomposition of buildings according to different semantic criteria, such as topographic space for 3D buildings and sensor space for sensor range partition (see Figure 16). Nodes in dual space represent possible states of a navigating subject or object, and joint states between the different space models mutually constrain possible locations in either space model. Space models for different sensors and topography can be represented independently from each other and changes to one of the models do not affect the structure of other models. About the automatic extraction of the geometric network model out of topographic space layer, it

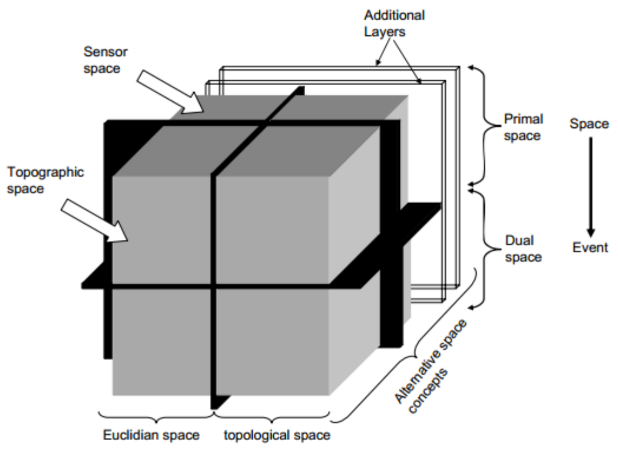


Figure 16: Multilayer combination of alternative space concepts (work from [4]

is suggested to use ISO 19107 primitives like "representativePoint()" and "centroid()". These methods return a point geometry representing the centroid of the volumetric object. It is clear that this approach leads to a very coarse network which could be unusable when the internal structure of a building is complex. Nevertheless, the need for the decomposition of rooms into smaller units is remarked. Figure 17

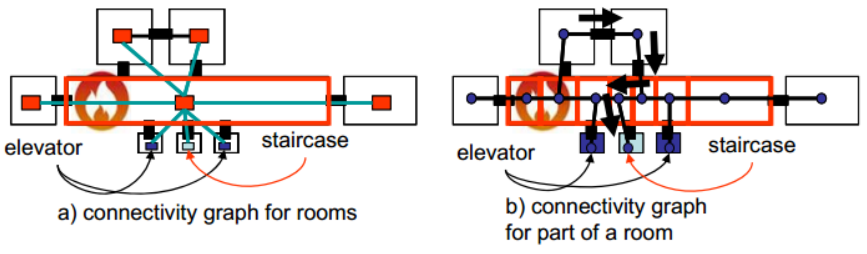

elevator
staircase
a) connectivity graph for rooms
elevator
staircase
b) connectivity graph for part of a room

Figure 17: The effect of spatial decomposition along escape routes (image taken from [4]

illustrates that a coarse-grained model in a fire scenario would have marked the whole corridor as non-navigable, hence smaller partitions of the topographic space can provide the necessary means for a more precise indoor route planning.

Yuan and Schneider proposed in [40] an indoor model that produces optimal non-circuitous routes. They consider building parts as cells and classify them into simple cells, complex cells, open cells, connectors. In their model doors are mapped to nodes and rooms are mapped to edges, with the assumption that doors are the destination of users and furthermore, this allows the construction of length-dependent routes. Moreover, a methodology for connecting doors in concave-shaped rooms is described. A similar methodology is presented by [25], introducing the so called "door-to-door path finding approach for indoor navigation". It consists in an algorithm applied to 2D floor plan of buildings with complex indoor structure, working on two routing levels, one to get coarse route between rooms, and the other applied to single rooms to acquire the detailed route. Doors (or openings) are approximated with nodes and the rooms with edges. This is in contrast to most currently available network models that treat doors (or openings) as edges connecting rooms (nodes). The algorithm strongly relies on Visibility Graph (VG) construction, a graph of intervisible locations, typically for a set of points and obstacles in the Euclidean plane. Each node in the graph represents a point location, and each edge represents a visible connection between them.

In [10] a model which represents indoor environments with topologic, semantic and metric information that allows nearly length-optimal routing in complex building structures (2D floor plan) is presented. The authors focus also on relevant parts of a complex indoor environment which are needed for network construction like corridors and stairs. Additionally they consider obstacles and special semantic areas inside rooms and how to integrate those areas in the routing graph (see Figure 18) . But there is no description on how to displace the nodes and how to automatically derive the network graph. Apparently the process seems to be not automatic.

In [17] Khan and Kolbe discuss the problem that most of the frameworks for indoor navigation focus on single type of locomotion. They define and formalize the requirements for three types of locomotion walking, driving, flying leading to the determination of individual constraints for each locomotion type. These constraints constitute the

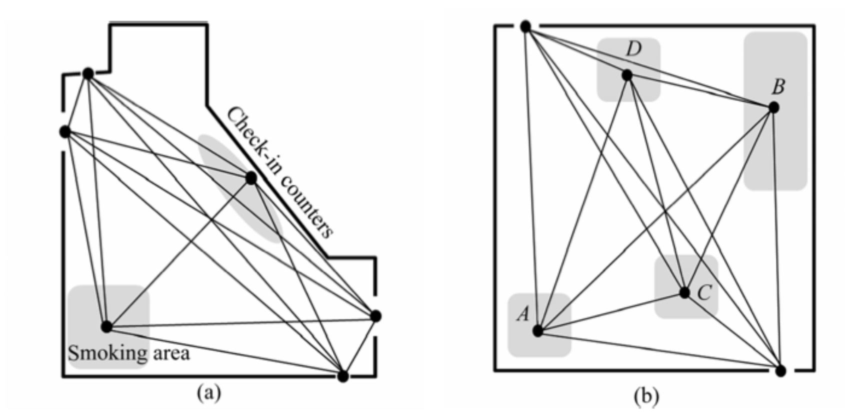
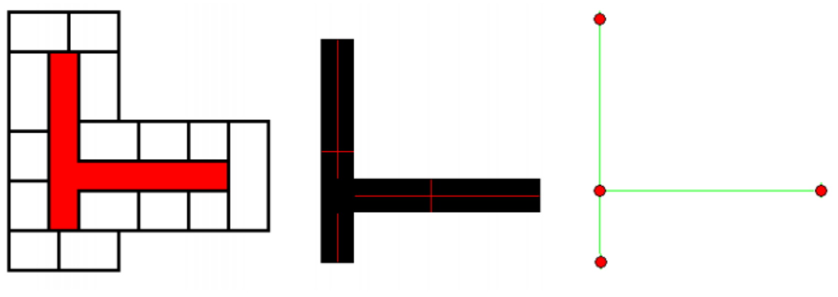


Figure 18: Routing graph for an airport entrance hall (a) and for an exhibition hall (b) [10]

input for the determination of navigable or non-navigable indoor space cells for the specific locomotion type, leading to different 3d subspacings of indoor space. But, a practical solution for the derivation of these different space cells from topographic space has not been introduced.

Regarding the automatic extraction of the Geometric Network Model from 3D semantically rich building models like CityGML and BIM a very little amount of references has been found in current literature. In [37] Vanclooster and De Maeyer claim that *"for the moment no universal automatic generation of 3D indoor networks has been developed"*. Most of the implemented solutions rely on skeleton-abstraction process, which could fail for large and arbitrary shaped spaces. In [16] an image processing method is proposed. It first constructs a 3D building model from 2D image based-floorplans and then extracts Geometric Network Model identifying corridors and rooms by image analysis. The network is generated by placing middle points inside rooms and selecting middle lines of the corridors, as displayed in Figure 19.

Another reference of automatic extraction can be found in [39]. A 3D-GIS based framework that supports BIM for topologic analysis-oriented indoor navigation is proposed. The BIM file serves geometric


(a) Detection of corridors

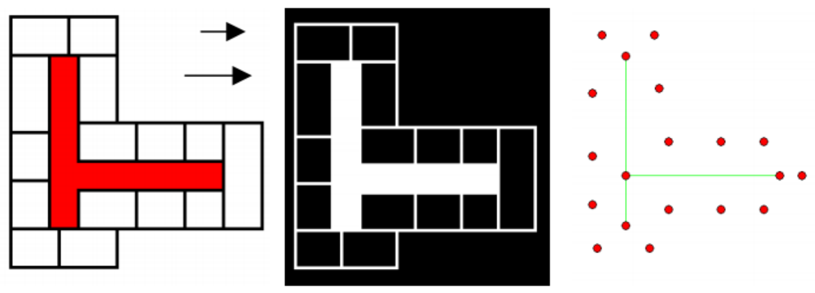

(b) Detection of rooms and network construction

Figure 19: Network generation process via image analysis according to [16]

and semantic information and it is utilized for creating the network model. Not a very detailed description about the graph model has been provided, but the automatic extraction process basically relies on Straight Medial Axis Transformation.

In [35] a transformation from an IFC-based Building Layout Information into a Geometric Network Model for Indoor Navigation is proposed. But also in this case it consists in a skeleton abstraction process that looks very similar to MAT with some minor modifications.

A very promising data model is presented in [26], named Indoor Navigation Space Model, designed for automatic derivation of connectivity graph. The model consists in an extension of the concepts presented in MLSEM framework and it provides further semantics specializations of building spaces in support of navigation. Furthermore a conceptual formal method for transforming the building data from other sources to INSM and a path planning solution based on "door-to-door" approach [25] are provided. Yet, the model has to be validated with real building data and complex configuration and a practical implementation of 3D building models data conversion is still missing.

Therefore, resuming, one of the more significant findings to emerge from this study of the current literature is that existing indoor navigation data models suffer from various problems. Nevertheless, the automatic derivation of connectivity network out of complex semantically rich 3D models like CityGML is still a challenging and not very straightforward task. Therefore, this research aims on one hand to provide a data model beneficial for indoor navigation applications and inspect geometric and semantic properties that might be potentially extracted from CityGML to fill the model as datasources, and on the other hand to provide new techniques for automatically deriving geometric-topologic networks from 3D topographic space and a finite amount of semantics.

3 RESEARCH METHODOLOGY

The aim of this research project is the automatic derivation of an Improved Geometric Network Model from CityGML LOD4. Hence the research process can be conceptually split into three components:

1. Extraction of semantics and geometric-topological properties from CityGML
2. Design and implementation of the Data Model
3. Design and implementation of algorithms for Network extraction

The first task consists in evaluating how CityGML could be beneficial for Indoor Navigation applications. Consequently some techniques for extracting semantics and gemoetric-topological properties out of CityGML have been proposed. This kind of information is essential for populating a specifically designed Data Model.

As it emerged from current literature existing Indoor Navigation Data Models suffer from various problems. Hence, the second component of this research project consists in the definition of the specifications of a novel Data Model capable of filling the limitations of the previous approaches. As it will be shown in next Chapters, the proposed Data Model has not been designed from scratch, on the contrary it has been defined as an extension of Indoor Navigation Module of OGC candidate standard named IndoorGML.

The third and most important conceptual aspect of this research involves the design and implementation of a novel algorithmic approach for the automatic derivation of a connectivity graph out of 3D topographic indoor spaces and a finite amount of semantics. Figure 20 displays the research methodology workflow diagram that served as a continuous point of reference to evaluate job order working progress.

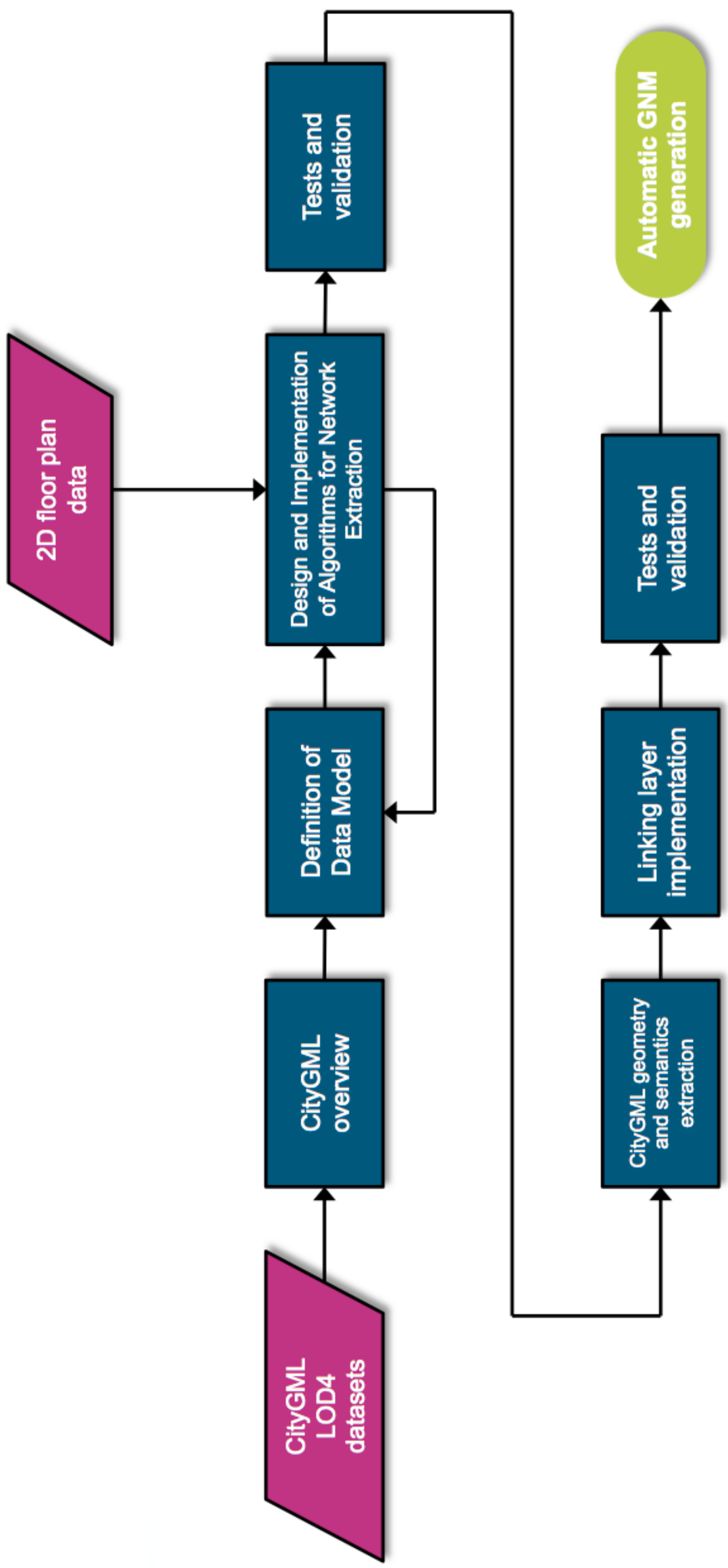


Figure 20: Research Methodology Workflow Diagram

- CityGML LOD4 datasets: as previously stated CityGML is intended to be the data input for this research project.

- CityGML overview: this process consists in an analysis of CityGML format, in order to better understand potential semantic and topological properties of this information model. The motivation of the introduction of this phase along the workflow is that although it has been given pre-emption to the definition of the algorithm, the Data Model couldn't have been a priori determined without considering at all the underlying input dataset.

- Definition of the Data Model: having in mind what are the semantics that can be potentially extracted from CityGML, it is possible to design the Data Model with Unified Modelling Language (UML). The model contains all of the classes and objects that in a second phase are manipulated by the network extraction algorithms. Semantics and geometry extracted from CityGML will be fetched into the model. During this phase also Improved Geometric Network specifications have been designed.

- Design and implementation of Algorithms for network extraction: this is the most crucial process of the workflow. Network extraction algorithm has been designed and implemented. The proposed approach, as it will be described later, is 2D based. For this reason, since data extraction from CityGML has not been yet implemented, factitious 2D floor plan datasets have been used. The feedback loop linked to the previous process block is motivated by the fact that some modifications made during this process had some significant impact on the other and vice-versa.

- Test and validation: this process consists in verifying if the algorithms have produced the desired result, responding to the requirements outlined in the previous processes.

- CityGML geometry and semantics extraction: during this process necessary geometrical/topological and semantic information have been extracted from CityGML. The implementation of the extraction consisted in two steps: firstly, the CityGML document has been parsed; secondly, for each entity (e.g. a room) geometry and semantics have been extracted based on data model requirements.

- Linking layer implementation: Some mediator classes have been implemented in order to link CityGML data extraction process with network generation algorithms. Efficient low coupling between the two modules is a best practice in order to achieve maintainability, readability, lower dependency between the classes.

- Test and validation: in order to verify that extraction and linking processes have been correctly implemented some tests have been conducted.

4 CITYGML: DERIVATION OF GEOMETRIC, TOPOLOGICAL AND SEMANTIC PROPERTIES

Part of this research involves the analysis of CityGML in order to estimate how the latter could be beneficial for Indoor Navigation applications. This study is motivated by the fact that most of Indoor Navigation Frameworks and Data Models assume that geometry and semantics of Indoor Spaces are given, whereas in real scenarios this kind of information is incomplete or inconsistent most of the time. The CityGML Model might provide the topographic space of the indoor environment. Furthermore it presents geometric and topological relationships and certain semantics of the indoor environment. So this Model can potentially provide a part of or all of the necessary information for indoor navigation, despite it is not specifically designed for this purpose. Furthermore, it might help in taking into account obstacles that can disturb navigation process. For instance, in the CityGML building model a class named *IntBuildingInstallation* represents *"an object inside a building with a specialized function or semantic meaning"* [13]. Thus it can be regarded as a potential fixed obstacle, which means it can't be moved when pedestrians attempt to get through the space which it occupies. Another class named *BuildingFurniture* can be considered as the potential moveable obstacles, such as chairs which can be budged.

In this Chapter, more background information on CityGML will be exposed, then research methodology for deriving meaningful information for Indoor Navigation out of CityGML will be outlined.

4.1 CITYGML BACKGROUND INFORMATION

CityGML is a common semantic information model for the representation of 3D urban objects that can be shared over different applications. The latter capability is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the possibility of selling the same data to customers from different application fields. The targeted application areas explicitly include city planning, architectural design, tourist and leisure activities, environmental simulation, mobile telecommunication, disaster management, homeland security, real estate management, vehicle and pedestrian navigation, and training simulators [13].

CityGML is designed as an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is implemented as an application schema of the Geography Markup Language 3 (GML3), the extendible international standard for spatial data exchange and encoding issued by the Open Geospatial Consortium (OGC)[1] and the ISO TC211. CityGML is based on a number of standards from the ISO 191xx family, the Open Geospatial Consortium, the W3C Consortium, the Web 3D Consortium, and OASIS.

CityGML defines the classes and relations for the most relevant topographic objects in cities and regional models with respect to their *geometrical, topological, semantical,* and *appearance* properties. "City" is broadly defined to comprise not just built structures, but also elevation, vegetation, water bodies, "city furniture", and more. Included are generalisation hierarchies between thematic classes, aggregations, relations between objects, and spatial properties. CityGML is applicable for large areas and small regions and can represent the terrain and 3D objects in different levels of detail simultaneously. Since either simple, single scale models without topology and few semantics or very complex multi-scale models with full topology and fine-grained

1 http://www.opengeospatial.org/standards/citygml

semantical differentiations can be represented, CityGML enables lossless information exchange between different GI systems and users [13].

4.1.1 *CityGML modularization*

The CityGML data model consists of class definitions for the most important types of objects within virtual 3D city models. These classes have been identified to be either required or important in many different application areas. However, implementations are not required to support the overall CityGML data model in order to be conformant to the standard, but may employ a subset of constructs according to their specific information needs. For this purpose, modularisation is applied to the CityGML data model The CityGML data model is

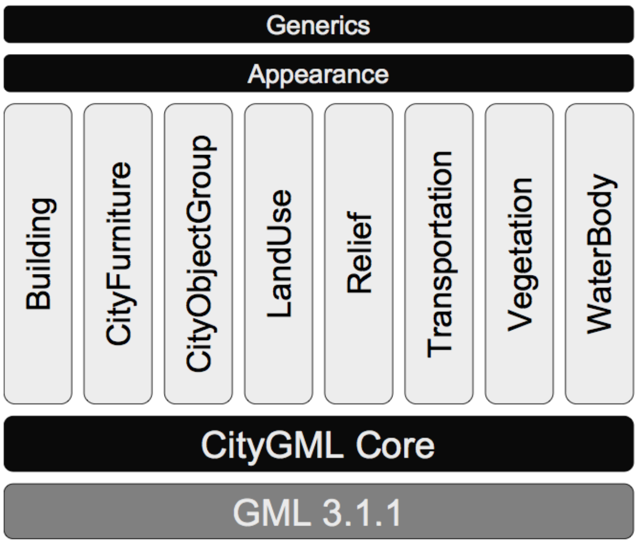


Figure 21: Modularization in CityGML

thematically decomposed into a *core module* and thematic *extension modules* (see Figure 21). The core module comprises the basic concepts and components of the CityGML data model and, thus, must be implemented by any conformant system. Based on the core module, each extension covers a specific thematic field of virtual 3D city models. CityGML introduces the following thirteen thematic extension modules:

- Appearance
- Bridge
- Building
- CityFurniture
- CityObjectGroup
- Generics
- LandUse
- Relief
- Transportation
- Tunnel
- Vegetation
- WaterBody
- TexturedSurface (deprecated)

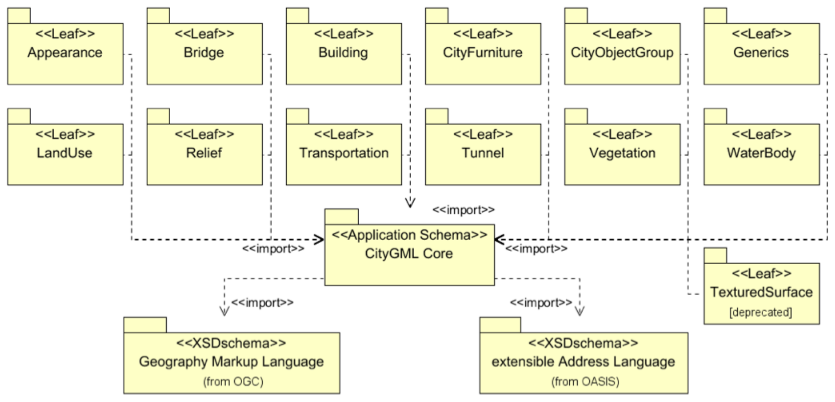


Figure 22: Modularization in CityGML: packages. Each extension module further imports the GML 3.1.1 schema definition in order to represent spatial properties of its thematic classes.

Each CityGML module is specified by its own XML Schema definition file and is defined within an individual and globally unique XML target namespace. According to dependency relations between modules, each module may, in addition, import namespaces associated to such related CityGML modules. However, a single namespace shall not be directly included in two modules. Thus, all elements belonging to one module are associated to the module's namespace only. By this means, module elements are guaranteed to be properly separated and distinguishable in CityGML instance documents.

4.1.2 *The concept of Level of Detail*

CityGML supports different Levels of Detail (LOD) (see Figure 23). LODs are required to reflect independent data collection processes with differing application requirements. LODs facilitate efficient visualisation and data analysis. In a CityGML dataset, the same object may be represented in different LOD simultaneously, enabling the analysis and visualisation of the same object with regard to different degrees of resolution. Furthermore, two CityGML data sets containing the same object in different LOD may be combined and integrated. However, it will be within the responsibility of the user or application to make sure objects in different LODs refer to the same real-world object. The coarsest level LOD0 is essentially a two and a half dimensional Digital Terrain Model over which an aerial image or a map may be draped. Buildings may be represented in LOD0 by footprint or roof edge polygons. LOD1 is the well-known blocks model comprising prismatic buildings with flat roof structures. In contrast, a building in LOD2 has differentiated roof structures and thematically differentiated boundary surfaces. LOD3 denotes architectural models with detailed wall and roof structures potentially including doors and windows. LOD4 completes a LOD3 model by adding interior structures for buildings. For example, buildings in LOD4 are composed of rooms, interior doors, stairs, and furniture. In all LODs appearance information such as high-resolution textures can be mapped onto the structures [13].

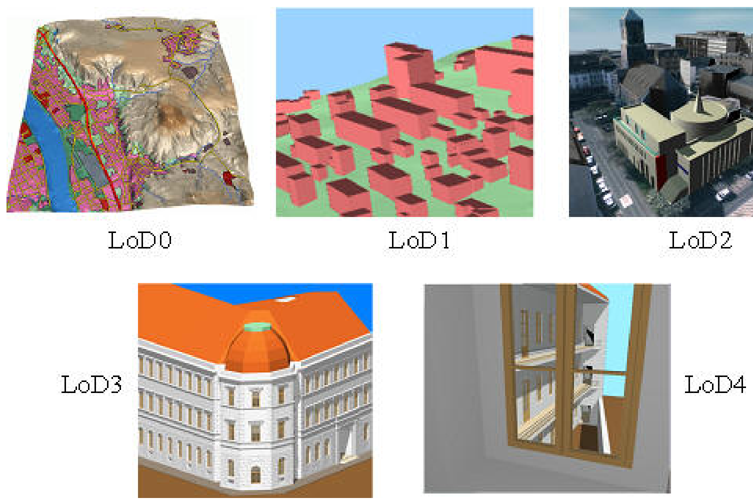


Figure 23: Different LODs in CityGML

4.1.3 *Semantics, Geometry, Topology*

One of the most important design principles for CityGML is the coherent modelling of semantics and geometrical/topological properties. Semantics enrich the model giving a meaning to the geometry. By using semantics, it can be known that one polygon is part of the ceiling surface of a building while another represents a wall. At the semantic level, real-world entities are represented by features, such as buildings, walls, windows, or rooms. The description also includes attributes, relations and aggregation hierarchies (part-whole-relations) between features. Thus the part-of-relationship between features can be derived at the semantic level only, without considering geometry. However, at the spatial level, geometry objects are assigned to features representing their spatial location and extent. So the model consists of two hierarchies: the semantic and the geometrical in which the corresponding objects are linked by relationships [32]. The advantage of this approach is that it can be navigated in both hierarchies and between both hierarchies arbitrarily, for answering thematic and/or geometrical queries or performing analyses. If both hierarchies exist for a specific object, they must be coherent (i.e. it must be ensured that they match and fit together). For example, if a wall of a building has two windows and a door on the semantic level, then the geometry representing the wall must contain also the geometry parts of both windows and the door [13].

Spatial properties of CityGML features are represented by objects of GML3's geometry model. This model is based on the standard ISO 19107 "Spatial Schema" [14], describing 3D geometry according to the well-known Boundary Representation (B-Rep). CityGML actually uses only a subset of the GML3 geometry package, defining a profile of GML3. B-Rep represents a solid indirectly by a representation of its bounding surface. The merit of a B-Rep is that a solid is bounded by its surface and has its *interior* and *exterior*.Normally a face is a bounded region of a planar, quadratic, toroidal, or sculptured surface. The bounded region of the surface that forms the face is represented by a closed curve that lie on the surface. A face can have several bounding curves to represent holes in a solid. The bounding curves of faces are represented by edges. The portion of the curve that forms the edge is represented by two vertices. A B-Rep model has to fulfil the following conditions: The set of faces forms a complete skin of the solid with no missing parts, and faces do not intersect each other except at common vertices or edges. Furthermore, the boundaries of faces do not intersect themselves. These conditions disallow self-intersecting and open objects. Figure 24 displays an example of B-Rep. GML3 provides a special aggregate for each dimension, a *MultiPoint*, a *MultiCurve*, a *MultiSurface*, and a *MultiSolid*, and they may be disjoint, overlapping, touching, or disconnected. As a consequence, the geometry in CityGML can be modelled as *MultiSurface*, when the topological relationships between surfaces are not known. Otherwise, to ensure valid geometry, Composite objects like *CompositeSolid*,

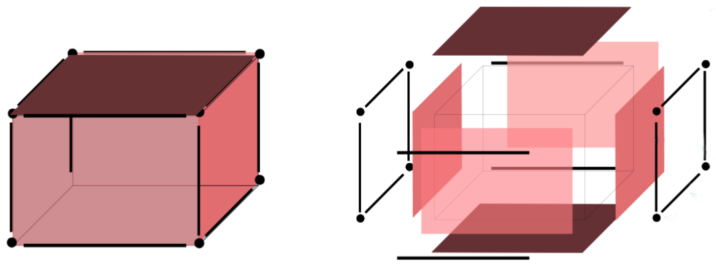
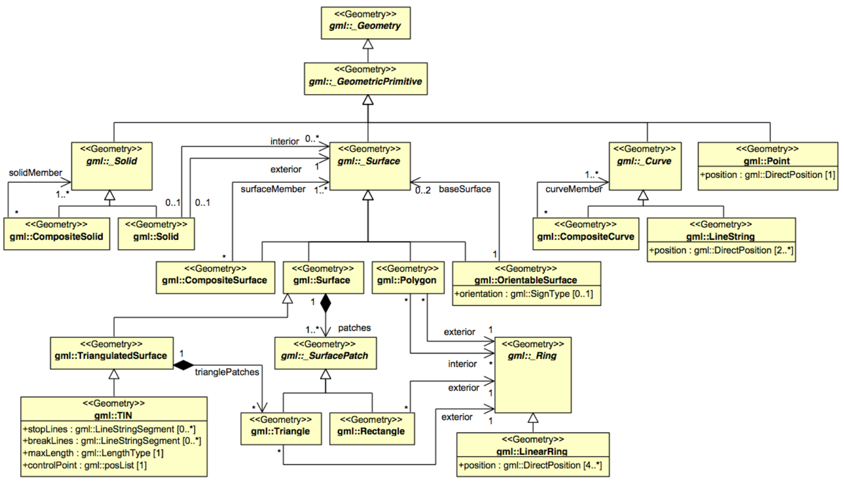


Figure 24: Representation of a 3D-Solid by B-Rep

a *CompositeSurface* must be used. Figure 25 displays UML diagram of CityGML's geometry model.



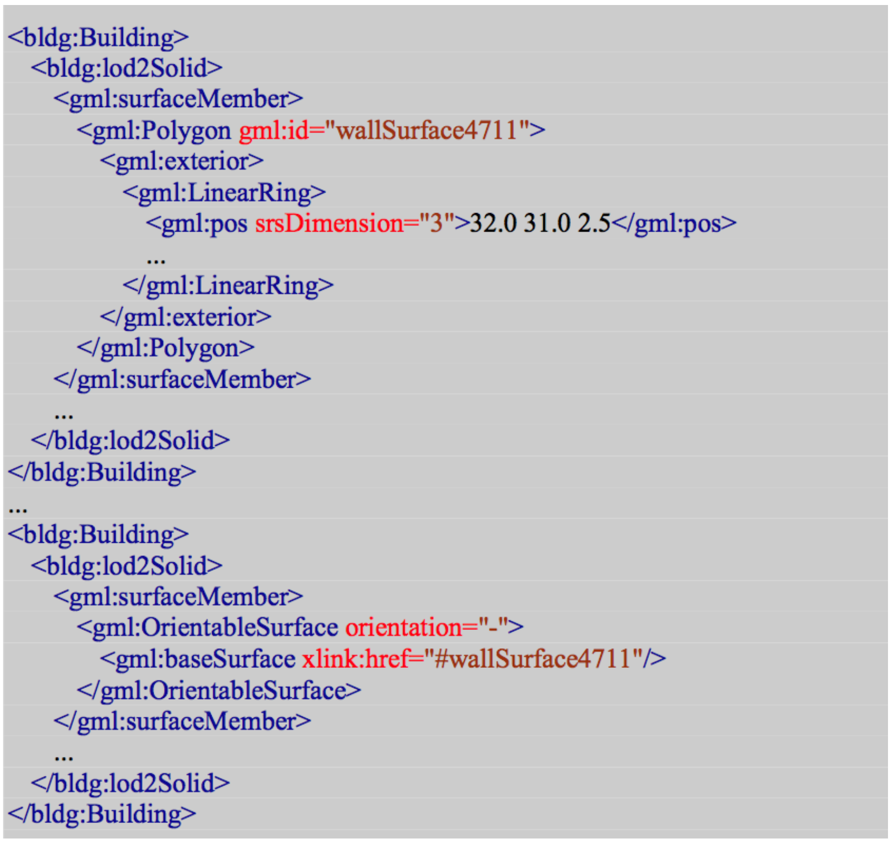
<<Geometry>> gml::_Geometry
<<Geometry>> gml::_GeometricPrimitive
<<Geometry>> gml::_Solid
<<Geometry>> gml::_Surface
<<Geometry>> gml::_Curve
<<Geometry>> gml::Point
+position : gml::DirectPosition [1]
solidMember
1..*
*
<<Geometry>> gml::CompositeSolid
<<Geometry>> gml::Solid
interior 0..*
exterior 1
0..1
0..1
surfaceMember 1..*
0..2 baseSurface
curveMember
1..*
*
<<Geometry>> gml::CompositeCurve
<<Geometry>> gml::LineString
+position : gml::DirectPosition [2..*]
*
<<Geometry>> gml::CompositeSurface
<<Geometry>> gml::Surface
<<Geometry>> gml::Polygon
1
<<Geometry>> gml::OrientableSurface
+orientation : gml::SignType [0..1]
1
1..* patches
* *
exterior 1
interior *
<<Geometry>> gml::TriangulatedSurface
1
trianglePatches
<<Geometry>> gml::_SurfacePatch
<<Geometry>> gml::_Ring
exterior 1
exterior 1
<<Geometry>> gml::TIN
+stopLines : gml::LineStringSegment [0..*]
+breakLines : gml::LineStringSegment [0..*]
+maxLength : gml::LengthType [1]
+controlPoint : gml::posList [1]
*
<<Geometry>> gml::Triangle
*
<<Geometry>> gml::Rectangle
*
<<Geometry>> gml::LinearRing
+position : gml::DirectPosition [4..*]

Figure 25: UML diagram of CityGML's geometry model

Regarding topology, CityGML provides the explicit modelling of topological relationships, for example the sharing of geometry objects between features or other geometries. One part of space is represented only once by a geometry object and is referenced by all features or more complex geometries which are defined or bounded by this geometry object. Thus redundancy is avoided and explicit topological relations between parts are maintained. Basically, there are three cases. First, two features may be defined spatially by the same geometry. For example, if a path is both a transportation feature and a vegetation feature, the surface geometry defining the path is referenced both by the transportation object and by the vegetation object.

Second, geometry may be shared between a feature and another geometry. A geometry defining a wall of a building may be referenced twice: by the solid geometry defining the geometry of the building, and by the wall feature. Third, two geometries may reference the same geometry, which is in the boundary of both. For example, a building and an adjacent garage may be represented by two solids. The surface describing the area where both solids touch may be represented only once and it is referenced by both solids [13].

Topology can be practically implemented by the usage of *Xlink* construct provided by GML3: each geometric object may be referenced wherever else by its unique identifier, using a *href* attribute (see Figure 26).



```
<bldg:Building>
  <bldg:lod2Solid>
    <gml:surfaceMember>
      <gml:Polygon gml:id="wallSurface4711">
        <gml:exterior>
          <gml:LinearRing>
            <gml:pos srsDimension="3">32.0 31.0 2.5</gml:pos>
            ...
          </gml:LinearRing>
        </gml:exterior>
      </gml:Polygon>
    </gml:surfaceMember>
    ...
  </bldg:lod2Solid>
</bldg:Building>
...
<bldg:Building>
  <bldg:lod2Solid>
    <gml:surfaceMember>
      <gml:OrientableSurface orientation="-">
        <gml:baseSurface xlink:href="#wallSurface4711"/>
      </gml:OrientableSurface>
    </gml:surfaceMember>
    ...
  </bldg:lod2Solid>
</bldg:Building>
```

Figure 26: example of topological relationship in CityGML

In the next Section some rules are defined. They have been formalized in order to constitute a suitable base to derive from CityGML datasets essential information for Indoor Navigation. The aim of these rules is to characterize indoor space in an univocal and unambiguous way.

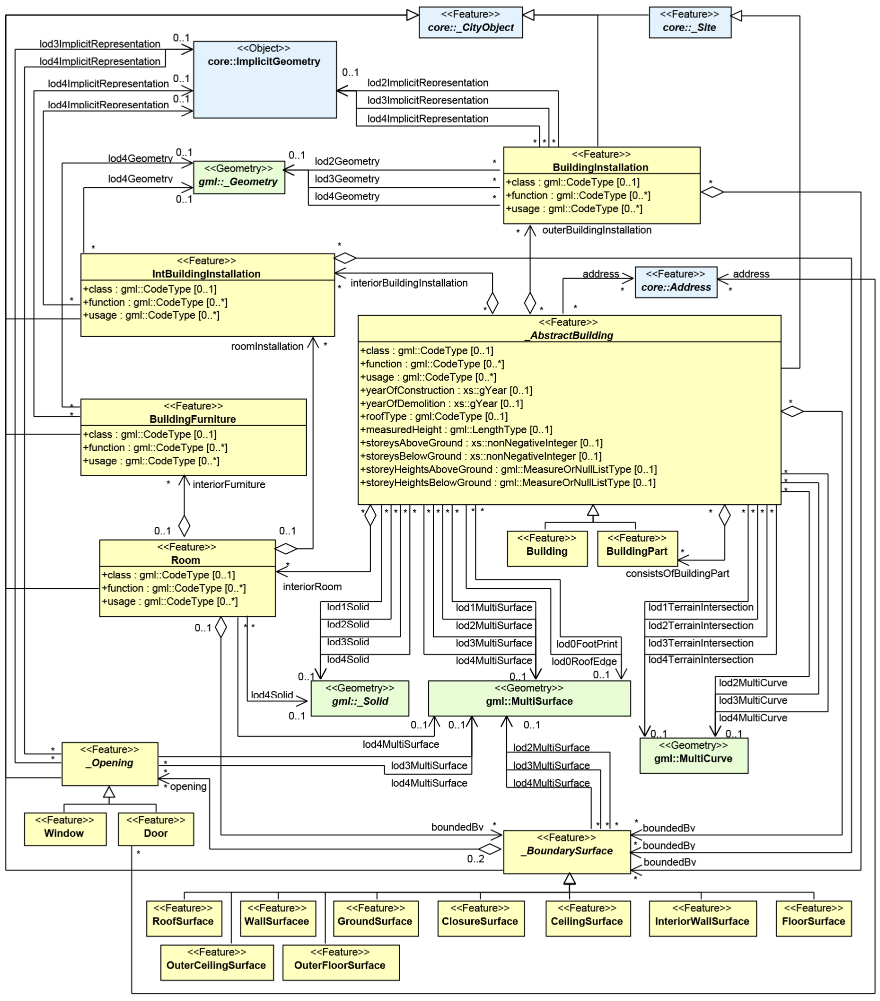
4.2 FORMAL RULES FOR MODELING INDOOR SPACE

- The navigable space of a room is a space that users can move freely in. Within this context we specifically refer to pedestrian movement.
- Objects that obstruct the natural pedestrian movement are considered as obstacles. A refined classification can be inferred:
  - objects that are permanently attached to the building structure and cannot be moved are fixed obstacles
  - objects that are a movable part of a room are removable obstacles
- Two rooms are defined as being adjacent, if they have common openings like doors, windows or hatches.
- Vertical connectors are objects that allow users to move in vertical direction (up and down) and usually connect pedestrian navigable spaces located at different heights.

In the next Section it will be displayed how semantic, geometric, topological properties have been extracted from CityGML, based on the previously defined rules. It has been inspected how CityGML could contribute in providing essential information that could be fetched into a Data Model or into an Application Framework for Indoor Navigation, since it meets the basic semantic requirements of indoor modeling, describing building interiors with LOD4.

4.3 DERIVATION OF GEOMETRIC AND SEMANTIC PROPERTIES

As previously stated, the focus of this part of the research is on CityGML. More specifically, LOD4 has been analysed because of its capability of representing interiors. Firstly, the Building thematic extension module has been inspected, in order to obtain a holistic perspective of how indoor spaces have been modelled within CityGML. As displayed in Figure 27, the main class is the abstract class _Abstract-



<<Feature>> core::_CityObject
<<Feature>> core::_Site
<<Object>> core::ImplicitGeometry
<<Geometry>> gml::_Geometry
<<Feature>> BuildingInstallation
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
outerBuildingInstallation
<<Feature>> IntBuildingInstallation
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
interiorBuildingInstallation
<<Feature>> core::Address
address
<<Feature>> _AbstractBuilding
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
+yearOfConstruction : xs::gYear [0..1]
+yearOfDemolition : xs::gYear [0..1]
+roofType : gml::CodeType [0..1]
+measuredHeight : gml::LengthType [0..1]
+storeysAboveGround : xs::nonNegativeInteger [0..1]
+storeysBelowGround : xs::nonNegativeInteger [0..1]
+storeyHeightsAboveGround : gml::MeasureOrNullListType [0..1]
+storeyHeightsBelowGround : gml::MeasureOrNullListType [0..1]
roomInstallation
<<Feature>> BuildingFurniture
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
interiorFurniture
<<Feature>> Room
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
interiorRoom
<<Feature>> Building
<<Feature>> BuildingPart
consistsOfBuildingPart
<<Geometry>> gml::_Solid
<<Geometry>> gml::MultiSurface
<<Geometry>> gml::MultiCurve
lod0FootPrint
lod0RoofEdge
<<Feature>> _Opening
opening
<<Feature>> Window
<<Feature>> Door
<<Feature>> _BoundarySurface
boundedBy
<<Feature>> RoofSurface
<<Feature>> WallSurface
<<Feature>> GroundSurface
<<Feature>> ClosureSurface
<<Feature>> CeilingSurface
<<Feature>> InteriorWallSurface
<<Feature>> FloorSurface
<<Feature>> OuterCeilingSurface
<<Feature>> OuterFloorSurface

Figure 27: UML diagram of the building module from CityGML

Building which may be realized either by *Building* or by *BuildingPart*. The latter is used to describe the structural part of a complex Building, but if a Building consists of only one block, the former class is used. Both of the classes inherit the attributes of their abstract class:

the class of the building, function, usage, year of construction, year of demolition, roof type, measured height, and number and individual heights of the storys above and below ground.

The next sections, till the end of the chapter, illustrate how semantic, geometric, topological properties have been analysed and derived from CityGML. Specifically, it will be shown how geometry of walkable and drivable surfaces has been extracted from 3D topographic space (Section 4.3.1), how topological concept of connectedness/adjacency of spaces has been derived from thematic objects like *Room* (Section 4.3.2), how vertical connectivity between spaces/storeys is represented (Section 4.3.3), and finally how furniture and building installations could be meaningful in the determination of obstacle-free paths (Section 4.3.4).

4.3.1 *Derivation of walkable – drivable surfaces*

A building in CityGML consists of rooms, where *Room* is a space surrounded by different boundary surfaces that represents all type of spaces as semantic objects for modeling the free spaces inside a building itself. Storeys are not explicitly defined but they can be considered as an explicit aggregation of all building features according to arbitrary user defined criteria on a certain height level. Geometrically, spaces within the Building Module are not objects in itself, as Figure 28 depicts. Differently from IFC, they are represented by boundary surfaces that topologically close the rooms (B-Rep). Spaces should be closed (if necessary by using *ClosureSurfaces*) and their geometries normally are described by a solid (*lod4Solid*). However, if the topological correctness of the boundary cannot be guaranteed, the geometry can alternatively be given as a *MultiSurface (lod4MultiSurface)*. The surface normals of the outer shell of a GML solid must point outwards. In addition to the geometrical representation, different parts of the visible surface of a room can be modelled by specialised *Bound-*

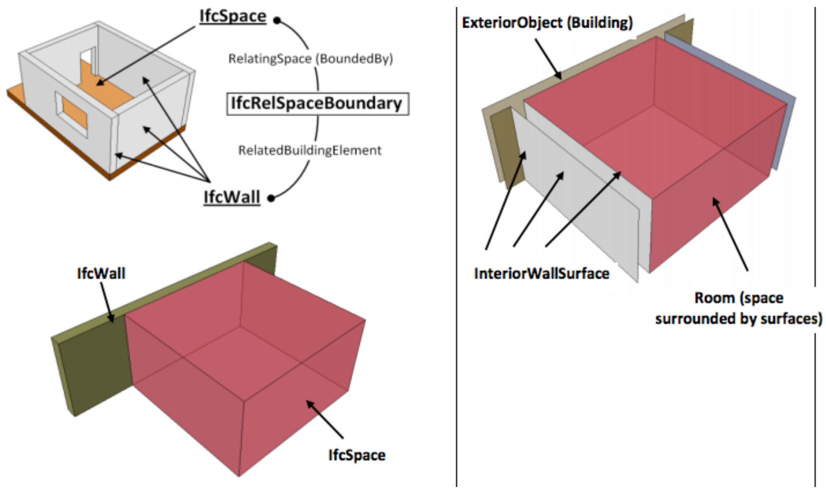


Figure 28: CityGML geometric representation of indoor spaces (i.e. Rooms)

arySurfaces (*FloorSurface*, *CeilingSurface*, *InteriorWallSurface*, *ClosureSurface*). Since the algorithmic approach for Network derivation is a 2D-based approach, it is essential to determine walkable and drivable surfaces that will be potentially taken into account for Indoor Navigation. In other words, the aggregation of the footprints of each *Room* space constitutes the overall Navigable Space of the building. How to derive these spaces has been implemented by adopting the following strategy:

- For each *Room*, where possible it has been extracted the geometric information of the *FloorSurface* property, instead of inspecting *lod4Solid* and *lod4Multisurface*, because it already provides geometric description of walkable surface (it is used in the LOD4 interior building model for modelling the floor of a room). As Listing 1 displays, retrieving geometric information is quite a straightforward task.

- Since *FloorSurface* is optional, this kind of information is not always present, so in case the CityGML file is missing this class instantiation, the geometry representing the shell of the *Room* has to be inspected: as already stated it may be either *lod4Solid* or *lod4Multisurface*. As Listing 2 suggests, it is more complex to retrieve floor geometric information yet by following this latter approach. In fact, all of the *surfaceMembers* composing *lod4Solid* have to be inspected. The surface that describes floor space can be clearly identified because it has the lowest *z*-coordinate (which is, in this particular case, the surface instantiated at Line 31. For more clarity coordinates of the whole set of surfaces have not been reported).

Listing 1: Excerpt of CityGML: FloorSurface class

```
1  /...
2  <bldg:boundedBy>
3     <bldg:FloorSurface>
4        <gml:name>Floor</gml:name>
5        <bldg:lod4MultiSurface>
6           <gml:MultiSurface>
7              <gml:surfaceMember>
8                 <gml:Polygon gml:id="PolyID63169_246_262718_408360">
9                    <gml:exterior>
10                      <gml:LinearRing
11                        gml:id="PolyID63169_246_262718_408360_0">
12                        <gml:pos>445542.82182673 5444898.93782957 2.94 </gml:pos>
13                        <gml:pos>445543.42553037 5444899.09959148 2.94 </gml:pos>
14                        <gml:pos>445542.42907705 5444902.81840596 2.94 </gml:pos>
15                        <gml:pos>445539.31396626 5444901.98371449 2.94 </gml:pos>
16                        <gml:pos>445539.33078950 5444901.92092931 2.94 </gml:pos>
17                        <gml:pos>445539.25834506 5444901.90151788 2.94 </gml:pos>
18                        <gml:pos>445539.48739991 5444901.04667359 2.94 </gml:pos>
19                        <gml:pos>445539.55984435 5444901.06608495 2.94 </gml:pos>
20                        <gml:pos>445540.31041958 5444898.26490006 2.94 </gml:pos>
21                        <gml:pos>445541.24253800 5444898.51466043 2.94 </gml:pos>
22                        <gml:pos>445541.29818410 5444898.30698638 2.94 </gml:pos>
23                        <gml:pos>445542.87747282 5444898.73015552 2.94 </gml:pos>
24                        <gml:pos>445542.82182677 5444898.93782957 2.94 </gml:pos>
25                      </gml:LinearRing>
26                   </gml:exterior>
27                </gml:Polygon>
28             </gml:surfaceMember>
29          </gml:MultiSurface>
30       </bldg:lod4MultiSurface>
31    </bldg:FloorSurface>
32 </bldg:boundedBy>
33 ...
```

Listing 2: Excerpt of CityGML: lod4Solid geometry

```
/...
<bldg:interiorRoom>
   <bldg:Room gml:id="GMLID_BUI192177_154_13254">
      <gml:description>Kinderzimmer</gml:description>
      <gml:name>203</gml:name>
      <bldg:lod4Solid>
         <gml:Solid>
            <gml:exterior>
               <gml:CompositeSurface>
                  <!-- Wall East -->
                  <gml:surfaceMember>
                     <gml:OrientableSurface orientation="-">
                        <gml:baseSurface
                           xlink:href="#GML_6d2d4915-1568-4337-895c-62a02dc63751"
                        />
                     </gml:OrientableSurface>
                  </gml:surfaceMember>
                  <!-- Wall East -->
                  <!-- Window East 4 -->
                  <gml:surfaceMember xlink:href="PolyID3861_336_46376_225301"> </
                      gml:surfaceMember>
                  <!-- Window East 4 -->
                  <!-- Wall North -->
                  <gml:surfaceMember>
                     <gml:OrientableSurface orientation="-">
                        <gml:baseSurface
                           xlink:href="#PolyID63168_292_89310_266717"/>
                     </gml:OrientableSurface>
                  </gml:surfaceMember>
                  <!-- Wall North -->
                  <!-- Floor -->
                  /<gml:surfaceMember>
                     <gml:OrientableSurface orientation="-">
                        <gml:baseSurface
                           xlink:href="#PolyID63169_246_262718_408360"/>
                     </gml:OrientableSurface>
                  </gml:surfaceMember>
                  <!-- Floor -->
...
```

Moreover, the concept of Storey has been derived, even though it is not specifically defined in CityGML Schema. For each room inspected during the parsing of CityGML file, it has been recorded the *z*-coordinate of its own floor surface. Then, the rooms have been grouped based on these collected values: to each retrieved *z*-value it has been added or subtracted a user-defined positive quantity in order to define a fault tolerance capable of filling potential discrepancies between different room heights but still appealing to the same storey. Figure 29 displays on left the visualization of CityGML LOD4 building, whereas on the right the visualization of storeys as a result of aggregation process of different Room spaces.

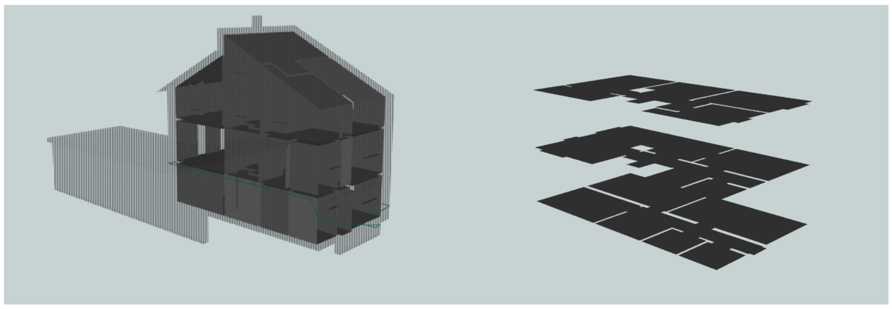


Figure 29: derivation of concept of Storeys from CityGML

4.3.2 *Derivation of connectivity between spaces*

One of the most important tasks for Indoor Navigation applications consists in analysing topological properties of spaces such as connectedness. Indeed, as already discussed in Chapter 1, the adjacency of room spaces impies an accessibility graph. In fact the so called Node Relation Structure, via Poincarè Duality, transforms discrete 3D objects of buildings (i.e. rooms) into nodes, and their topological relationships (i.e. adjacency) into arcs, preserving all topological properties under the duality transformation. In CityGML the room solids are topologically connected by the surfaces representing hatches, doors or closure surfaces that seal open doorways. Rooms are defined as being adjacent, if they have common *_Openings* or *ClosureSurfaces* [13]. Practically, connectivity has been identified implementing the following approach:
for each *Room* every *InteriorWallSurface* has been scanned, in order to evaluate if the aforementioned surface contains an *_Opening* (Fig-

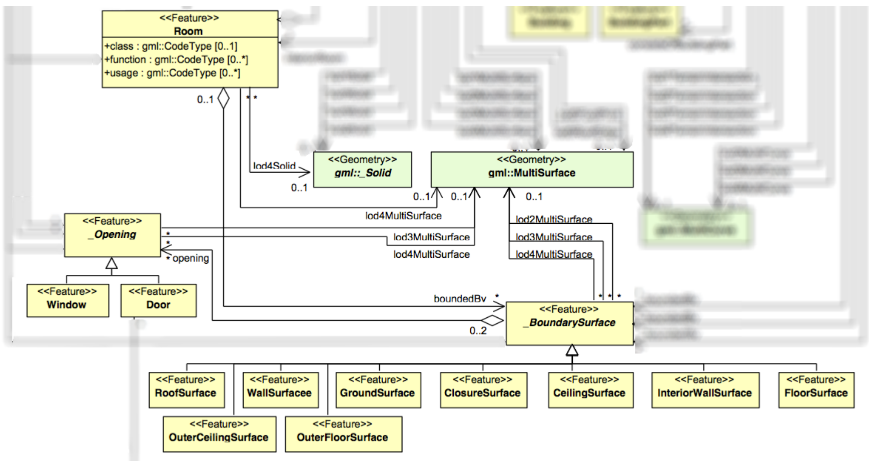


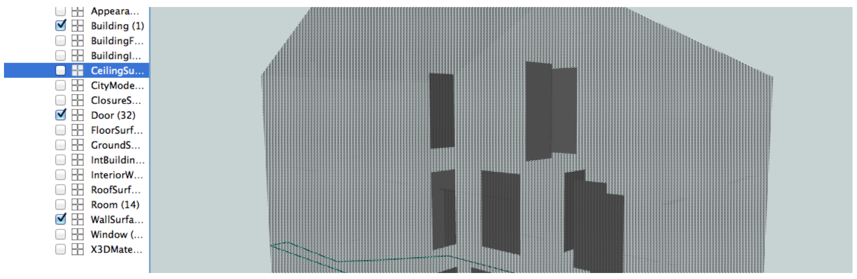
Figure 30: CityGML Building Module's classes involved in the determination of connectivity between spaces

ure 30 displays the relationships between *Room, InteriorWallSurface, _Opening* UML classes of the CityGML Building Module. The thematic surfaces bounding a room are referenced by the boundedBy property, whereas Window and Door objects are defined as subclasses of the abstract class Opening). Two rooms are connected if both use the same opening object or the same closure surface. Now, two possible alternatives may occur:

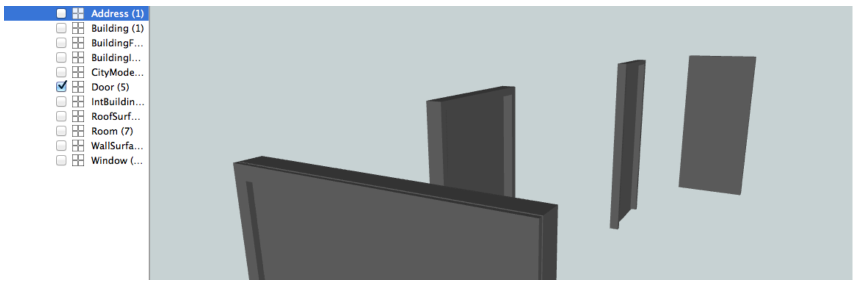
- the opening is referenced by both rooms on the semantic level
- the surface that represents the opening geometrically is part of the boundaries of the solids of both rooms

In the first case, as previously mentioned in Section 4.1.3, the *Xlink* construct provided by GML3 allows to determine where the *_Opening* has been referenced within the CityGML file. It is important to recall the fact that a thematic opening instance will figure twice in the gml file yet with different geometries (since door panel is a surface).
If the opening is not referenced on the semantic level, connectivity has been assessed by performing geometric operations. This sentence highlights the problem that in this latter case, some extra overhead in computation is added: the CityGML file has to be parsed and for each *_Opening* geometric operations have to be performed in order

to check if the geometries of the two *_Openings* being analysed are exactly one counterpart of the other. As Figure 31 displays, it is not possible to a-priori determine the geometric dimension of an *_Opening* in CityGML. They may be surfaces or complex solids. This means that assessing if the two objects are adjacent, or touch each other, or intersect is not an easy and straightforward task. More information about 3D Topological Relationships can be found in [43]. This topic is out of the scope of this research, so operators to assess topological relashionships have been hardcoded based on the quality of the input and datasets. For instance, the 3D geometries of Openings have been projected on 2D floor plan, so the problem has been reduced of 1 dimension and topological relationships between 2D polygons on plane have been assessed.



(a) Door as a surface

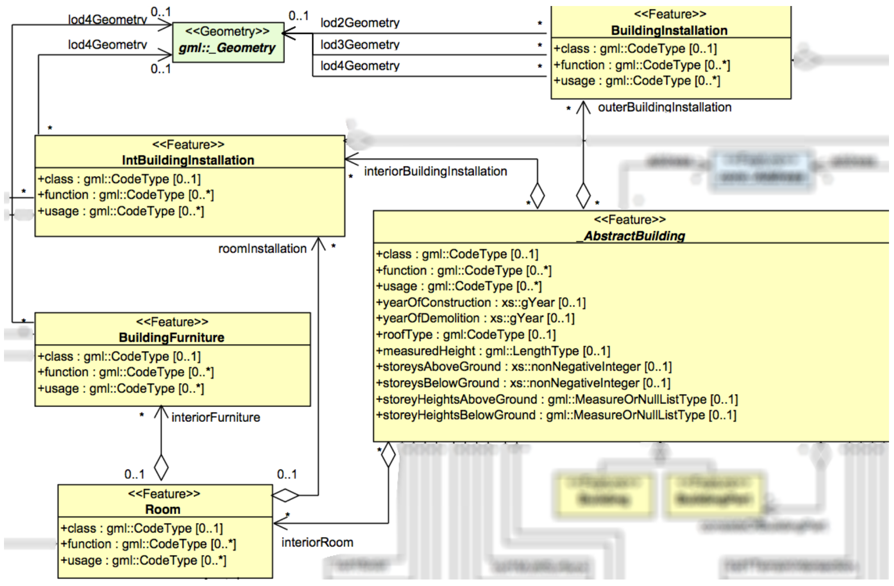


(b) Door as a solid

Figure 31: Different examples of geometries of Door Objects in CityGML

4.3.3 *Derivation of concept of vertical spaces*

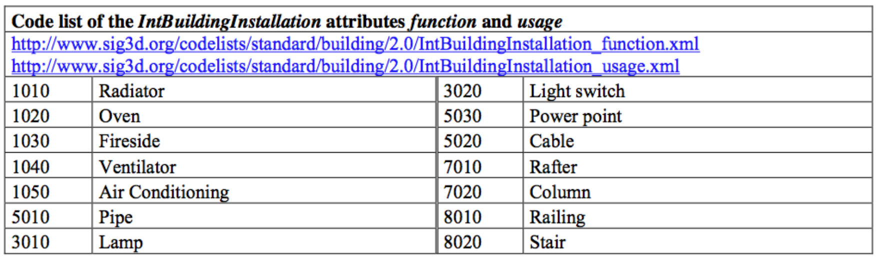
Unfortunately, there is no obvious way in CityGML for deriving information about vertical connectivity. Regarding the latter, we are referring to elevators, stairs, escalators, ramps. These semantic objects seem not to have a precise counterpart in CityGML. In fact, they are generically modelled as *IntBuildingInstallation*, and no specific tag for these objects is provided in schema definition. An *IntBuildingInstallation* is an object inside a building with a specialised function or semantic meaning. In contrast to *BuildingFurniture*, *IntBuildingInstallations* are permanently attached to the building structure and cannot be moved. Typical examples are interior stairs, railings, radiators or pipes. Objects of the class *IntBuildingInstallation* can either be associated with a room (class *Room*), or with the complete building / building part [13] (See Figure 32 to better understand the relationships that occur between the aforementioned classes). But as previously stated they are very generic so no specific function or usage can be evaluated to determine wheter they represent vertical connectors or not. A naive approach for determining if an *IntBuildingInstallation* is



lod4Geometry
0..1
<<Geometry>>
gml::_Geometry
lod4Geometry
0..1
0..1
lod2Geometry
lod3Geometry
lod4Geometry
<<Feature>>
BuildingInstallation
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
outerBuildingInstallation
<<Feature>>
IntBuildingInstallation
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
interiorBuildingInstallation
roomInstallation
<<Feature>>
_AbstractBuilding
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
+yearOfConstruction : xs::gYear [0..1]
+yearOfDemolition : xs::gYear [0..1]
+roofType : gml::CodeType [0..1]
+measuredHeight : gml::LengthType [0..1]
+storeysAboveGround : xs::nonNegativeInteger [0..1]
+storeysBelowGround : xs::nonNegativeInteger [0..1]
+storeyHeightsAboveGround : gml::MeasureOrNullListType [0..1]
+storeyHeightsBelowGround : gml::MeasureOrNullListType [0..1]
<<Feature>>
BuildingFurniture
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
interiorFurniture
0..1
0..1
<<Feature>>
Room
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
interiorRoom

Figure 32: CityGML Building Module classes representing interior installations and furniture

a vertical connector consists in analysing *function* and *usage* attributes. Indeed, nominal and real functions of a building installation can be described by those attributes. CityGml provides specification of code lists for enumerative attributes of type gml:CodeType and provides proposals for selected attributes. Usage is non-normative and the presented code lists are neither mandatory nor complete. As claimed in CityGML Specification [13], the governance of code lists is explicitly decoupled from the governance of the CityGML schema and specification. Thus, code lists can be freely defined outside the CityGML specification by any organization or information community according to their information needs. Hence, code lists mechanism provide extensibility and flexibility, but on the other hand, it cannot be assumed as a standard for deriving explicitly information about the function of a building installation. For illustrative, yet incomplete purposes, an example of code lists usage is provided in Listing 3. At line 6 it is possible to see an instantiation of "<bldg:function>" tag. As Figure 33 displays, code list value 8020 stands for *"Stair"*.



Code list of the *IntBuildingInstallation* attributes *function* and *usage*
http://www.sig3d.org/codelists/standard/building/2.0/IntBuildingInstallation_function.xml
http://www.sig3d.org/codelists/standard/building/2.0/IntBuildingInstallation_usage.xml

1010	Radiator	3020	Light switch
1020	Oven	5030	Power point
1030	Fireside	5020	Cable
1040	Ventilator	7010	Rafter
1050	Air Conditioning	7020	Column
5010	Pipe	8010	Railing
3010	Lamp	8020	Stair

Figure 33: CityGML Code list of the IntBuildingInstallation attributes function and usage

Listing 3: Excerpt of CityGML: usage of code lists for describing the function of an installation

```
1  /...
2  <bldg:roomInstallation>
3     <bldg:IntBuildingInstallation
4        gml:id="GML_8708e206-7f9e-4fe2-b09e-ff9aad59aae6">
5        <gml:name>Treppe KG</gml:name>
6        /<bldg:function>8020</bldg:function>
7        <bldg:lod4Geometry>
8           <gml:MultiSurface>
9              <gml:surfaceMember>
10                <gml:Polygon gml:id="PolyID5310_1989_169943_397327">
11                   <gml:exterior>
12                      ....
13                   </gml:exterior>
```

```
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:MultiSurface>
          </bldg:lod4Geometry>
        </bldg:IntBuildingInstallation>
      </bldg:roomInstallation>
```

Since semantics seem not to be helpful in the determination of vertical spaces, another approach could be geometry-based. The geometry of *IntBuildingInstallation* objects is represented, as can be noticed from Figure 34, in different surfaces that are aggregated and defined by *gml:Geometry* type to build the installation object similar to the building elements.

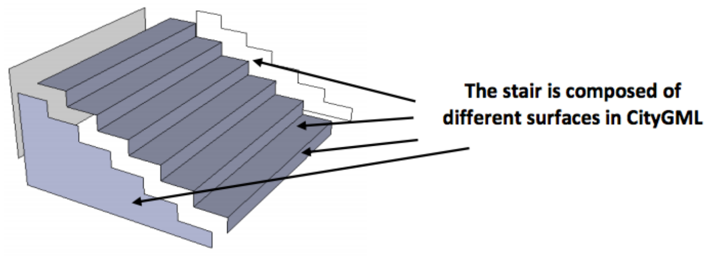


Figure 34: CityGML geometric representation of stairs. Image taken from [8]

A first attempt in geometric analysis is to verify if the *IntBuildingInstallation* spans over two or more *Rooms*. The same thematic object indeed would be referenced in each *Room* instantiation. But, as noticeable, this condition is only sufficient, and more investigation has to be done. A really interesting solution that could partially be reused has been proposed in [7]. This paper provides a machine learning approach based on Inductive Logic Programming (ILP) with the aim of learning grammar rules. As Figure 35 highlights, stairs are disaggregated into their primitives. Disaggregation occurs in a top-down manner, whereas learning is realised by a bottom-up approach, starting from the smallest atomic feature to the whole stairs object. The disaggregation and learning directions are indicated by descending and ascending arrows, respectively. Due to practical constraints, further investigations regarding the possibility of applying this ap-

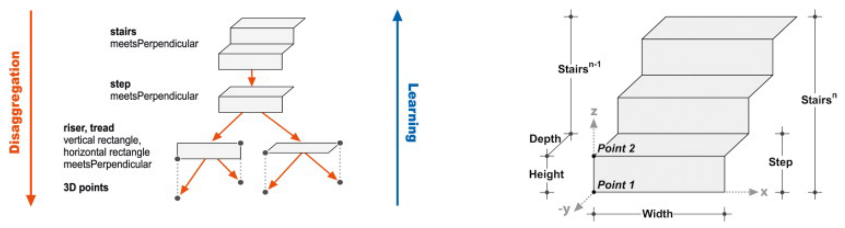


Figure 35: Incremental learning of semantic and topological primitives of stairs. Semantic primitives are marked in bold, whereas topological and geometric constraints are marked in normal font. Image taken from [7]

proach haven't been conducted since this topic goes beyond the scope of this research. Nevertheless, the reader should recall the fact that even though this approach could work in theory, practically it might be almost impossible: Figure 36 shows a detail of an *InternalBuildingInstallation*, composed by 15000 points and 2236 surfaces. Somehow noisy surfaces like the column and the railing should be removed in pattern recognition process and only surfaces that represent steps should be taken into consideration. But this is not an easy task to be implemented. Resuming, this dissertation has argued that still is almost infeasible to easily derive inormation about the vertical spaces in CityGML, by using both semantic and geometric approaches.

4.3.4 *Derivation of concept of non-navigable spaces*

As previously mentioned in the latter section, in CityGML only the main building structural objects are considered as building elements. Other spatial elements including some structural elements (e.g. columns, beams) are considered in CityGML as internal installations and represented by *IntBuildingInstallation* class. This class embodies only objects within a building which are permanently attached to the building structure (in contrast to furniture) and cannot be moved. Hence, every single instantiation of *IntBuildingInstallation* has been marked as a fixed obstacle.

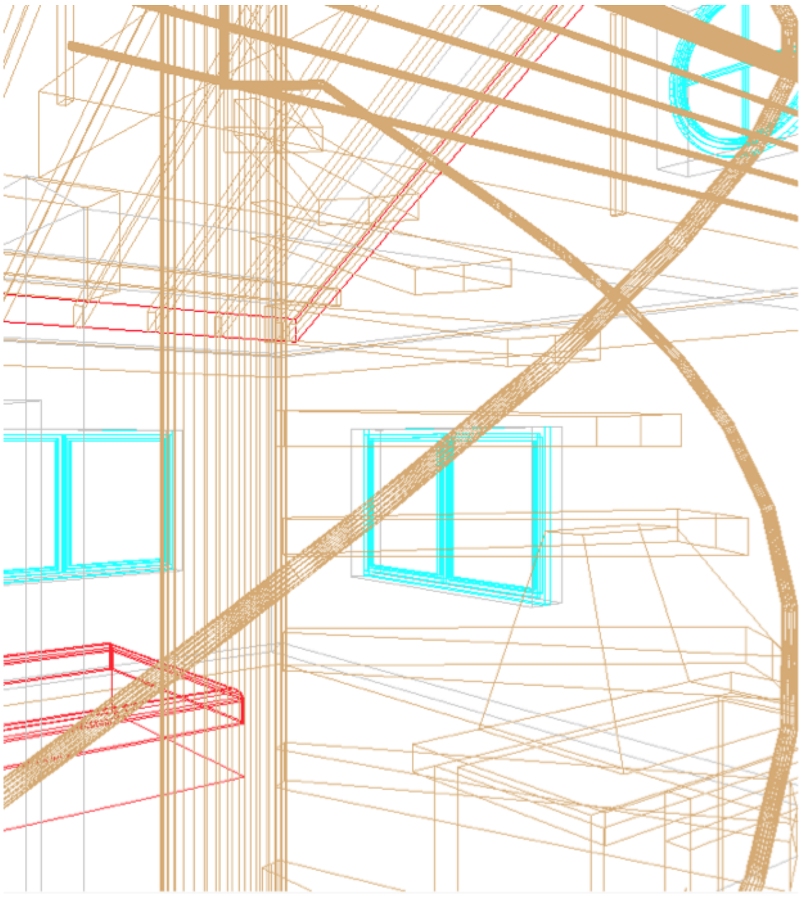
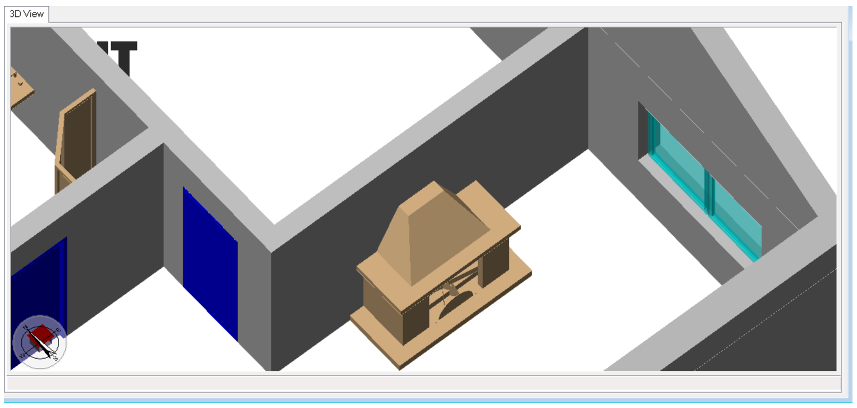


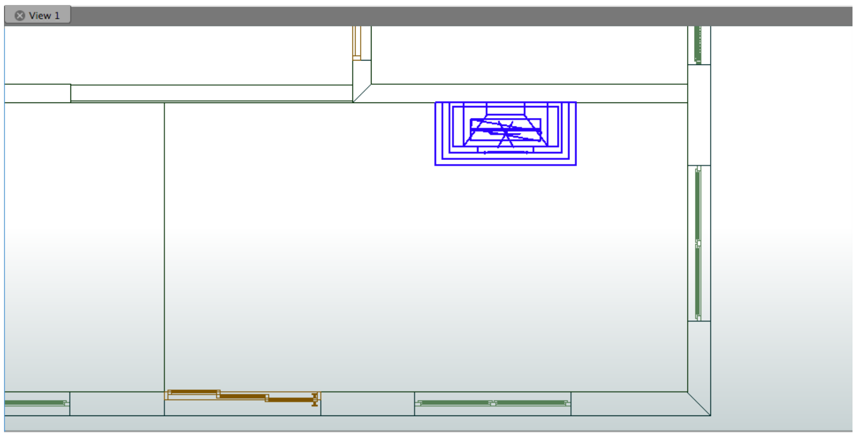
Figure 36: Detail of an *InternalBuildingInstallation* in CityGML

On the other hand, Room furniture, like tables and chairs, is represented in the CityGML building model with the class *BuildingFurniture*. The objects that appeal to this class, instead, have been marked as removable obstacles, since they may not necessarily obstruct navigation. But both of these elements could potentially contribute into the determination of obstacle-free paths: their footprints on 2D floorplan constitute the set of spaces where pedestrians cannot freely move. Figure 37 displays how fireplace installation has been subtracted from *Room* space in order to determine the actual navigable space.

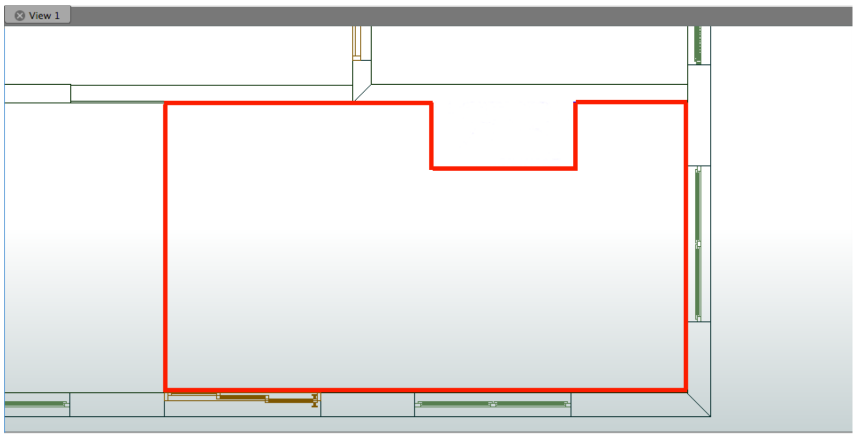
One of the more significant findings to emerge from this study is that CityGML has several potential capabilities in providing semantic, topological, geometric information for Indoor Navigation applications. Yet, the results of this investigation show that some limitations still exist: vertical connectors are not explicitly defined, the concept of storey is not existing, information retrieval of connectedness between spaces sometimes may result a complex task. For more implementation details regarding this portion of research, please refer to Section 7.2.



(a) A fireplace installation in a room



(b) 2D Footprints of the room and the fireplace



(c) The resulting 2D navigable space after obstacle removal

Figure 37: Obstacle removal by detecting IntBuildingInstallations in CityGML datasets

5 THE DEFINITION OF A DATA MODEL FOR INDOOR NAVIGATION

In Chapter 4 it has been inspected how CityGML could be beneficial for Indoor Navigation in terms of semantics, geometry, topology. The CityGML model might provide the topographic space of the indoor environment. Furthermore it presents geometric and topological relationships and certain semantics of the indoor environment like openings, installations, surfaces. It can potentially provide some of the necessary information for this kind of application. However, it is not a specific model for indoor navigation. A CityGML change request submitted by Hideki Hayashi of Hitachi Ltd. highlights the inability of this Model to answer navigation needs. The summary of change: "*Add a network topology model necessary to realize the indoor routing. The network topology model is used to indicate a connection between spaces or rooms, and it is also used to calculate migration pathway of humans, objects or robots*".

Hence, in this Chapter it will be presented a novel Data Model for Indoor Navigation, designed as an extension of Indoor Navigation Module of IndoorGML, an OGC candidate standard. This Data Model has been populated with information retrieved from CityGML. Why extending IndoorGML and not designing a new Data Model from scratch? Firstly because we do not want to reinvent the wheel, as this new and promising Data Model is capable of representing and allowing for exchange of geoinformation that is required to build and operate indoor navigation systems [31]. The model is intended to fill the lacks and the problems of existing approaches as they have been presented in Chapter 1, and where not compliant with our requirements it has been completed by the addition of new concepts. Secondly, we chose to extend IndoorGML to prove its flexibility, its modularization, and its capability of being adapted to different requirements as well.

In Section 5.1 some background information about IndoorGML will be provided, wheras in Section 5.2 it will be shown how IndoorGML has been extended.

5.1 INDOORGML BACKGROUND INFORMATION

Several standards such as CityGML, KML, and IFC have been published to describe 3D geometry and semantics of buildings not only for outdoor space but also indoor space, but they lack of important features, which are required by indoor navigation. This candidate standard aims to provide complementary features of indoor spatial information focused on indoor navigation. It is implemented as an application schema of the Geography Markup Language version 3.2.1. IndoorGML aims to establish a common schema for indoor navigation applications. It models topology and semantics of indoor spaces, which are needed for the components of navigation networks. However, it contains a minimum set of geometric and semantic modelling of construction components to avoid duplicated efforts with other standards, such as CityGML and IFC [31]. Participants in development of this standard include Pusan National University, University of Seoul, Technical University of Munich, Technical University of Berlin, Technical University of Delft, University of Nottingham, Hitachi Ltd., ETRI, Hyundai MNSoft Co., Applied Communication Sciences.

A rough classification of indoor spatial information infers two main categories:

- Management of building components and indoor facilities
- Usage of indoor space

Current Information Models like CityGML and IFC maninly concentrate on the first category: building construction and management, structural parts like roofs and walls are encompassed. Differently, the goal of IndoorGML is to define a framework of indoor spatial information to locate stationary or mobile features in indoor space and provide spatial information services referring their positions in indoor space, instead of representing building architectural components. In IndoorGML, the focus isn't on architectural components themselves but on the spaces (e.g. rooms, corridors, stairs) made by architectural components, where objects can be located and navigate, and the components (e.g. doors, lifts) to describe their properties and relationships between them. IndoorGML will provide the essential model and data for important applications like:

- building evacuation
- disaster management
- personal indoor navigation
- indoor robot navigation
- indoor spatial awareness
- indoor location based services
- the support for tracking people and goods

5.1.1 *Geometric representation of indoor space*

The geometric representation of 3D objects in indoor space does not belong to the major focus of IndoorGML, since it is extensively documented by ISO 19107, CityGML and IFC. However, IndoorGML adopts specifications described in ISO 19107. Euclidean Space (a subset of metric space) is used to model real world objects and the aforementioned B-Rep is used (see Figure 38). IndoorGML provides three possible options for geometric representation:

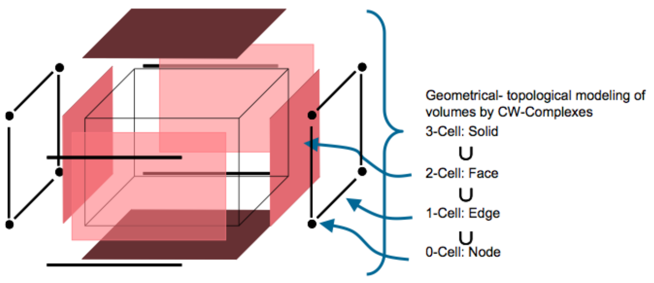

Geometrical- topological modeling of volumes by CW-Complexes
3-Cell: Solid
∪
2-Cell: Face
∪
1-Cell: Edge
∪
0-Cell: Node

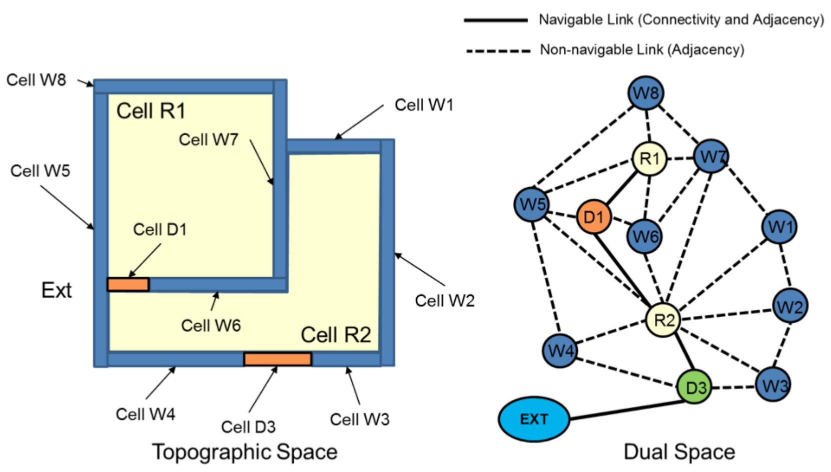
Figure 38: Representation of a 3D-Solid by B-Rep in Geometry and by CW-Complexes in Topology

1. IndoorGML documents contain references to the corresponding objects defined in external dataset of other standards such as CityGML or IFC
2. geometric representation is directly included in IndoorGML document as defined by ISO 19107 (gml::Solid for 3D or gml::Surface for 2D)
3. no geometric information is included

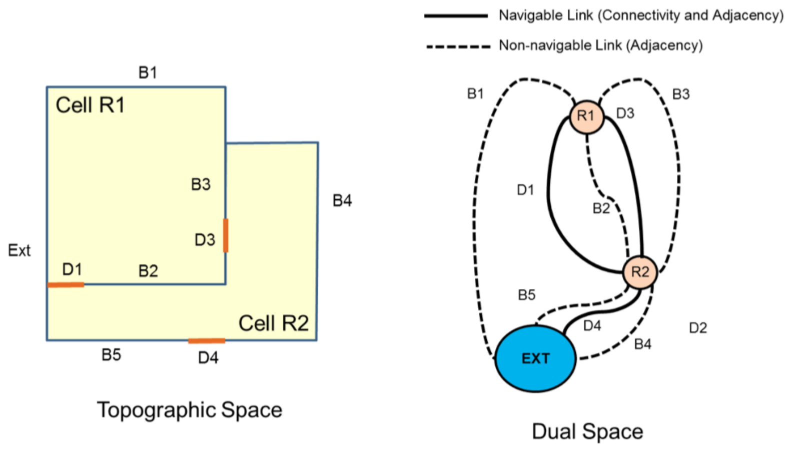
As shown, IndoorGML is very flexible regarding geometry, and it is also possible to choose between 3D geometry or 2D footprint ssimplification to describe geometric *features*.

5.1.2 *Topological representation of indoor space*

IndoorGML supports topology in order to simplify the complex spatial relationships between 3D objects. In fact it relies on Poincarè Duality for generating Node Relation Structure (as cited in Chapter 1). Solid 3D objects in primal space, e.g., rooms within a building, are mapped to vertices (0D) in dual space. The common 2D face shared by two solid objects is transformed into an edge (1D) linking two vertices in dual space. Figure 39 shows an example of Node Relation Structure, displaying how different network graphs can be produced if different topographic space representations are taken into account (thick-wall and paper-wall representations).



(a) Thick-wall representation



(b) Paper-wall representation

Figure 39: Graph Representation of Indoor Space

CONCEPT OF CELLULAR SPACE In semantic 3D building model, a space within building can be modeled as cellular space by non-overlapping cells such as room or corridors. In IndoorGML the so-called "*cells*" are treated as nodes and are stored in a graph structure. Cellular space is a kind of symbolic representation of space which provides qualitative human-readable descriptions about moving objects based on structural entities and/or points of interest (e.g., room or floor identifier, building name) and in contrast to geometric information it allows topological relations between entities. A cell represents the structurally smallest unit of the topographic space, in such a kind that the union of all cells located in a space arise again a space [18]. Cells may be of different size or shape.

Smaller partitions of the topographic space according to a semantic decomposition of objects (rooms) provide the necessary means for a more precise indoor route planning.

5.1.3 *Multilayered space event model (MLSEM)*

A key concept that characterizes IndoorGML is the Multilayered Space Event Model already introduced by Thomas Kolbe et al. in [4]. As previously mentioned in Chapter 1, the structured space model that allows for the distinct separation of primal space from dual space on the one hand, and geometry and topology on the other hand, defines the general layout of each space layer independent from the specific space model which it represents (see Figure 5). The concept of Structure Space model is further extended to *Multi-layered Space Model* (MLSM). MLSM provides an approach for combining differing indoor symbolic spaces to support full indoor navigation, as depicted in Figure 40. The vertical division corresponds to space representations within Euclidean space and topology space. The horizontal partitioning indicates primal and dual spaces. According to different semantic

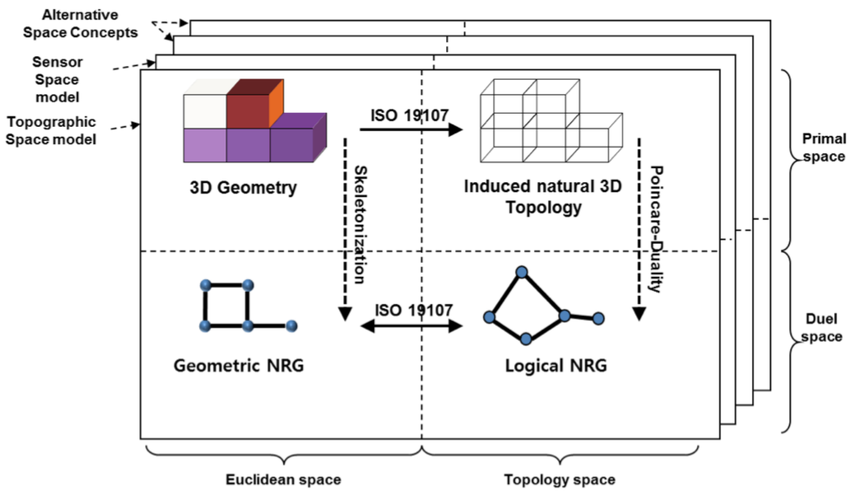
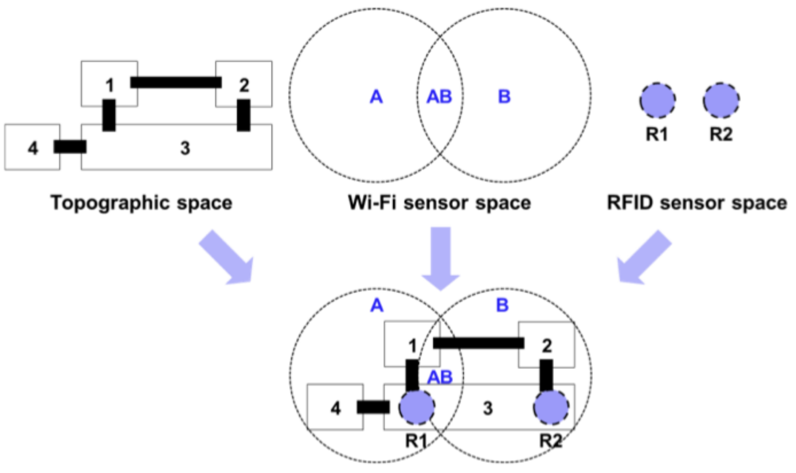
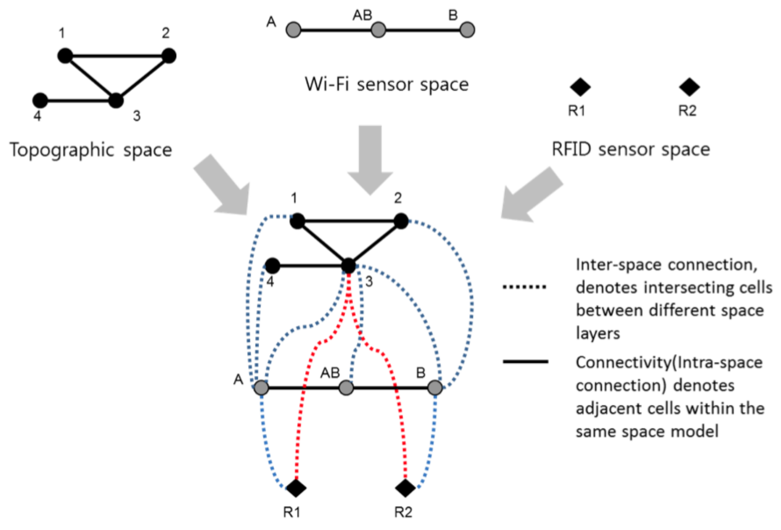


Figure 40: Multi-layered Space Model in IndoorGML

criteria, different semantic modelling may result in different decompositions of the same indoor space. Hence, each layer represents a space, e.g., topographic space or sensor space and so on. Layers of MLSM can be connected by inter-layer connections and combined into a MultiLayered Graph as Figure 41 illustrates. Recall that different space layers are only a different point of view of the same real space. The concepts introduced in MLSM contribute into the defini-



(a) Overlaid space layers in primal space


Topographic space
Wi-Fi sensor space
RFID sensor space
Inter-space connection, denotes intersecting cells between different space layers
Connectivity(Intra-space connection) denotes adjacent cells within the same space model

(b) Overlaid space layers in dual space

Figure 41: Overlaid space layers in primal and dual space

tion of the IndoorGML Core Module which will be presented in the next section.

5.1.4 *IndoorGML's Core Module*

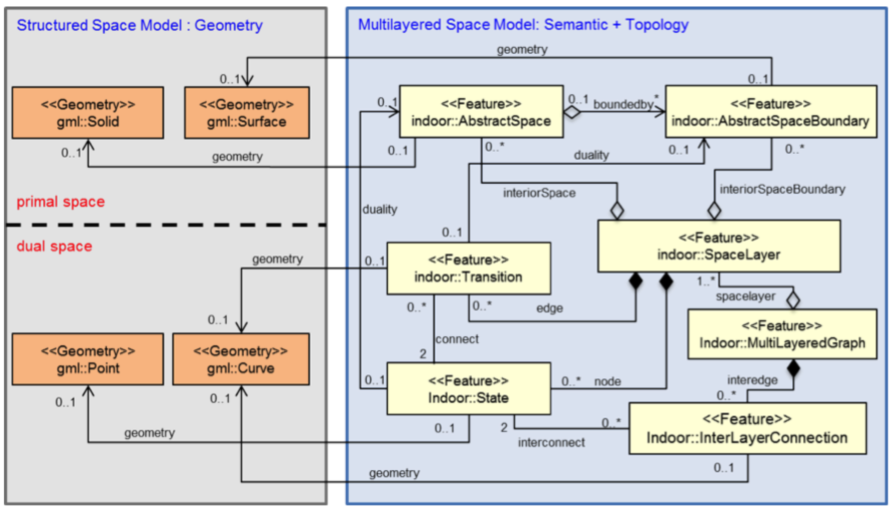

Structured Space Model : Geometry
Multilayered Space Model: Semantic + Topology
<<Geometry>> gml::Solid
<<Geometry>> gml::Surface
<<Geometry>> gml::Point
<<Geometry>> gml::Curve
primal space
dual space
<<Feature>> indoor::AbstractSpace
<<Feature>> indoor::AbstractSpaceBoundary
<<Feature>> indoor::SpaceLayer
<<Feature>> indoor::Transition
<<Feature>> Indoor::MultiLayeredGraph
<<Feature>> Indoor::State
<<Feature>> Indoor::InterLayerConnection
geometry
boundedby
duality
interiorSpace
interiorSpaceBoundary
spacelayer
edge
connect
node
interedge
interconnect

Figure 42: UML diagram of IndoorGML's Core Module

Figure 42 displays UML diagram of IndoorGML's Core Module. The classes *AbstractSpace* and *AbstractSpaceBoundary* represent topographic objects according to the concept of geographic *features* defined by ISO 19109. An *AbstractSpace* is a semantic class corresponding to one cell in Euclidean primal space of one layer. As a consequence, an *AbstractSpaceBoundary* is used to semantically describe the boundary of each cell. Those classes can be seen as an interface of the IndoorGML Core Module to the thematic objects that appear on 3D semantic models like CityGML (for instance, *AbstractSpace* could be mapped to *Room* wheras *AbstractSpaceBoundary* to *_BoundarySurface*). A *SpaceLayer* represents each separate layer of the MLSM and aggregates *States* and *Transitions*, which are dual representations of respectively *AbstractSpace* and *AbstractSpaceBoundary* objects in primal space (*State* is a node in dual space and can be associated to a room, corridor, door and is geometry consist in a *Point*. *Transition* is an edge that represents the adjacency or connectivity relationships among dual geometry spaces and connects two *States*).

In the next Section it will be firstly displayed the Indoor Navigation Module, secondly how the latter has been extended in order to fulfill our needs.

5.2 THE PROPOSED DATA MODEL: AN EXTENSION OF INDOORGML'S INDOOR NAVIGATION MODULE

IndoorGML provides a mechanism of modularization which splits the framework into a core module and some optional thematic extension modules which have a mandatory dependency on the core (see Figure 43: each extension package uses UML's "*import*" stereotype). Each extension module is specifically designed to cover a particu-

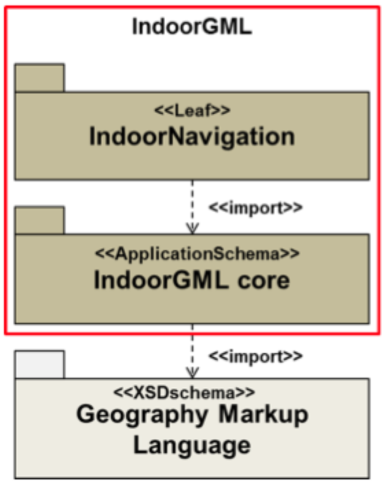


Figure 43: IndoorGML's extensibility mechanism displayed using an UML package diagram

lar application domain related to indoor navigation applications. IndoorGML already provides a default extension module called *Indoor Navigation Module*. On the basis of this package some modifications have been introduced in order to enrich this framework with some concepts that were missing from our perspective.

5.2.1 *IndoorGML's Indoor Navigation Module: how it has been extended*

Since the Core Module is very generic, the Indoor Navigation Module better refines the abstract concpepts of space like a detailed semantic classification of spaces, the concepts on communicating and visualizing navigable route sections, and the introduction of additional navigation constraints such as temporal access constraints as opening hours, or constraints resulting from material properties of the navigation path. Figure 45 displays UML diagram of Indoor Navigation Module, wheras Figure 46 displays the ExtendedIndoorGML, the proposed Data Module built on top of Indoor Navigation Module. In orange color the user-defined classes have been highlighted. In the next paragraph underlying concepts behind those UML classes will be discussed.

CLASSES OVERVIEW *AbstractSpace*: this is the base class for representing Indoor space, as it has been defined in the Core Module. It contains properties for space attributes and purely geometric representations of space, and like in CityGML, visible surfaces describing the shell of the space object can be represented by *AbstractSpaceBoundary* objects. No particular semantics of Indoors are mapped by this base class.

AbstractSpaceBoundary is used to semantically describe the boundary of each cell in topography space. The geometry normally will be described by a *Surface* in 3D, and by a *Curve* in 2D.

Building class, not officially present in Indoor Navigation Module, has been added in order to aggregate several *AbstractSpaces* as building parts. A *Building* infers the concept of *Storey*.

Storey, another class introduced in our extended model, is a part of a Building comprising all the rooms that are on the same level, that could be used by people (for living, work, storage, recreation, etc.). Each *Building* aggregates *Storeys* and each *AbstractSpace* has from zero to a maximum of two *Storeys* associated to it: a mezzanine, for instance, is typically a floor halfway between the current floor and the next higher floor. This kind of spaces typically spans over two storeys.

NavigableSpace class denotes a space that users can move freely in. It has two subclasses named *GeneralSpace* and *TransferSpace*. The subclasses are classified depending on the purpose of the space. This class has a recursive aggregation relationship to represent aggregated space. In fact, in some occasions either a semantic or a geometric decomposition of indoor space may be given in order to better classify space itself. Smaller partitions of topographic space and the corresponding semantic decomposition of room objects provide the necessary means for a more precise indoor route planning. The necessity of the decomposition of rooms into smaller units has been extensively addressed in [27], [4], and also in [31].
An extra recursive aggregation, not officially present in Indoor Navigation Module has been added to this class to represent topological relashionships between spaces such as connectivity/adjacency. In fact, in case topological information is provided from external sources, this cannot be stored into the model, and it has to be induced from geometric operations which are not computationally cost-effective, especially in 3D environment.

NavigableSpaceBoundary class extends *AbstractSpaceBoundary* and consists in a boundary representing connectivity relationships between cells in topography space. This class only models physical boundary as connections between real spaces (e.g. doors, openings, windows). To represent virtual boundary defined as connections between subspaces or openings in a building not filled by a doors/window another subclass has been introduced into the model not initally provided by IndoorGML, named *VirtualNavigableBoundary*

NonNavigableSpace is another user-defined class introduced to refine the concept of constraints that may limit navigation. This class represents obstacles that obstruct passages so that objects or subjects cannot move through easily. This class extends *AbstractSpace* and is refined by two subclasses that more specifically define the type of obstacle: *FixedNonNavigableSpace* and *RemovableNonNavigableSpace*. The former refers to objects that are permanently attached to the building structure and cannot be moved, while the latter refers to a movable part of a room, such as a chair or furniture. Those two classes can be easily mapped to CityGML's classes *IntBuildingInstallation* and *BuildingFurniture*. *NonNavigableSpace* is associated to the *NavigableSpace* that embeds it.
GeneralSpace class is one of the two subclasses of *NavigableSpace*. *Gen-*

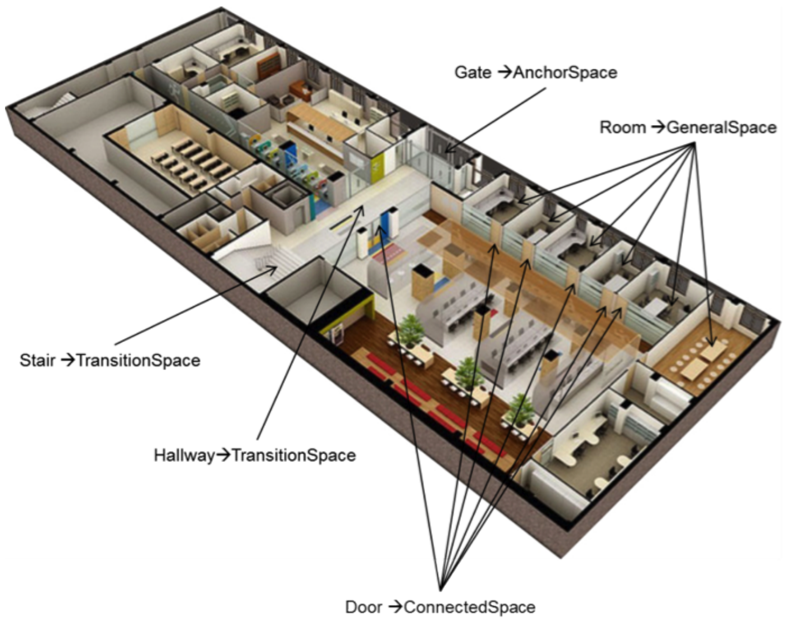


Figure 44: Indoor space as it has been modeled by IndoorGML

eralSpace class shall be used if users can move freely in the space and the space doesn't be used for transferring to other spaces in such as room, terrace, lobby, etc. In other words it is an endpoint space.

TransferSpace is derived from *NavigableSpace*. It is used to model a space for providing passages between *GeneralSpaces*. It has three subclasses: *ConnectedSpace AnchorSpace*, and *TransitionSpace*. Typically *TransferSpace* has more than one entrance/exit. See Figure 44 to better understand this classification of spaces.

ConnetedSpace represents an opening space that provides passages between two indoor spaces. It has been refined by two user-defined classes: *Door* and *Window*. Its CityGML counterpart is *_Opening* class.

AnchorSpace is the space connecting Indoors and Outdoors. It is mainly the entrance to the premises.

TransitionSpace models passages between two indoor spaces. This concept has been refined by the addition of a user-defined class called *VerticalTransitionSpace*.

VerticalTransitionSpace is a user defined class which refines and extends the concept of *TransitionSpace*. In fact, as the latter, is always a passage between two indoor spaces, but in which people can move in vertical directions (up and down, typically between two *Storeys*). It has also four subclasses: *Stairs, Ramp, Elevator, Escalator* that further specify the type of the space.

RouteNode represents a node inherited from State class. Within a dual graph structure is an *AbstractSpace* in primal space. This justifies the association between those two classes.

RouteSegments connects two *RouteNodes*, thus it represents a connectivity between spaces in primal space. It extends *Transition* class already defined in the Core Module.

Route represents a possible path to navigate indoor space. The Route aggregates *RouteNodes* and *RouteSegments*.

By comparing Figure 45 with Figure 46 it is possible to appreciate the differences between the original Indoor Module and the Extended Module. The latter semantically enriches the former, introducing new thematic concepts such as storeys, vertical spaces, non-navigable spaces. By these additions to the original Indoor Navigation Module a more appropriate binding with CityGML concepts is guaranteed, such that eventual imports from the latter to our internal model are done without information loss.

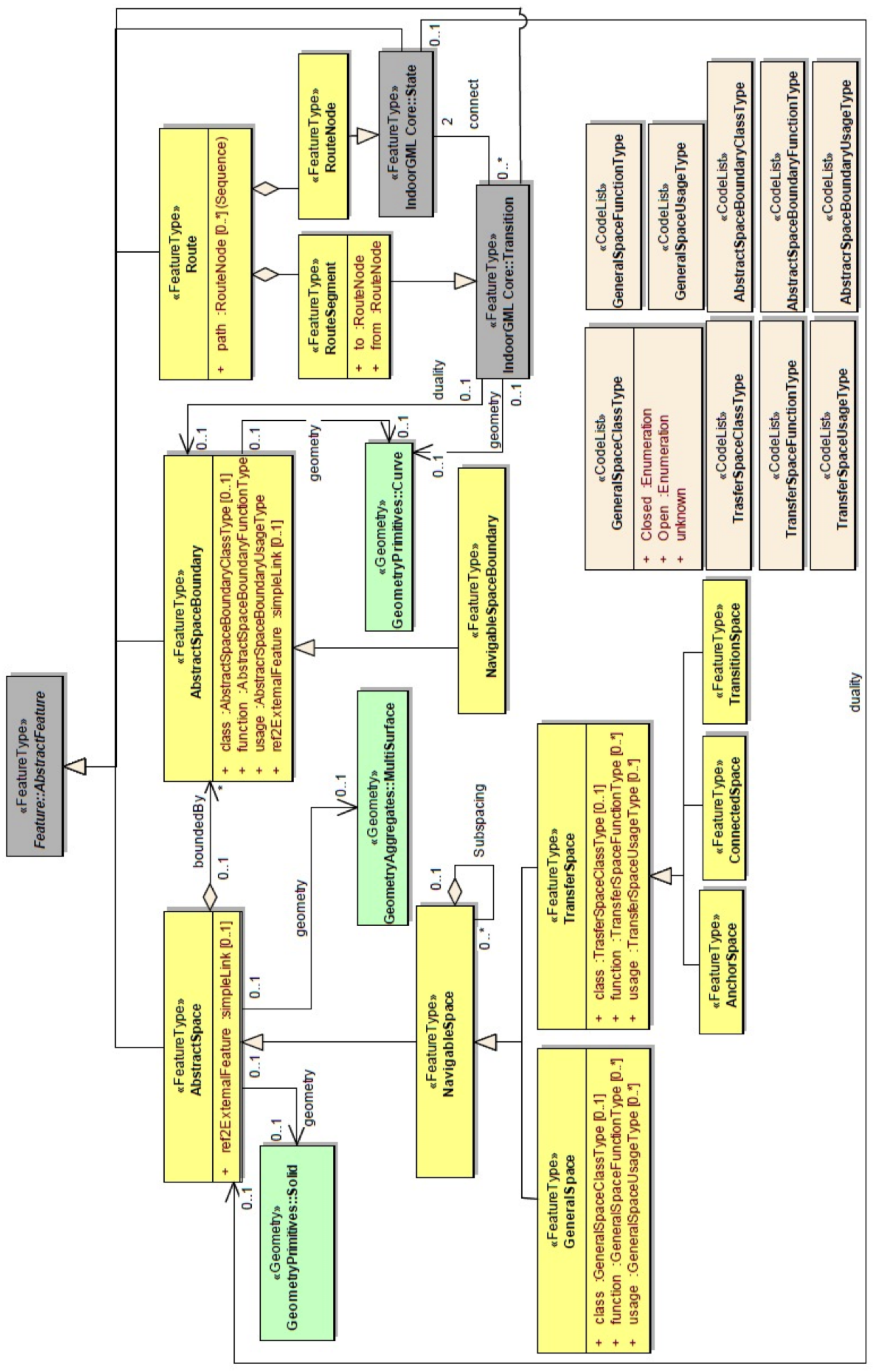

«FeatureType» Feature::AbstractFeature
«FeatureType» AbstractSpaceBoundary
+ class :AbstractSpaceBoundaryClassType [0..1]
+ function :AbstractSpaceBoundaryFunctionType
+ usage :AbstracrSpaceBoundaryUsageType
+ ref2ExternalFeature :simpleLink [0..1]
«FeatureType» Route
+ path :RouteNode [0..*] {Sequence}
«FeatureType» RouteNode
«FeatureType» RouteSegment
+ to :RouteNode
+ from :RouteNode
«FeatureType» IndoorGML Core::State
«FeatureType» IndoorGML Core::Transition
connect
2
0..*
0..1
duality
geometry
«Geometry» GeometryPrimitives::Curve
«FeatureType» NavigableSpaceBoundary
«Geometry» GeometryAggregates::MultiSurface
boundedBy
*
«FeatureType» AbstractSpace
+ ref2ExternalFeature :simpleLink [0..1]
«Geometry» GeometryPrimitives::Solid
«FeatureType» NavigableSpace
Subspacing
«FeatureType» TransferSpace
+ class :TrasferSpaceClassType [0..1]
+ function :TransferSpaceFunctionType [0..*]
+ usage :TransferSpaceUsageType [0..*]
«FeatureType» GeneralSpace
+ class :GeneralSpaceClassType [0..1]
+ function :GeneralSpaceFunctionType [0..*]
+ usage :GeneralSpaceUsageType [0..*]
«FeatureType» AnchorSpace
«FeatureType» ConnectedSpace
«FeatureType» TransitionSpace
«CodeList» GeneralSpaceClassType
+ Closed :Enumeration
+ Open :Enumeration
+ unknown
«CodeList» GeneralSpaceFunctionType
«CodeList» GeneralSpaceUsageType
«CodeList» TrasferSpaceClassType
«CodeList» TransferSpaceFunctionType
«CodeList» TransferSpaceUsageType
«CodeList» AbstractSpaceBoundaryClassType
«CodeList» AbstractSpaceBoundaryFunctionType
«CodeList» AbstracrSpaceBoundaryUsageType

Figure 45: UML Diagram of Indoor Navigation Module

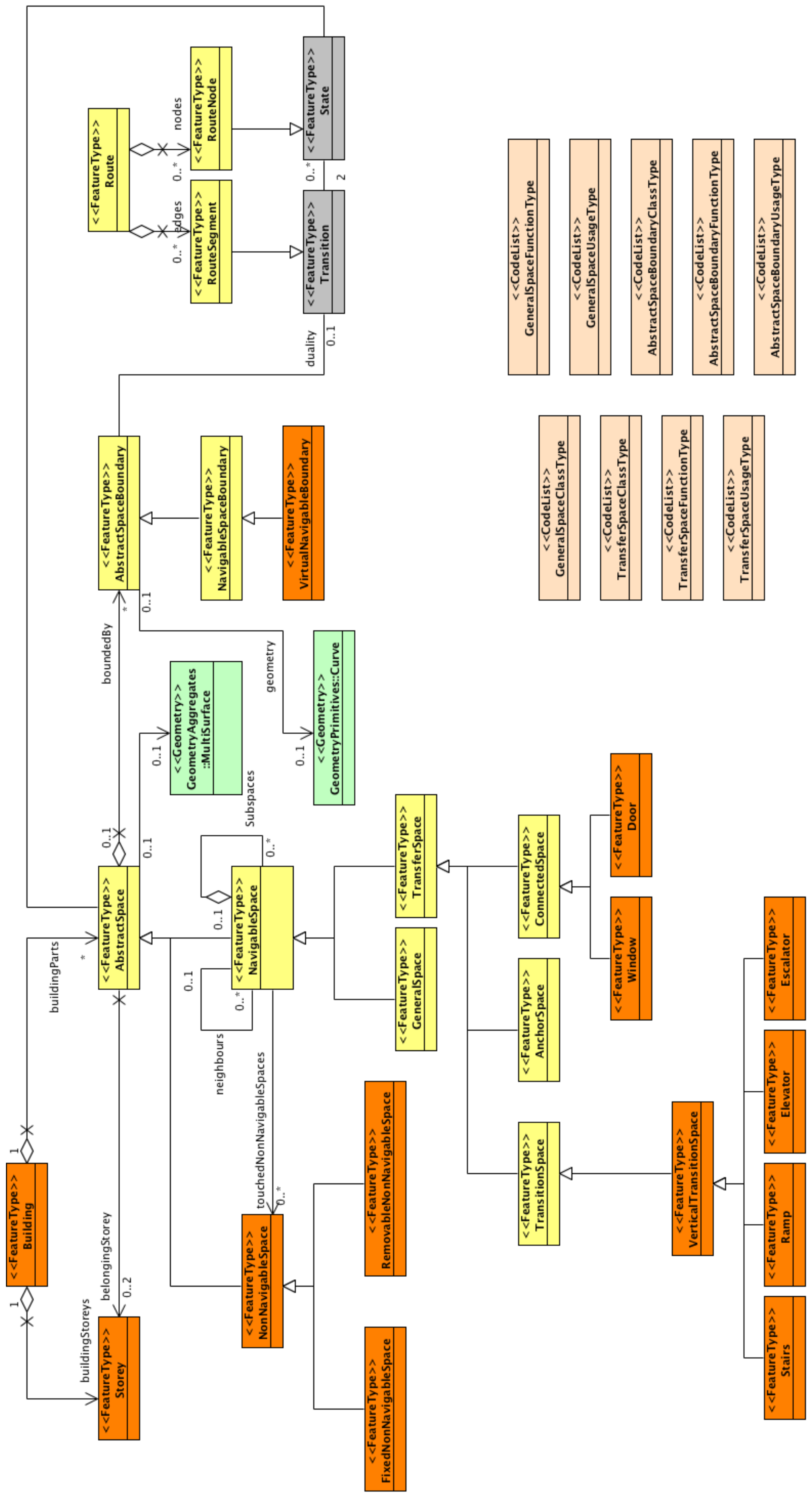

<<FeatureType>> Route
<<FeatureType>> RouteSegment
<<FeatureType>> RouteNode
<<FeatureType>> Transition
<<FeatureType>> State
edges
nodes
duality
0..*
0..1
2
<<FeatureType>> AbstractSpaceBoundary
<<FeatureType>> NavigableSpaceBoundary
<<FeatureType>> VirtualNavigableBoundary
<<CodeList>> GeneralSpaceFunctionType
<<CodeList>> GeneralSpaceUsageType
<<CodeList>> AbstractSpaceBoundaryClassType
<<CodeList>> AbstractSpaceBoundaryFunctionType
<<CodeList>> AbstractSpaceBoundaryUsageType
<<CodeList>> GeneralSpaceClassType
<<CodeList>> TransferSpaceClassType
<<CodeList>> TransferSpaceFunctionType
<<CodeList>> TransferSpaceUsageType
boundedBy
geometry
<<Geometry>> GeometryAggregates ::MultiSurface
<<Geometry>> GeometryPrimitives::Curve
Subspaces
<<FeatureType>> AbstractSpace
<<FeatureType>> NavigableSpace
<<FeatureType>> TransferSpace
<<FeatureType>> GeneralSpace
<<FeatureType>> ConnectedSpace
<<FeatureType>> AnchorSpace
<<FeatureType>> TransitionSpace
<<FeatureType>> Door
<<FeatureType>> Window
<<FeatureType>> VerticalTransitionSpace
<<FeatureType>> Escalator
<<FeatureType>> Elevator
<<FeatureType>> Ramp
<<FeatureType>> Stairs
buildingParts
neighbours
touchedNonNavigableSpaces
<<FeatureType>> RemovableNonNavigableSpace
<<FeatureType>> FixedNonNavigableSpace
<<FeatureType>> NonNavigableSpace
<<FeatureType>> Building
<<FeatureType>> Storey
buildingStoreys
belongingStorey
1
*
0..2

Figure 46: UML Diagram of ExtendedIndoorGML

6 THE AUTOMATIC NETWORK EXTRACTION PROCESS

In Chapter 4 it has been displayed that CityGML may potentially provide a part of or all of the necessary means for indoor navigation, since it holds information about the topographic space of the indoor environment, it presents geometric and topological relationships and certain semantics of the indoor environment. Furthermore, in Chapter 5 a Data Model obtained as an extension of IndoorGML's Indoor Navigation Module has been introduced. This model presents several analogies with CityGML LOD4 classes, so, once information has been retrieved from CityGML, it can be easily stored in a Model that, unlike CityGML, is specifically designed for Indoor Navigation. Now, in this Chapter, a strategy for automatically deriving a Network Graph will be discussed. The approach is geometry based, though it also relies on a finite amount of semantic information derived from the Data Model. To better understand the proposed solution, some preliminary background information will be given in the next Section. It refers to underlying theory behind geometric computation algorithms.

6.1 COMPUTATIONAL GEOMETRY BACKGROUND INFORMATION

6.1.1 *Delaunay Triangulation*

In mathematics and computational geometry, a Delaunay triangulation for a given set P of points in a plane is a triangulation DT(P) such that no point in P is inside the circumcircle of any triangle in DT(P) (see Figure 47). Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid skinny triangles. The triangulation is named after Boris Delaunay for his work on this topic from 1934. By considering cir-

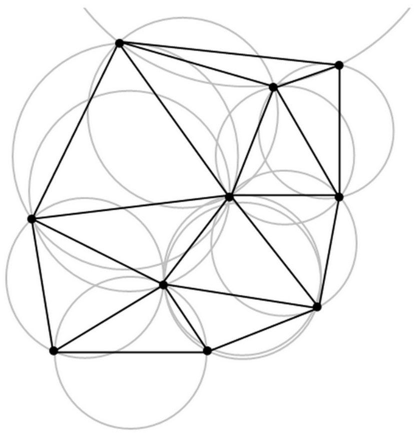


Figure 47: A Delaunay Triangulation in the plane with circumcircles shown

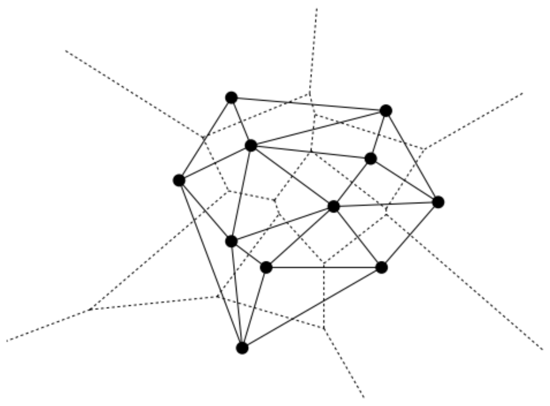


Figure 48: A Delaunay Triangulation as the dual graph of Voronoi Diagram

cumscribed spheres, the notion of Delaunay triangulation extends to three and higher dimensions. Generalizations are possible to metrics other than Euclidean. However in these cases a Delaunay triangulation is not guaranteed to exist or be unique. The Delaunay Triangulation is the dual graph of the Voronoi diagram (see Figure 48). It has a node for every Voronoi cell and it has an arc between two nodes if the corresponding cells share an edge.

6.1.2 *Constrained Delaunay Triangulation*

Given a set of n vertices in the plane together with a set of non-crossing, straight-line edges, the constrained Delaunay triangulation (CDT) is the triangulation of the vertices with the following properties: (1) the prespecified edges are included in the triangulation, and (2) it is as close as possible to the Delaunay triangulation [6]. As constrained edges are not necessarily Delaunay edges, the triangles of a constrained Delaunay triangulation do not necessarily fulfill the empty circle property but they fulfill a weaker constrained empty circle property. Figure 49 illustrates in a funny way a CDT (the picture is taken from https://code.google.com/p/poly2tri/, an open source C++ CDT implementation).

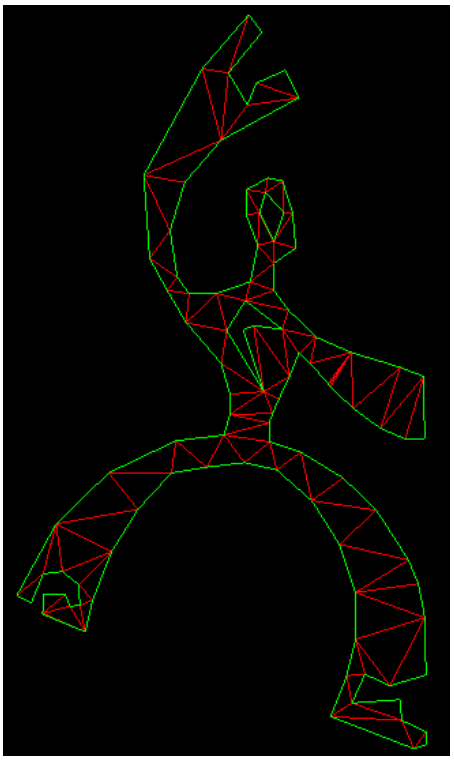


Figure 49: A Constrained Delaunay Triangulation: green lines are the constrained edges

6.1.3 *Medial Axis Transform*

The medial axis of an object is the set of all points having more than one closest point on the object's boundary. In 2D, the medial axis of a plane curve S is the locus of the centers of circles that are tangent to curve S in two or more points, where all such circles are contained in S. (It follows that the medial axis itself is contained in S). It is a representation often used in navigation applications, as it can generate valid and natural appearing paths with maximal clearance from all obstacles. The medial axis together with the associated radius function of the maximally inscribed discs is called the medial axis transform (MAT). The medial axis transform is a complete shape descriptor (see also shape analysis), meaning that it can be used to reconstruct the shape of the original domain.

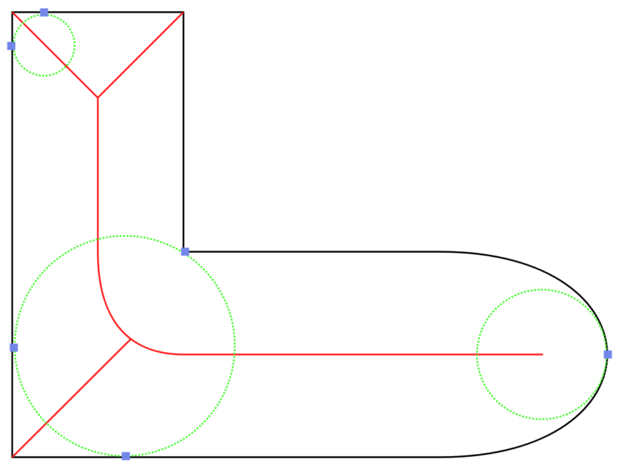


Figure 50: Medial Axis Transform

6.1.4 *Visibility Graph*

A visibility graph is a graph of intervisible locations, typically for a set of points and obstacles in the Euclidean plane. Each node in the graph represents a point location, and each edge represents a visible connection between them. That is, if the line segment connecting two locations does not pass through any obstacle, an edge is drawn between them in the graph. Visibility graphs have many applications, including finding the shortest path (see Figure 51), robotic motion planning and the art-gallery problem.

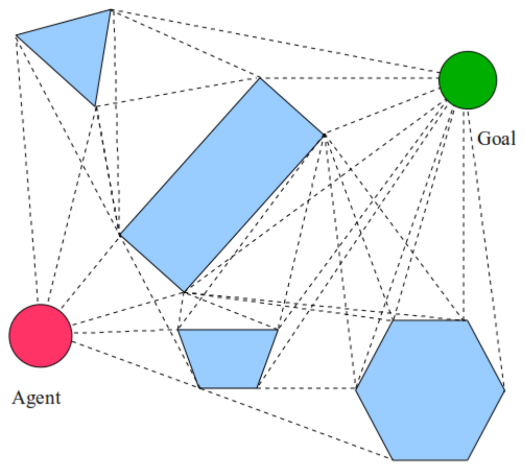


Figure 51: A visibility Graph built to detect shortest obstacle-free path between a source and a target

6.1.5 *Inward Polygon Offsetting (Inset)*

A polygon offset is a computational geometry primitive that given a polygon will trace a inner or outer version of it that it is either totally contained inside the polygon or that it will contain it entirely whose perimeter will be at a constant distance from the perimeter of the initial polygon. For any 2D non-degenerate strictly-simple polygon with holes called the source, there can exist a set of 0, 1 or more inward offset polygons with holes, or just offset polygons for short, at some euclidean distance $t>0$ (each being strictly simple and non-degenerate). Any contour edge of such offset polygon, called an offset edge corresponds to some contour edge of the source polygon, called its source edge. An offset edge is parallel to its source edge and has the same orientation. The Euclidean distance between the lines supporting an offset edge and its source edge is exactly t. Topological changes might occur, as some edges might contract until they vanish. Figure 52

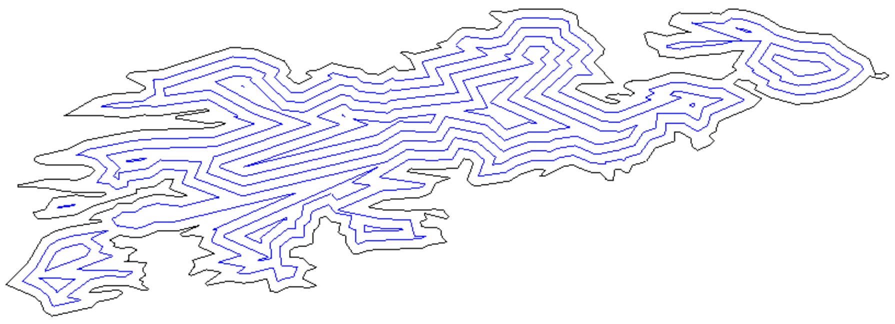


Figure 52: Offset contours of a sample polygon

6.2 INTRODUCTION TO THE PROBLEM OF AUTOMATIC NETWORK GENERATION

As already discussed in Chapter 1 and in Chapter 2 poor references have been found from current literature regarding approaches that lead to the derivation of 3D navigation model from 3D digital model of complex buildings. More specifically, automatic derivation of network from complex 3D building models is one of the most critical obstacles presently.

The most considerable approaches to the problem of generation of a geometric-topological graph that represents the reachability inside a building from our perspective are two: 1) Straight Medial Axis Transformation (S-MAT), a skeleton abstraction algorithm, which can abstract linear features from simple polygons, based on medial axis and designed by Lee [23]; 2) A "Door-to-Door" path finding approach for Indoor Navigation, by Liu and Zlatanova [25], [26], an algorithm based on Visibility Graph construction.

Both of the approaches are hybrid, in the sense that they consist in a combination of geometry and topology approaches. Geometric models can efficiently integrate metric properties to provide highly accurate location and distance information—necessary elements in most of context-aware applications, whereas topological models maintain a more abstract view of space by providing users with easily-recognizable information and by materializing more complex relationships between entities [1]. Nevertheless, they are both 2D-geometry based, but they both produce a 3D Geometric Network Graph. Let us illustrate the distinctive features of them.

6.2.1 *Straight-Medial Axis Transform*

S-MAT is a centerline based topology network which can abstract linear features from simple polygons. It actually has been originally developed by Yao and Rokne in 1991 for computational geometry and modified by Lee in 2004 for creating 3D topology network using CAD files for geospatial analysis in an urban environment (see Figure 53). Indeed, it has its own benefits:

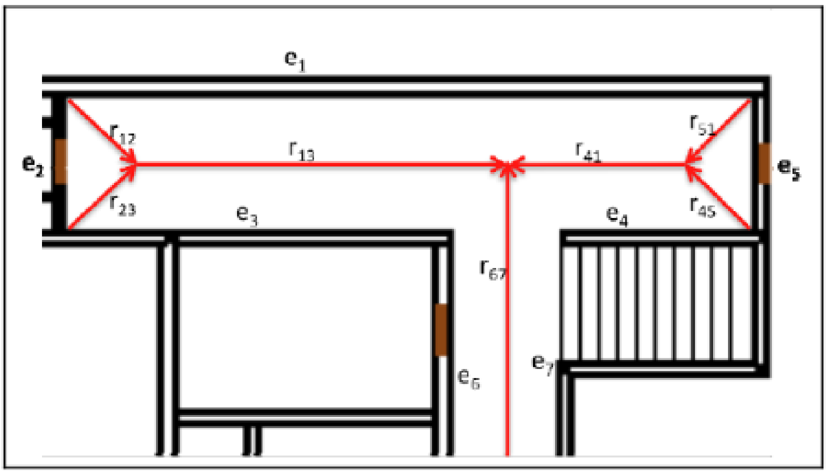


Figure 53: An example of S-MAT of a polygonal hallway

- it can be generated automatically
- indoor route calculation can be conducted readily and fast
- the resulting Geometric Network Graph could support dynamic changes (insert or delete edges quickly) as remarked in [30]

- a centerline approach results to be a good representation of natural human behaviour within indoor environment. Valid paths are represented concisely, especially in presence of obstacles.

Despite of these benefits, there are also some drawbacks:

- it may not represent accessibility within buildings accurately. For instance, it may create weird networks for corridors. It fails for large arbitrary shaped spaces. Figure 54 shows S-MAT of the hallway of OTB (TUDelft) building. As the space gets wider, a detour is generated, paths are distorted towards the middle of the open space. Hence the medial axis does not represent a typical path taken by a person. It is evident that these geometrical paths are not suitable for direct translation into route directions.

- Since the algorithm involves constructing bisectors of only convex vertices, whenever there is an intersection at a concave vertex, a new bisector does not emerge from that intersection point, as visible in Figure 55

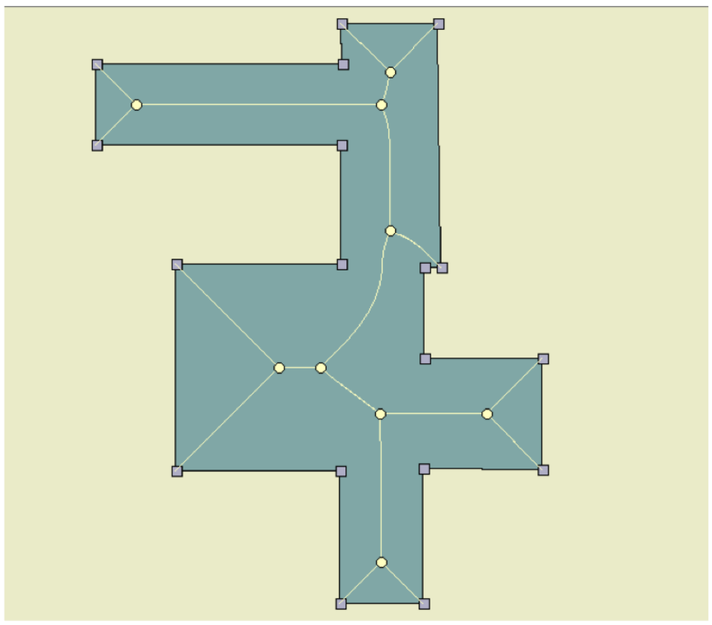


Figure 54: S-MAT of the hallway of OTB (TUDelft) building. A weird network with some detours is displayed

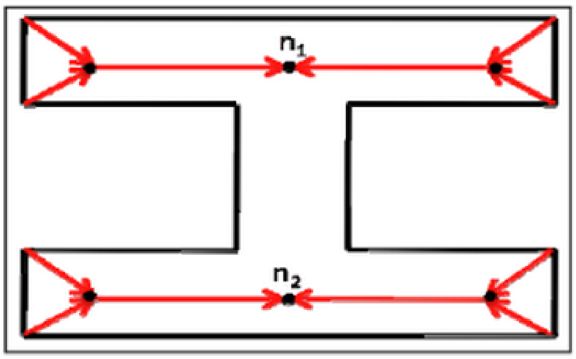


Figure 55: S-MAT deficiencies. The algorithm gets stuck after reaching nodes n_1 and n_2

6.2.2 *Door to Door path finding*

Door-to-door approach is interpreted as the direct walking way from a door to the next visible door or the shortest possible way between two invisible doors. The routing strategy is organized as a two-level approach: 1) coarse: it is based on room-to-room connectivity on the floor level to determine the direction of movement. 2) refined: it is used in a single closed space to avoid all kind of obstructions and make people transfer in a door-to-door way. Even though this strategy is new, it is substantially based on Visibility Graph, which is an established tool for robot path planning. Nodes represent the convex corners of a region, including the corners of all obstacles, and additionally all portals/entries to the region. Two nodes are considered as mutually visible if their direct connection does not intersect any of the region's boundaries (see Figure 56). The advantages of this approach are:

- it can be automatically constructed
- It provides optimal non circuitous routes

But, also in this case some drawbacks have to be taken into account:

- route descriptions may result very complex to be interpreted by humans. We find evident difficulties in following instructions

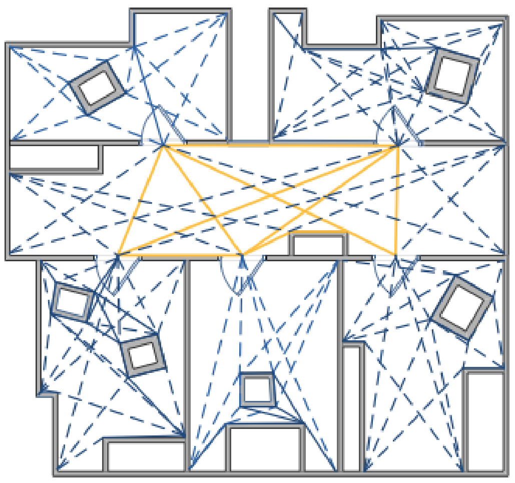


Figure 56: Door to door path finding

like "the next door is at an angle of 134 degrees and a distance of 9.2 metres"

- the amount of space required for storing nodes and edges: primarily in convex spaces with no obstacles, the number of possible combinations is quadratic (please refer to Figure 57 to get a holistic idea). As discussed in [35], unlike the centerline-based approaches, where there is only one consistent network for the whole building, in visibility-based networks there are numerous networks that are possible, based on the different routes a person can take. Creating this network as needed, using a lightweight mobile device, can prove to be a significant challenge.

- Since the intermediate points of a path between a source and a target are only the concave corners, the tracking of a subject or object is not efficient.

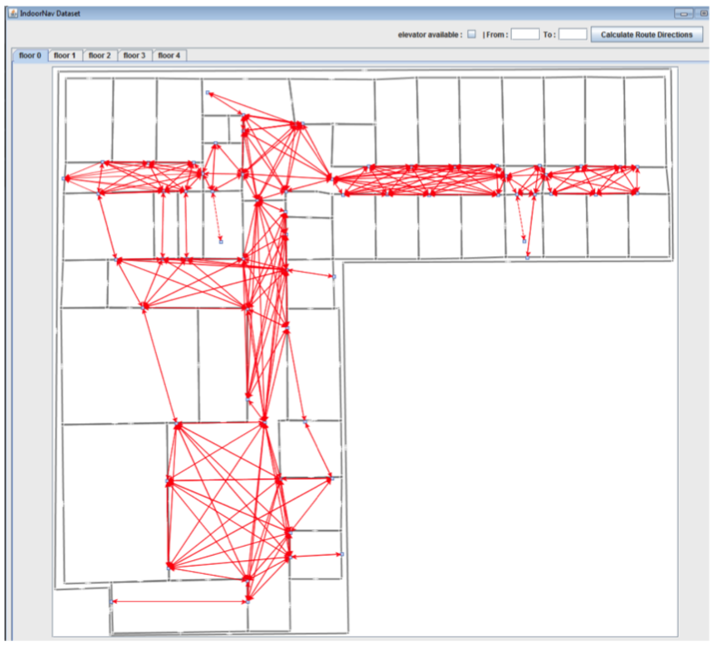


Figure 57: VG of the first floor of OTB (TUDelft) building. As the screenshot suggests, the amount of edges is considerable

6.2.3 *Why introducing a novel strategy: the requirements*

As it has been shown in the previous Sections, these two major approaches have their own benefits and peculiarities. Besides the valuable merits some limitations have been illustrated, for both of the algorithms. We would like to introduce a novel approach which aims to encompass the positive features of both S-MAT and Door-To-Door on one hand, on the other hand tries to overcome the problems these two strategies present. The requirements (partially inspired by [35]) of the new Improved Geometric Network Model are listed as follows:

- Improved Geometric Network Model should be a dimensionally weighted topology network of connected spaces in indoor environments so as to accurately represent indoor route lenghts.

- The Improved indoor network should be a network-based representation of indoor navigable routes that results from decomposition of planar polygons such as hallways.

- Space centerlines are an appropriate abstraction for indoor navigation networks. The network should avoid as much as possible detours and circuitous routes.
- Network generation should result in an auotmatic process.
- Network generation should preserve duality relashionship between primal space (i.e. topographic space) and dual space (topology). Recall that this property is not maintained neither in S-MAT, nor in Door-to-Door. In fact both of the strategies do not map directly nodes and edges in dual space with topographic space in primal space
- As a consequence of the previous requirement, the tracking of a subject or object should be easy and accurate
- Improved Geometric Network Model should be semantically-rich (in the sense that should encompass semantics like information about doors, windows, etc..)

In the next Section our approach will be illustrated.

6.3 GENERATION OF IMPROVED GEOMETRIC NETWORK MODEL

The proposed solution consists in a network-generation strategy organized in a hierarchical two-level way, similarly to the approach proposed in [25]:

- firstly a coarse-grained research can be implemented: this task involves the construction of a weighted graph that only encompasses inter-space connectivity. In other words we are referring to the aforementioned Node Relation Structure. Graph construction can be done readily and fast, according to Extended Indoor Navigation Data Model introduced in Chapter 5. In fact, if the Model is populated with correct and consistent information, it can be interrogated in such way: each *AbstractSpace* is associated to its dual representation, *State* class. The latter, for Poincarè duality, is the centroid of the topographic space. Adjacency of spaces can be semantically assessed by inspecting the existence of the recursive relationship which aggregates a *NavigableSpace* to its neighbours. Edges in dual representation are placed between adjacent spaces, and weights are calculated as the distance between space centroids. Once the weighted graph is constructed, a shortest path algorithm [1] can be launched on top of it to determine an optimal sequence of spaces between a source and a target (see Figure 58).

- once the inter-space path has been found, intra-space path (i.e. physical path) has to be computed. This is done by means of geometric operations that will be illustrated in Section 6.3.1.

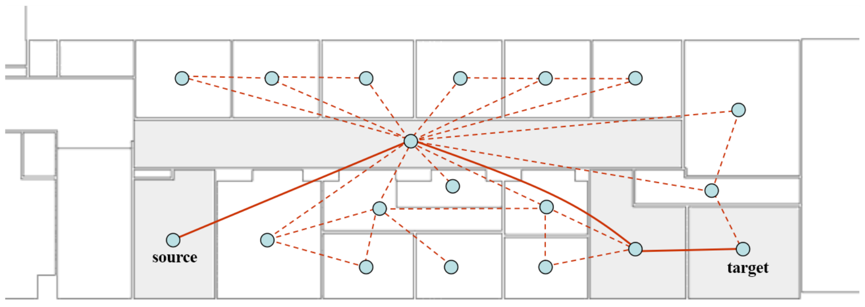


Figure 58: NRS for a coarse path finding

6.3.1 *Intra-space connectivity: the construction of physical path*

In this section the algorithm that has been designed in order to construct the physical path will be presented. The strategy is local, in the sense that path construction takes place incrementally room by room along the sequence of spaces previously identified by the first step of the proposed solution. The algorithm is composed of several 2D geometric operations that will be illustrated one by one. Even though

1 Dijkstra's shortest path algorithm is the most popular.

the approach is 2D based, the generated network graph will result in a 3D hierarchical network structure like the one proposed in [23]. For clarity, we'll refer to what happens only in one room, since this procedure will be iteratively repeated for the other rooms composing the inter-space optimal sequence.

INWARD OFFSETTING Let us concentrate on the main hallway located in the ground floor of OTB Building, within the TUDelft Campus, highlited with yellow dots in Figure 59. The first geometric operation to be computed on the 2D polygon that represents the footprints of the hallway is the INWARD OFFSETTING as it has been described in Section 6.1.5. This operation is essential in detecting large, open spaces that are clearly the weak spot of Medial Axis Transform and will help in overcoming the aforementioned problem of the distortion of the path towards the middle of the open space. Recall that there can exist a set of 0, 1 or more inward offset polygons, depending on space configuration. The parameter $t>0$ that defines the euclidean distance at which the *inset* polygon will be constructed can be tuned according to some user-defined criteria. As mentioned in [10], from a geometric point of view, corridors are a special type of rectangular rooms in which two opposing sides are short (i.e., just a few meters) and the other two sides are way much longer (i.e., $shortSide \ll longSide$). After some testing, we came to the conclusion that a good compromise for parameter t is the lenght of the short side of the corridor, because it generates a sort of virtual navigable corridors around the *inset* polygon itself. As a consequence, natural human movement is preserved in the proximity of the boundaries of a large space. Unfortunately this value, except from the trivial case of a rectangularly shaped corridor, cannot be extracted automatically and has to be manually assigned as an input value of the algorithm. A good choice of the parameter t is essential. Basically this value proves to be constant for the whole Building, but for some configurations has to be modified according to the size and the shape of the space. Figure 60 illustrates the results.

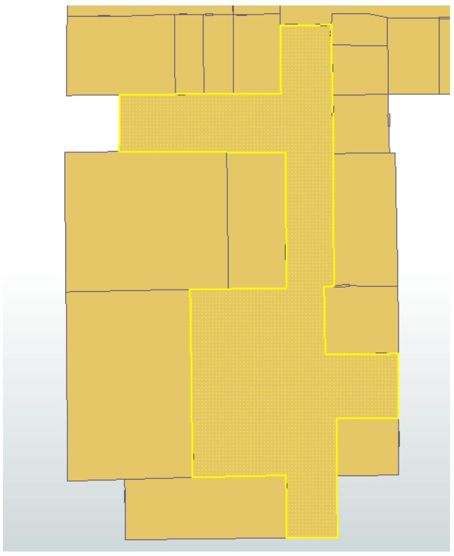


Figure 59: OTB Building Ground Floor Main Hall

SAMPLING OF THE BOUNDARY OF THE POLYGONS Once inward offsetting has been computed, a boundary sampling process of both inset polygons and originating polygon can be implemented. This process is legitimized by two motivations: mainly because we want to encompass semantic objects like doors, windows or more generally speaking openings. Secondly, because a fine grained partitioning of indoor navigable space can significantly contribute in facilitating the tracking of objects within the environment. Moreover the sampling regularizes the shape of the Constrained Delaunay Triangulation. As it will be displayed later, a more regular Triangulation generates a more regular network (in the sense that no detours or strange unexpected behaviour occur). The result of this process is clearly visible in Figure 61. The sampling ratio is once again based on user-defined criteria. Also in this case a good parameter is the lenght of the short side of the corridor.

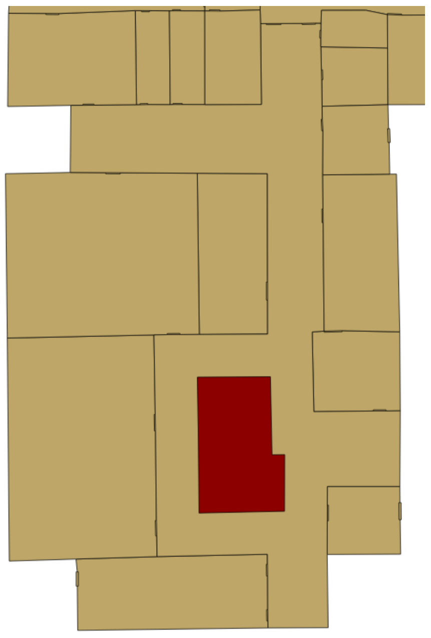


Figure 60: Computation of Inward Offset applied to the 2d footprints of OTB ground floor hallway

CONSTRAINED DELAUNAY TRIANGULATION Reached this point, a Constrained Delaunay Triangulation can be computed. The CDT takes as input the point-set obtained in the previous step. The closed chain of edges defining the polygons is meant to be the constraint of the triangulation. As Figure 62 suggests, facets that fall outside the domain delimited by the polygons have to be discarded (see Figure 63). As it will be outlined later the Constrained Delaunay Triangulation can generate a centerline-based path, similarly to MAT. Nevertheless, the introduction of the inward offset polygons can overcome the problems that MAT generates in large open spaces.

SUBSPACE FACETS THAT HAVE NO CONSTRAINED EDGES After computing CDT and having limited it to the domain identified by polygons, an additional geometric operation has been performed: each facet that has no constrained edges has been further subdivided into three smaller triangles. The subdivision has been made by placing an additional point located on the centroid of the facet (see Figure 64). We named this kind of facet *"Crossing"* since, as it is visible

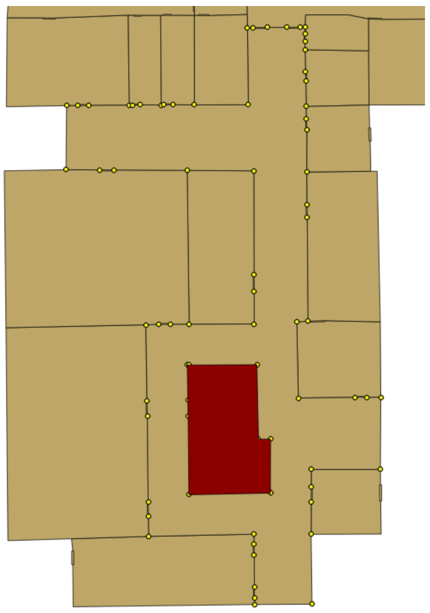


Figure 61: Sampling of polygons' boundaries

from Figure 67, it represents a space in which possible route alternatives occur. This operation has been adopted in order to suppress unwanted detours on the path.

NODES DISPLACEMENT Finally, nodes that represent topographic space in dual representation can be assigned to the spaces they belong to. Points are placed at the midpoint of all contour edges of the generated triangles that fall inside facets with no constrained edges (see Figure 65a). For facets having only one constrained edge, a node is placed on the midpoint of the segment (see Figure 65b) connecting the edges passing through their relative midpoints. A node is also assigned to the space that appeals to the polygon generated by inward offsetting. Regarding the latter, two mutually exclusive conditions may occur:

- the centroid of the polygon falls inside the contour
- the centroid of the polygon falls outside the contour

For the first case, the node can be positioned in the centroid. For the second case, a convex decomposition has to be computed and for

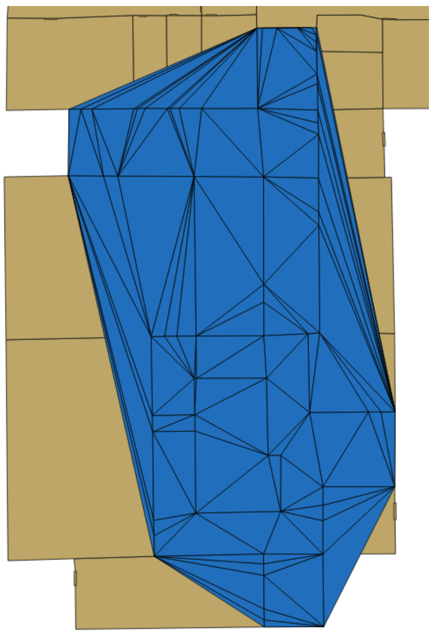


Figure 62: Constrained Delaunay Triangulation computed on the point-set obtained in the previous step. Constraints consist in the edges that constitute the boundary of the polygons.

each partition a node is assigned, located in the centroid of the partition itself (this latter case hasn't been yet implemented). Furthermore, each node has been marked with particular semantics, based on the type of the topographic space it represents: it might be a *DoorAdjacentNode*, a *DeadEndNode* (a node representing a facet with two constrained edges, i.e. walls), a *HoleAdjacentNode* (a node whose topographic space touches the contour of the inset polygon), *Crossing* (a node whose topographic space –facet– has no constrained edge), and an *ObstacleAdjacentNode*. Figure 66 depicts the nodes displaced within the whole environment.

LINK NODES The previous steps contributed to identify and displace the nodes that will populate the Improved Geometric Network graph. Hence, the last operation consists in linking the nodes to each other based on adjacency of the topographic space they represent in primal space. This process has its own peculiarities: *DeadEndNodes* are skipped since no one wants to navigate to the corner of a room;

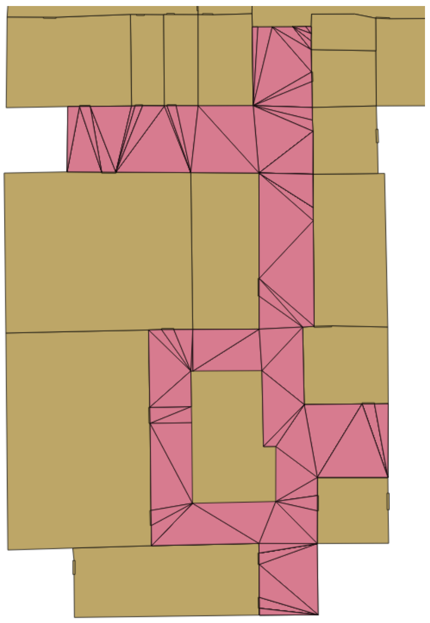


Figure 63: Constrained Delaunay Triangulation limited to the polygonal domain.

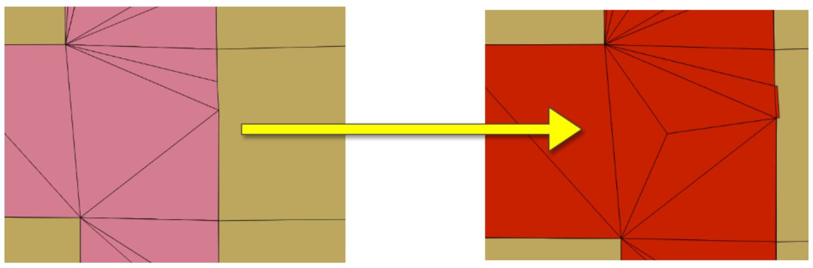
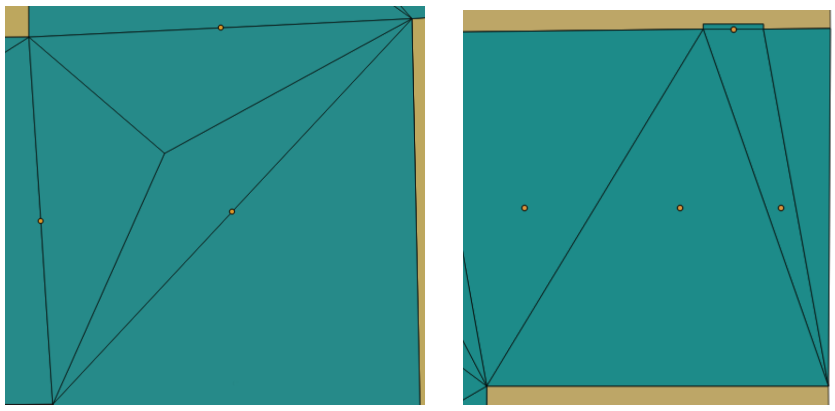


Figure 64: Triangles with no constrained edges additionally subdivided.

moreover, some specific nodes (*DoorAdjacentNodes* and *Crossings* that are also *HoleAdjacentNodes*) have been linked to the node(s) representing inset polygons. As visible from Figure 67, the introduced *inset* polygon determines a "door-to-door" alike network that connects intervisible locations. Edges of the dual graph represent adjacency and connectivity relationships between the cells of topographic space. The graph is not only topological but also geometric: nodes have been georeferenced in a 3D context, whilst edges contain metric information (i.e. euclidean distance between two nodes).



(a) facets with no constrained edges (b) facets with one constrained edge

Figure 65: Node displacement strategies.

According to the ExtendedIndoorGML Data Model provided in Chapter 5, nodes are stored as *RouteNode* objects, whereas edges are stored as *RouteSegments*. One of the major strenghts of this approach consists in the bidirectionality between primal and dual space. *RouteNodes* are directly associated to the Topographic space (i.e. *AbstractSpace*) they belong to. As a consequence, queries for Indoor Navigation can be performed both geometrically or at the semantic level. More information about implementation details, tests and final remarks as well regarding IGNM can be found in Chapter 7.

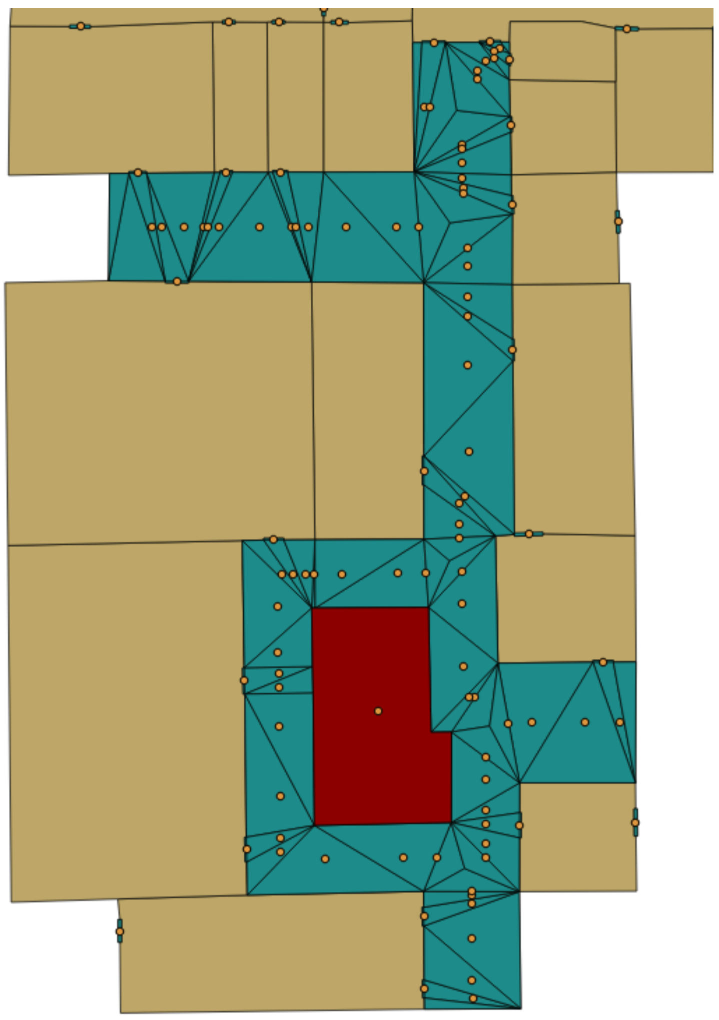


Figure 66: Node displacement.

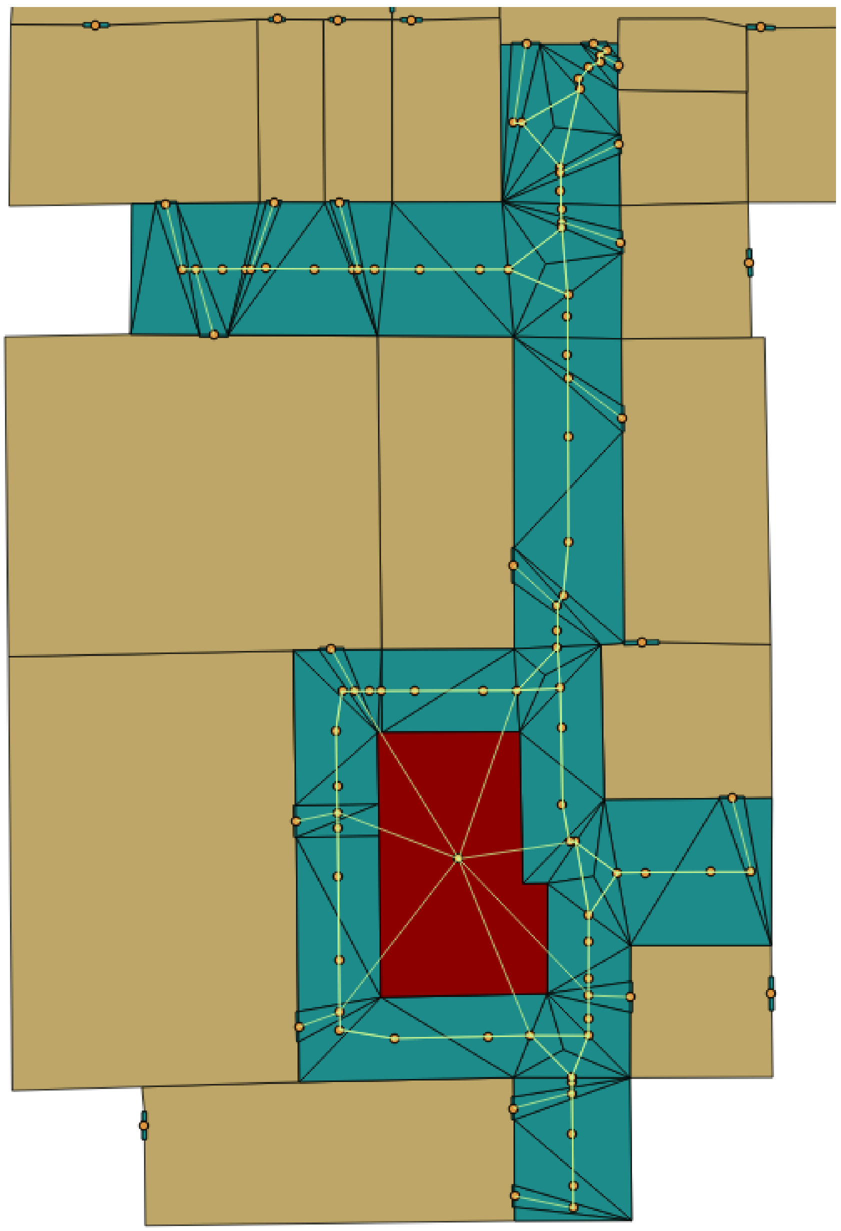
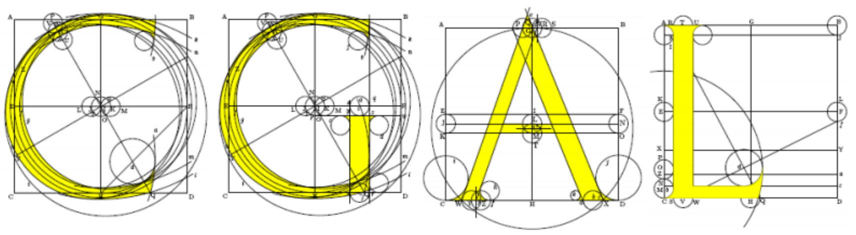


Figure 67: Node linking process: physical path has been finally built.

7 IMPLEMENTATION DETAILS

In this Chapter implementation of the software is discussed. Section 7.1 presents a holistic overview of the well-known Computational Geometry Algorithms Library (CGAL) which has been extensively used within this research project for geometric computations. Section 7.2 provides implementation details about the extraction process of geometry, topology, semantics from CityGML LOD4 datasets, whereas Section 7.3 describes the implementation of the automatic derivation of Improved Geometric Network Model. Moreover in Section 7.3.3 tests and benchmarks with Door-to-Door are presented. Finally some conclusions and final remarks about IGNM are drawn in Section 7.3.4.

7.1 CGAL BACKGROUND INFORMATION



The CGAL Open Source Project provides easy access to efficient and reliable geometric algorithms in the form of a C++ library, offering geometric data structures and algorithms, which are efficient, robust, easy to use, and easy to integrate in existing software. The usage of de facto standard libraries increases productivity, as it allows software developers to focus on the application layer. The CGAL project was founded in 1996, as a consortium of eight research institutions in Europe and Israel: Utrecht University, ETH Zurich, Free University of Berlin, INRIA Sophia Antipolis,Martin-Luther-University Halle-Wittenberg, Max Planck Institute for Informatics Saarbrücken, Johannes Kepler University Linz, and Tel-Aviv University. Its purpose was to *"make the large body of geometric algorithms developed in the field of computational geometry available for industrial applications"* (CGAL Project Proposal, 1996). Some numbers:

- 500,000 lines of C++ code
- 10,000 downloads per year (plus Linux distributions)
- 3,500 manual pages
- 3,000 subscribers to cgal-announce
- 1,000 subscribers to cgal-discuss
- 120 packages
- 60 commercial users (see Figure 68)
- 20 active developers
- 12 months release cycle
- 2 licenses: Open Source and commercial

CGAL is used in various areas needing geometric computation, such as: computer graphics, scientific visualization, computer aided design and modeling, geographic information systems, molecular biology, medical imaging, robotics and motion planning, mesh generation, numerical methods. CGAL offers data structures and algorithms like triangulations (2D constrained triangulations, and Delaunay triangulations and periodic triangulations in 2D and 3D), Voronoi diagrams (for 2D and 3D points, 2D additively weighted Voronoi diagrams, and segment Voronoi diagrams), polygons (Boolean operations, offsets, straight skeleton), polyhedra (Boolean operations), arrangements of curves and their applications (2D and 3D envelopes,

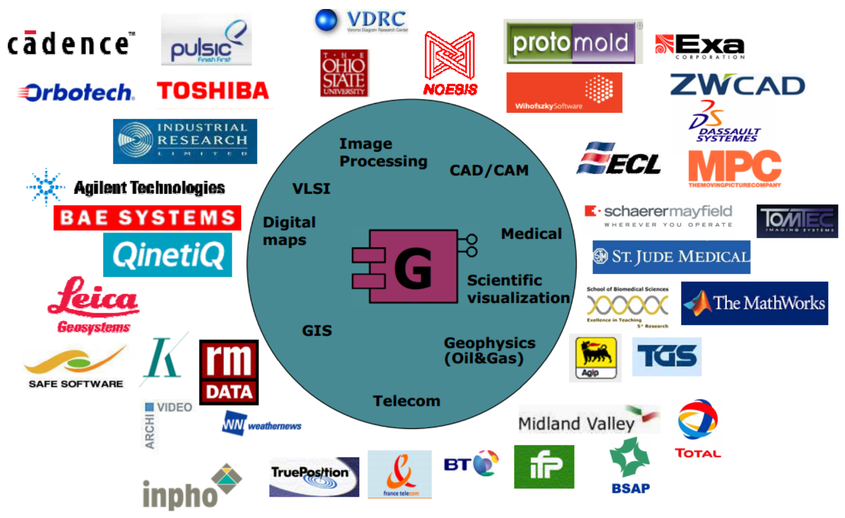


Figure 68: CGAL commercial users

Minkowski sums), mesh generation (2D Delaunay mesh generation and 3D surface and volume mesh generation, skin surfaces), geometry processing (surface mesh simplification, subdivision and parameterization, as well as estimation of local differential properties, and approximation of ridges and umbilics), alpha shapes, convex hull algorithms (in 2D, 3D and dD), search structures (kd trees for nearest neighbor search, and range and segment trees), interpolation (natural neighbor interpolation and placement of streamlines), shape analysis, fitting, and distances (smallest enclosing sphere of points or spheres, smallest enclosing ellipsoid of points, principal component analysis), and kinetic data structures [5].

CGAL library is basically structured in three major components. The first part is the kernel, which consists of constant-size non modifiable geometric primitive objects and operations on these objects. The second part is a collection of basic geometric data structures and algorithms, which are parameterized by traits classes that define the interface between the data structure or algorithm and the primitives they use. In many cases, the kernel classes provided in CGAL can be used as traits classes for these data structures and algorithms. The third part of the library consists of non-geometric support facilities, such as circulators, random sources, I/O support for debugging and for interfacing CGAL to various visualization tools.

The correctness proof of nearly all geometric algorithms presented in theory papers assumes exact computation with real numbers. This leads to a fundamental problem with the implementation of geometric algorithms. Naively, often the exact real arithmetic is replaced by inexact floating-point arithmetic in the implementation. This often leads to acceptable results for many input data. However, even for the implementation of the simplest geometric algorithms this simplification occasionally does not work. Rounding errors introduced by an inaccurate arithmetic may lead to inconsistent decisions, causing unexpected failures for some correct input data. In CGAL it is possible to choose the underlying number types and arithmetic. It is possible to use different types of arithmetic simultaneously and the choice can be easily changed, e.g. for testing. So it is possible to choose between implementations with fast but occasionally inexact arithmetic and implementations guaranteeing exact computation and exact results. Of course you have to pay for the exactness in terms of execution time and storage space [5].

More information about CGAL library can be found on CGAL online manual documentation.

7.2 CITYGML: DERIVATION OF GEOMETRIC, TOPOLOGICAL AND SEMANTIC PROPERTIES – IMPLEMENTATION DETAILS

This Section contains implementation details regarding the software designed for the extraction of geometric, topological and semantic properties from CityGML LOD4 model. In Section 7.2.1 the input files used in this research project will be discussed, whereas in Section 7.2.2 a holistic overview of code design will be given.

7.2.1 *CityGML LOD4 input datasets*

Although CityGML's popularity is increasing day by day, it has been difficult to find different CityGML LOD4 datasets. This last sentence can be justified by the fact that the attention is still focused on LOD2 and LOD3. Existing LOD4 are either incomplete, or not standard compliant. The files inspected in this research project are:

- Single building in Level-of-Detail 4 (11.2 MB), automatically generated from IFC data (Industry Foundation Classes), provided by Homepage of CityGML. This file, even though is provided by the official CityGML working group, is not standard compliant and suffers from various problems. For instance, the boundary surfaces composing the shell of the *Room* solids are not defined as *IntWallSurface*, but as *WallSurface* which, according to the standard, is used only for exterior walls (see Figure 69). Therefore, walls, and consequently Openings nested inside them, are semantically decoupled from rooms, so it is almost impossible to assess topological relashionships between *Room* spaces. Furthermore, objects like stairs have been instantiated as *BuildingInstallation*, whereas CityGML Standard Specification claims that "*BuildingInstallation class is used for building elements like balconies, chimneys, dormers or outer stairs, strongly affecting the outer appearance of a building*".

- FJK-House CityGML LOD4 (16.9 MB), provided by Karlsruhe Institute of Technology, Institute for Applied Computer Science. This file is very clean and it is standard compliant. It has been the mostly used file within this research project.

- OTB (TUDelft) Building (2.9 MB), provided by OTB (TUDelft) PhD Researcher Liu Liu (L.liu-1@tudelft.nl). As the firstly mentioned file, also this one suffers from the same kind of problems and discrepancies. This file has been automatically generated by OSM2CityGML Export Algorithm developed and im-

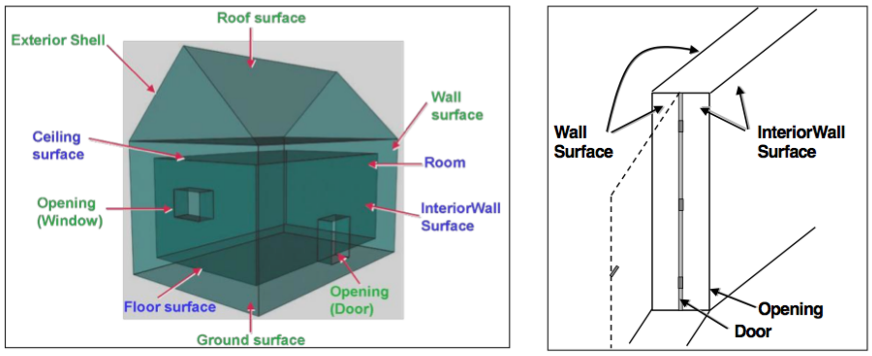


Figure 69: Classification of BoundarySurfaces (left), in particular for Openings (right) (graphic: IGG Uni Bonn)

plemented by Marcus Goetz (GIScience Research Group, Department of Geography, University of Heidelberg). More information about his research can be found in [11]. In this model *InteriorWallSurfaces* are instantiated as child classes of Building element, whereas the CityGML specification says that "*the class InteriorWallSurface must only be used in the LOD4 interior building model for modelling the visible surfaces of the room walls*". As a consequence, also in this case topological relashionships between *Room* spaces could not be retrieved.

7.2.2 *Code details*

The code for extracting geometric, topological, semantic features from CityGML for Indoor Navigation has been written in C++. During a *"make or buy"* analysis phase, an interesting library has been found on the Web: Libcitygml (https://code.google.com/p/libcitygml/)[1]. This is an open source C++ library for parsing CityGML files designed mainly for 3D rendering applications. But since it was poorly documented and most of its features like tessellation and optimization of geometry data for rendering during parsing weren't needed, it has been decided to implement from scratch the parser for CityGML. Fig-

1 available here: https://code.google.com/p/libcitygml/

ure 71 illustrates in Unified Modeling Language (UML) the classes designed for building the parser. For XML DOM parsing an external library has been used, named TinyXML2[2], a simple, small, efficient, DOM based XML parser. Basically, the software works as follows:

Figure 70: TinyXML2 Logo.

1. First of all, it loads the CityGML file
2. It iteratively scans each *Room* object instantiated. For each *Room*:
   a) It extracts geometry that describes floor surface
   b) It extracts information about *_Opening* objects such as *Door* and *Window* and stores their geometry. It determines 2D projection on floor of these objects
   c) It extracts information about objects that potentially may obstruct passages for navigation (in other words *IntBuildingInstallation* and *BuildingFurniture*) and subtracts its 2D projection from Room's navigable space (partially implemented)
3. It aggregates *Rooms* and *IntBuildingInstallation* objects based on certain height criteria in order to explicitly define the concept of Storey, not present in CityGML
4. It reconstructs topological relationships between spaces (i.e. adjacency / connectedness)

2 available here: http://www.grinninglizard.com/tinyxml2/

THE OUTPUT During the processing, the acquired data is incrementally fetched into the Data Model, such that, at the end of the acquisition, the latter is fully and consistently populated. As depicted in Figure 71, cityGMLParser object invokes methods defined inside *Building* class, in order to populate the Model with acquired information. In fact, *Building* is the Information Expert, and brakes the coupling between the Data Model and the CityGML parsing component. This principle protects elements from the variations on other elements (objects, systems, subsystems) by wrapping the focus of instability with an interface (Building).

SOURCE CODE This software package is available under GNU GPL v2 code license at this Google Code Repository
https://code.google.com/p/subdivision-thesis-new/.

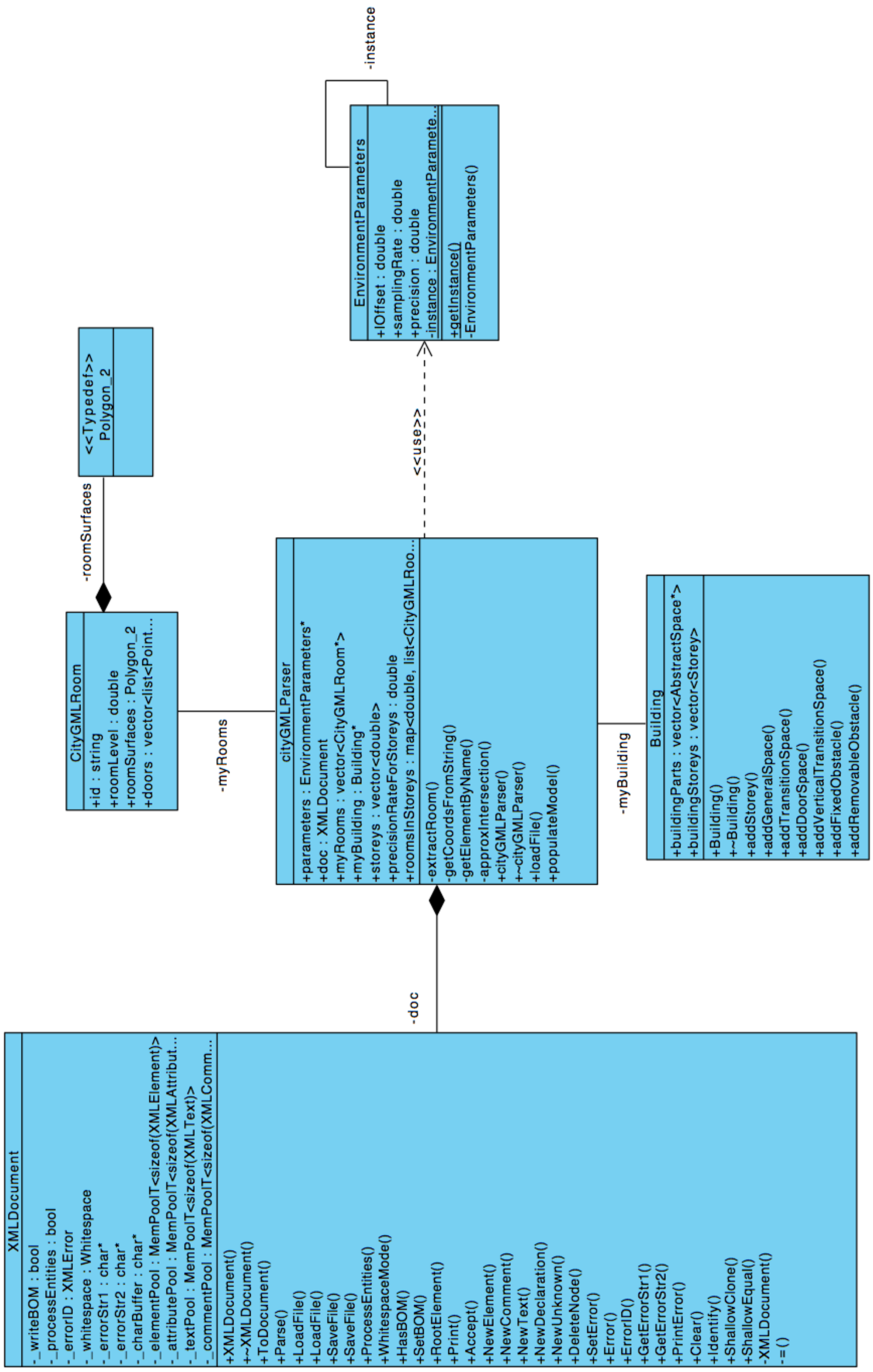


Figure 71: UML Diagram of the classes designed for CityGML Parser

VISUALIZATION For CityGML visualisation, this software has been used:

- Safe Software FME Data Inspector 2013 (commercial) - visualisation tool part of the FME Desktop, an integrated collection of Spatial ETL (Extract, Transform, and Load) tools for spatial data transformation and data translation produced by Safe Software Inc. of Surrey, British Columbia, Canada. This is considered to be a GIS (Geographic Information Systems) utility to help users convert data between formats as well as process data geometry and attributes.
- Autodesk LandXplorer CityGML Viewer 2009 (free), a free downloadable program to view CityGML data sets.
- FZKViewer 4.0 (Build 744) (free) - visualisation tool capable of reading several Semantic Geographic Information Models like IFC, CityGML amd gbXML, developed by Karlsruher Institute of Technology, Institute for Applied Computer Science

7.3 THE AUTOMATIC NETWORK EXTRACTION PROCESS - IMPLEMENTATION DETAILS

This Section contains implementation details about the software designed for the automatic derivation of Improved Geometric Network Model. In Section 7.3.1 the input files are discussed, whereas in Section 7.3.2 holistic overview of code design is given.

7.3.1 *Input Files*

Most of the datasets taken as input files for Improved Network Model algorithm have been generated by using JOSM software[3] (see Figure 72). JOSM (Java OpenStreetMap Editor) is an editor for Open-

3 available here: http://josm.openstreetmap.de/

StreetMap data. It does not require an Internet connection while editing, and is suited for the needs of advanced users. A majority of edits to the OpenStreetMap database are contributed using JOSM. Since as

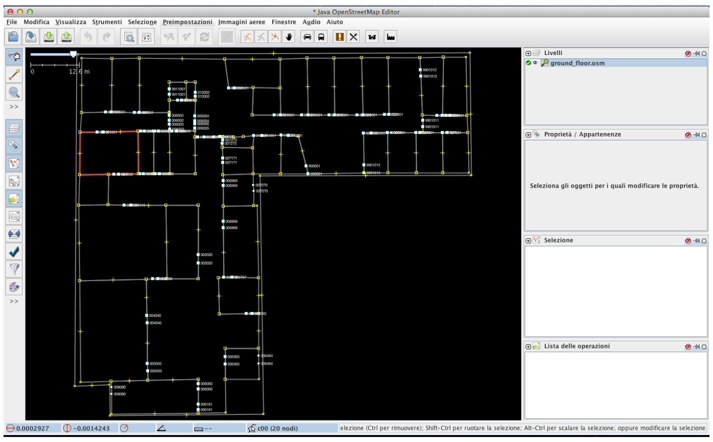

* Java OpenStreetMap Editor
File Modifica Visualizza Strumenti Selezione Preimpostazioni Immagini aeree Finestre Audio Aiuto
Livelli
ground_floor.osm
Proprietà / Appartenenze
Seleziona gli oggetti per i quali modificare le proprietà.
Selezione
Lista delle operazioni

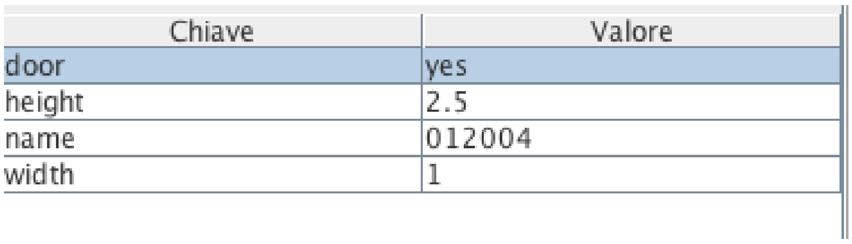
Figure 72: A screenshot of JOSM user interface

it has been argued in Section 7.2 a very poor amount of CityGML LOD4 datasets has been found, it has been essential to simulate, by the usage of this software, the semantics, the topology, the geometry provided by CityGML. For the purpose, ad-hoc tags have been introduced, as it is visible from Figure 73, like "object_type" for identifying *IntBuildingInstallation* or *BuildingFurniture*, "door" for marking specific edges as doors, "door_ name" in order to reference the same door by two rooms connected by it and so on.

7.3.2 *Code details*

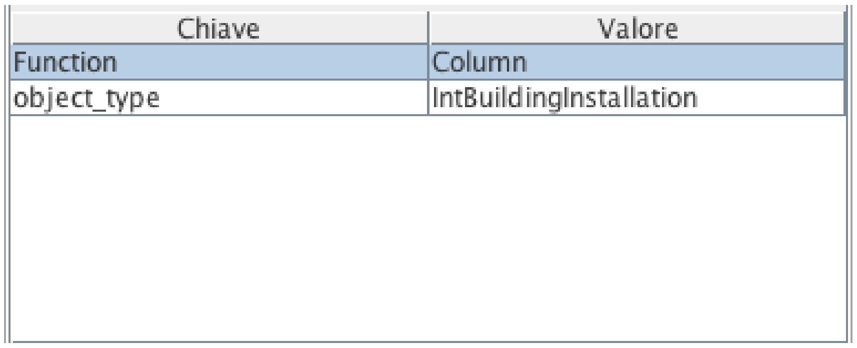
The software has been structured in four main packages, as it can be seen in Figure 74.

- Model Package contains domain classes as they have been introduced in Chapter 5. These classes constitute the core of the Data Model and collect the concepts that abstract Indoor Navigation application domain.



Chiave	Valore
door	yes
height	2.5
name	012004
width	1

(a) Semantics of doors



Chiave	Valore
Function	Column
object_type	IntBuildingInstallation

(b) Semantics of building installation

Figure 73: Ad-hoc tags introduced in JOSM

- CityGMLParser Package contains the classes that have been illustrated in Section 7.2. Their scope is limited to the derivation of geometry, topology, semantics from CityGML.
- JOSMDataFetcher Package contains the classes that perform extraction of information from JOSM files. As described in Section 7.3.1, it had been necessary to reproduce CityGML LOD4 major features, since no sufficient datasets were available.
- NetworkGeneratorControl Package is the software component responsible for the automatic derivation of Improved Geometric Network Model.

JOSM EXTRACTION A JOSM file (*.osm*) is essentially a XML file. Polygons are stored as *"ways"*, which are basically closed curves obtained as a concatenation of *"nodes"* that are 2D Points. In JOSM, as already mentioned in Section 7.3.1 semantics can be inserted as prop-

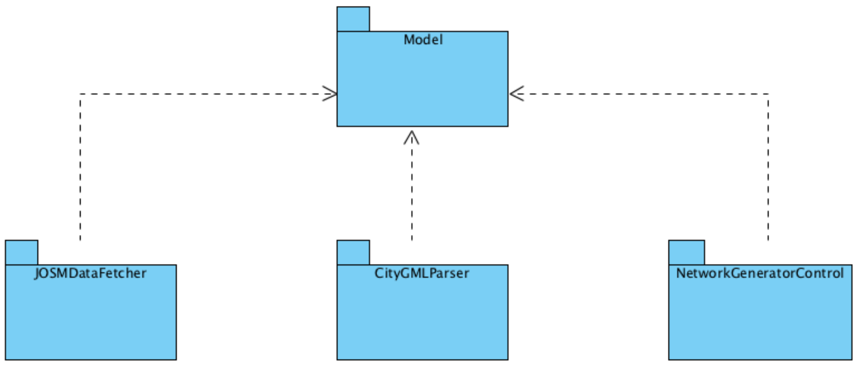


Figure 74: UML Diagram of packages designed for the implementation

erties both of *ways* and *nodes*. For instance, Listing 4 displays a detail of a *node* and a detail of a *way*.

Listing 4: Excerpt of a .osm file

```
/...
<node id='-334' timestamp='2012-11-26T15:33:49Z' visible='true' lat='3.53777E-4'
     lon='-0.001318082'>
   <tag k='connector:ids' v='106008' />
   <tag k='door' v='yes' />
   <tag k='height' v='3' />
   <tag k='name' v='006008' />
   <tag k='width' v='2' />
</node>
....
....
<way id='-492' action='modify' timestamp='2012-11-26T15:33:59Z' visible='true'>
   <nd ref='-338' />
   <nd ref='-336' />
   <nd ref='-4' />
   <nd ref='-334' />
   <nd ref='-332' />
   <nd ref='-330' />
   <nd ref='-338' />
   <tag k='buildingpart' v='verticalpassage' />
   <tag k='buildingpart:verticalpassage' v='stairs' />
   <tag k='buildingpart:verticalpassage:floorrange' v='0 to 1' />
   <tag k='height' v='3' />
   <tag k='indoor' v='yes' />
   <tag k='name' v='e01' />
 </way>
```

In order to read the file, a parser has been hardcoded using TinyXML2, a C++ DOM XML parser[4]. Once all the *nodes* and all the *ways* have been loaded in RAM, topology and semantics have been reconstructed. For the purpose a winged edge data structure (see Figure 75) has been implemented. For each room this kind of information is reconstructed:

- 2D geometry of room footprints
- doors and windows (both on the semantic and geometric level)
- neighbouring rooms (each instantiated *Room* is associated to its neigbours)
- obstacles (both on the semantic and geometric level)

Each of this information, once extracted, is fetched into the ExtendedIndoorGML Data Model, as it has been proposed in Chapter 5

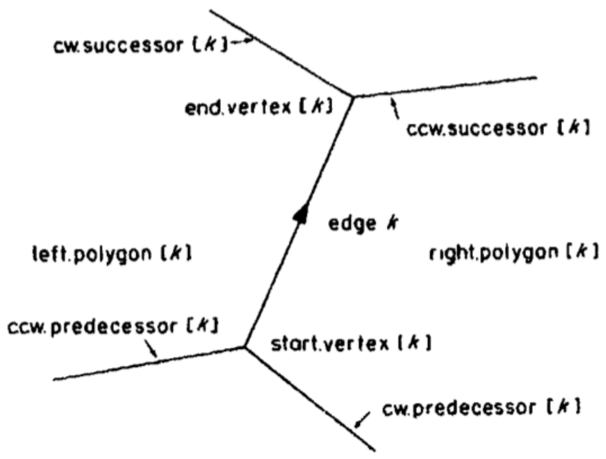


Figure 75: Winged Edge Data Structure

NETWORK CONSTRUCTION Once the model has been succesfully populated with appropriate information (coming either from CityGML or from JOSM), automatic extraction can be performed. Most of the geometric operations have been implemented using Computational

4 available here: http://www.grinninglizard.com/tinyxml2/

Geometry Algorithms Library (CGAL). For further information about CGAL please refer to Section 7.1. "Exact Predicates Exact Constructions Kernel" has been adopted. It uses Cartesian representation, it supports constructions of points from double Cartesian coordinates, it provides both exact geometric predicates and exact geometric constructions. Though, sometimes precision matters affected the results such that we had to implement some functions in a more naive way, like finding if a point lays on line segment.

Implementation follows the algorithm sketch described in Chapter 6. Coarse inter-space path finding has not yet being implemented and is left to future work. Regarding intra-space path-construction some remarkable aspects can be illustrated:

- for Inward Offset computation, it has been used "CGAL::create_ exterior_ skeleton_ and_ offset_ polygons_ 2" function. In case obstacles were present inside the room (i.e. the 2d polygon representing its footprints has holes), "CGAL::create_ interior_ skeleton_ and_ offset_ polygons_ with_ holes_ 2" has been used.
- for computing CDT, "CGAL::Constrained_ Delaunay_ triangulation_ 2" class has been used. For limiting the triangulation to the polygonal domain we explored the set of facets connected with non constrained edges, and attribute to each such set a nesting level. We started from facets incident to the infinite vertex, with a nesting level of 0. Then we recursively consider the non-explored facets incident to constrained edges bounding the former set and increase the nesting level by 1. Facets in the domain are those with an odd nesting level. Facets with particular semantics have been marked using "CGAL::Triangulation_ face_ base_ with_ info_ 2". This class let the programmer add some user defined information to triangulation facets. Listing 5 shows the *struct* that has been used to mark facets with additional information. Semantics like CROSSING, HOLENEIGHBOUR,

DOORADJACENT, DEADEND, OBSTACLEADJACENT describe the type of space as it has been classified in Section 6.3.1.

Listing 5: Excerpt of code: adding info to a triangulation facet

```
/enum FaceType {CROSSING, HOLENEIGHBOUR, DOORADJACENT, DEADEND, OBSTACLEADJACENT};
class NavigableSpace;
struct FaceInfo2
{
  FaceInfo2(){}
  int nesting_level;
  list<FaceType> faceType;
  int id;
  NavigableSpace* mySpace;

  bool in_domain(){
    return nesting_level%2 == 1;
  }
};
```

OUTPUT To validate results and to assess correctness of geometric operations, the obtained path (*RouteNodes* + *RouteSegments*) has been exported in ESRI Shapefile (.shp). For the purpose GDAL OGR Simple Feature Library has been used[5]. The OGR Simple Features Library is a C++ open source library (and commandline tools) providing read and write access to a variety of vector file formats including ESRI Shapefiles, S-57, SDTS, PostGIS, Oracle Spatial, and Mapinfo mid/mif and TAB formats.

RouteNodes and *RouteTransitions* have been also exported in CSV (Comma Separated Values) format to prove that once constructed, the intra-space path can be easily stored in a Database or exported in XML for web-services or web-GIS, for Personal Navigation Systems and similar applications.

SOURCE CODE This software package is available under GNU GPL v2 code license at this Google Code Repository

5 available here: http://www.gdal.org/ogr/

https://code.google.com/p/subdivision-thesis-new/. The source code is built upon CGAL and therefore GPL applies to this project.

7.3.3 *Tests and validation*

Test have been conducted in order to evaluate the quality of the proposed solution and to validate the given results. Intra-space connectivity algorithm proved to work correctly in:

- arbitrarily-shaped rooms with concave corners (i.e. irregular concave polygons)
- rooms with obstacles
- rooms with open spaces such that convex inset polygons are generated
- rooms with open spaces such that concave inset polygons are generated (only concave inset polygons that have their centroid inside the contour have been tested)

The floor plans that have been used are:

- OTB Building (TUDelft, Delft, The Netherlands) floorplan drawn with JOSM software (For more implementation details please refer to Section 7.3)
- FJK-House CityGML LOD4 (16.9 MB), provided by Karlsruhe Institute of Technology, Institute for Applied Computer Science. Since it is a 3D CityGML file, 2D footprints of each floor have been previously extracted before launching the algorithm.

Moreover, some benchmarks with Door-to-Door have been conducted in collaboration with OTB (TUDelft) PhD Researcher Liu Liu (L.liu-1@tudelft.nl). For illustrative purpose some results can be shown: Figure 78 displays the different networks constructed on the same hallway, wheras Figure 79 depicts the best paths from a source to a target found by running Dijkstra algorithm (another graphic example is shown in Figure 80). As Table 1 suggests, regarding the lenght of shortest path, IGNM generated path is ≈ 1.9 times longer than Door-to-Door which is known to be optimal. In average case, IGNM proves to be ≈ 1.6 longer than the optimal Door-to-Door, which is a promising result. Regarding the amount of nodes and edges, the obtained results are affected by two key factors:

- unfortunately for time constraints it has not been possible to encompass semantics in Door-to-Door implementation for tests. This means that Door-to-Door graph construction relies only on geometric aspects; in other words points that represent doors do not appear as intervisible locations. This consequently reduces the amount of nodes/edges that appear in the constructed Visibility Graph. In fact, just to prove this statement, following Door-to-Door approach, we manually drew nodes that represent either doors or concave corners, and edges connecting them, on the same hallway used for running Test Nr.1 (please refer to Figure 76). The result was not surprising: the amount of nodes increases from 9 to 24 and the amount of edges increases from 36 to 194.

- JOSM, the software used for drawing 2D floorplans, adds collinear points on the contour of the polygon representing a room everytime an incident edge of neighbouring rooms touches the contour itself. As a result, the number of facets significantly increases (see Figure 77). Due to time limits, it has not been possible to code a routine that could suppress redundant collinear points from the contour of the polygons.

Therefore, these results are distorted by the aforementioned factors. Still, the amount of nodes and edges is high. But a fine-grained classification of indoor space has the positive implication of improving the quality of localization, which is mostly faulty in network graphs generated by both S-MAT and Door-to-Door. Anyway in future investi-

	IGNM	DOOR-TO-DOOR
Nr. of Nodes	77	9
Nr. of Edges	152	36
Lenght of Best Path (m)	25.5705	46.2158
Time of graph construction + path-finding (s)	0.001	0.001

Table 1: Test Nr.1: Results from comparison between IGNM and Door-to-Door (referring to the hallway illustrated in Figure 78)

gations it is recomended to evaluate these results also by considering the suggestions previously given.

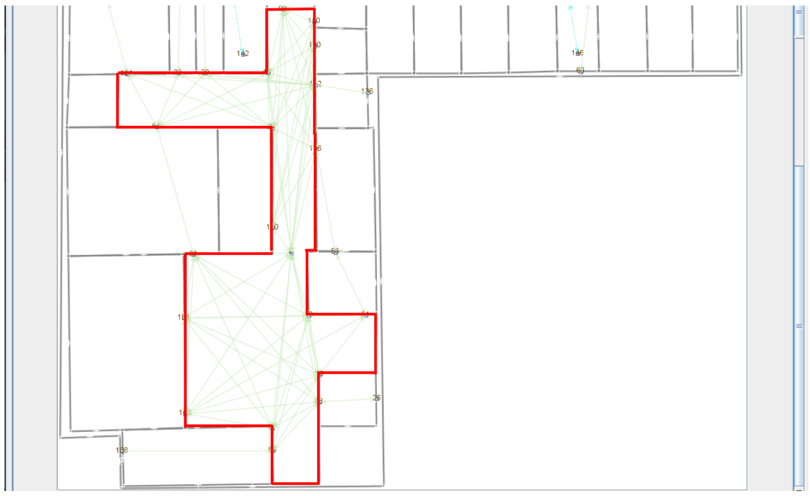


Figure 76: Manually-generated Door-to-Door Graph. Nodes represent either doors or concave corners.

7.3.4 *Final remarks about IGNM*

Improved Geometric Network Graph proved to be a valid alternative to S-MAT and Door-to-Door. In fact the graph is constructed with a two-level hierarchical strategy, which aims to increase performance, avoiding the geometric computation on unnecessay spaces that for sure will be not touched by the resulting navigation path from a

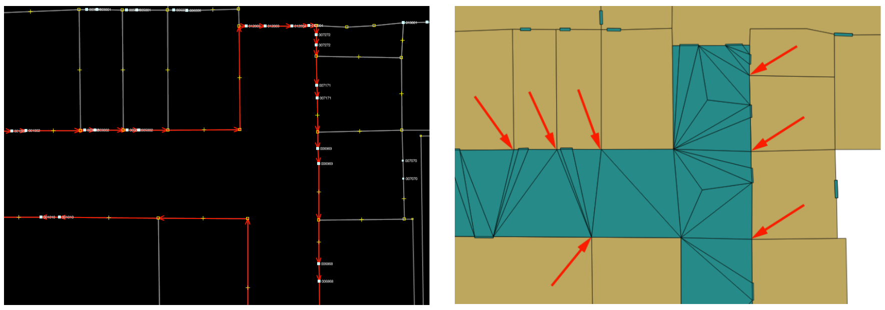
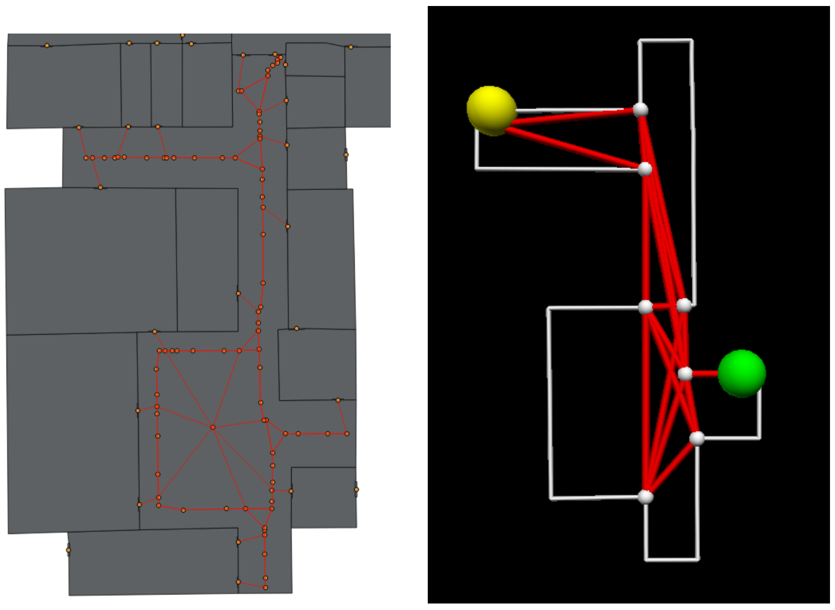


Figure 77: Collinear points on the contour significantly contribute to increasing the number of the facets of the constrained Delaunay Triangulation

	IGNM	DOOR-TO-DOOR
Nr. of Nodes	81	10
Nr. of Edges	172	32
Lenght of Best Path (m)	52.7788	79.4643
Time of graph construction + path-finding (s)	0.002	0.005

Table 2: Test Nr.2: Results from comparison between IGNM and Door-to-Door. Please refer to Figure 80 for visualization

source to a destination target. Geometric algorithm is local, so it is flexible and efficient, and moreover capable of supporting dynamic changes. Differently from S-MAT and Door-to-Door the transformations between primal space and dual space are bidirectional: after graph construction nodes are still aware of the topographic space they belong to. The latter property, plus contour sampling which regulularizes the triangulation, contribute to improving the accuracy of localization inside a building. The Constrained Delaunay Triangulation helps in generating a centerline based path which results to be a good representation of natural human behaviour within indoor environment. Nevertheless, as illustrated in [15], Constrained Delaunay Triangulation is an efficient tool for obstacle-avoidance path planning. Obstacles not only are encompassed, but are allowed to be inserted,

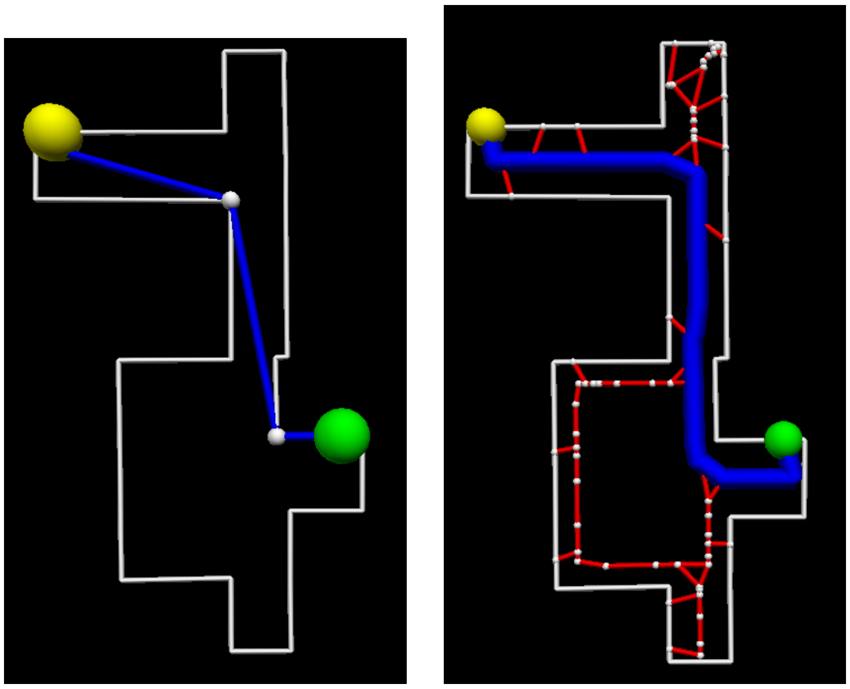


(a) IGNM approach resulting network (b) Door-to-Door apprach resulting network

Figure 78: Test Nr.1: Comparison between IGNM and Door-to-Door

removed or displaced in the CDT as required during run-time. Inward Offsetting, by detecting large, open spaces that are clearly the weak spot of Medial Axis Transform helps in overcoming the aforementioned problem of the distortion of the path towards the middle of the open space. Moreover, by adding extra nodes inside the generated inset polygon, and by linking them to their nearest *DoorAdjacentNodes* and *Crossings* that are also *HoleAdjacentNodes*, the network somehow tries to reproduce the positive aspects of a "Door-to-Door" navigation between intervisible spots. Summing up: centerline approach for narrow corridors, "Door-to-Door" for larger, open, environments.

Though, some drawbacks are present. For doors located in the corners of the rooms the algorithm might produce some serpentine routes, as displayed in Figure 81a. This is basically due to the shape of the triangulation, as the facets adjacent to doors are streched with angles $> 90°$. In order to overcome this deficiency, some path simplification process might be applied, following, for instance, the method



(a) IGNM path computation (b) Door-to-Door path computation

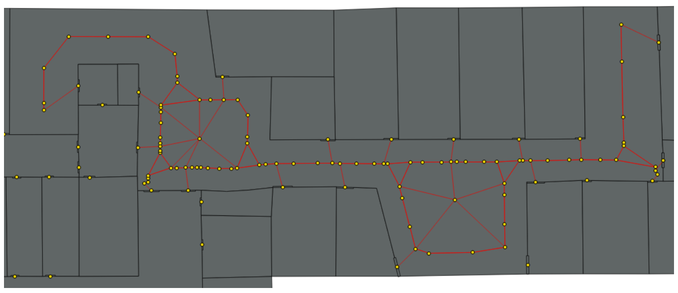
Figure 79: Comparison between IGNM and Door-to-Door: path computation

proposed in [2], being careful that removal of nodes doesn't produce unwanted intersections of the path with space boundaries. Let the reader recall that this might be a post-processing step that might result useful only for visualization and / or for generating route directions. In fact we still want to preserve the bidirectionality between primal space and dual space.

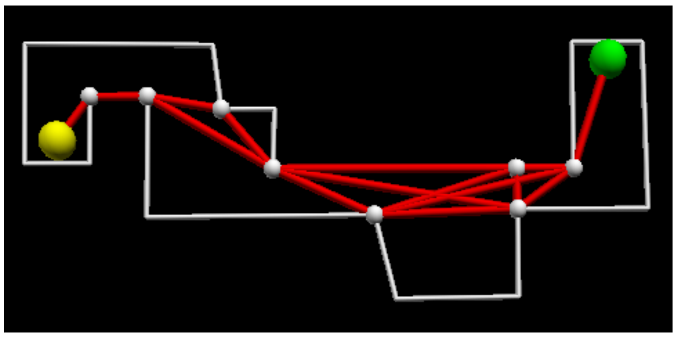
Another limitation consists in the technique for displacing one or additional nodes inside the inset polygon in case its centroid falls outside its contour. A non-convex object might have a centroid that is outside the figure itself. Usually, even though inward offsetting computation might produce non-convex polygons, those latter usually have regular shapes because building spaces themselves are usually regular. In case the centroid falls outside, the proposed technique of computing a convex decomposition and then displacing nodes in the centroids of the newly generated partitions might result too naive, since it may produce circuitous routes/detours. More investigation about this topic has to be done and is left to future work.

Moreover, Section 7.3.3 reveals that the lenght of the generated path might considerably differ from the optimal path which is constructed with visibility approach. Nevertheless, let the reader recall that centerline approaches generate routes that are an appropriate abstraction for indoor navigation networks. Even if visibility-driven approaches produce optimal-shortest paths, it may be difficult to generate clear, human-understandable route directions on top of them.

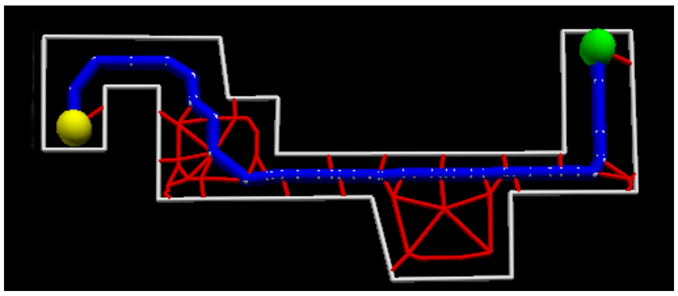
Another deficiency in Improved Network Model is the number of nodes and edges: even though the amount of edges is lower than the one produced with "Door-to-Door" (semantics included), still it is quite high compared to S-MAT. Several factors may affect this result, as it has been argued in Section 7.3.3. Though, as already discussed, a fine-grained classification of indoor space has the positive implication of improving the quality of localization, which is mostly faulty in network graphs generated by both S-MAT and Door-to-Door.



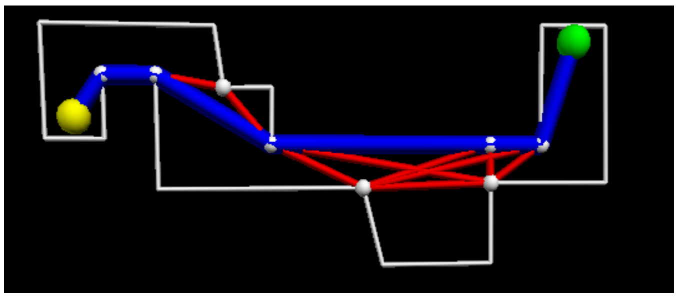
(a) IGNM path computation



(b) Door-to-Door path computation

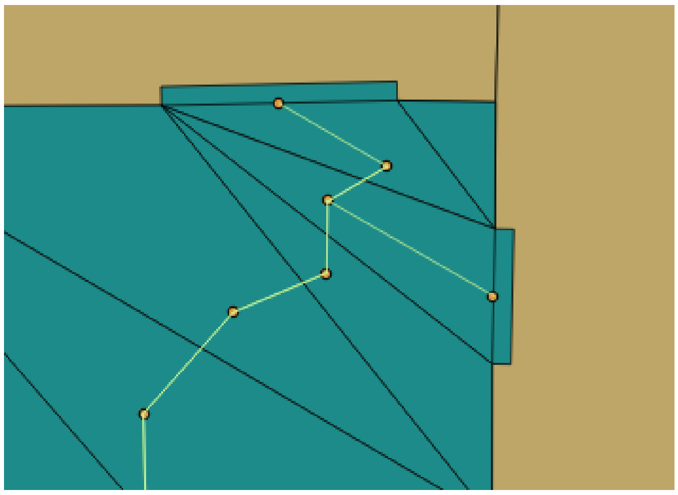


(c) IGNM shortest path



(d) Door-to-Door shortest path

Figure 80: Test Nr. 2: Comparison between IGNM and Door-to-Door: path computation


(a) Serpentine routes in proximity of corners

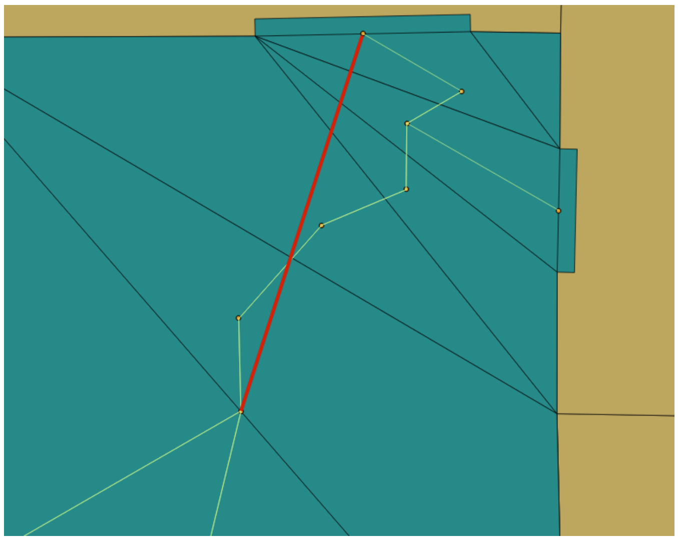

(b) Path simplification for serpentine routes

Figure 81: Serpentine routes in proximity of corners and path simplification

8 CONCLUSION AND RECOMMENDATIONS

This research project dealt with the automatic generation of an improved version of the so called "Geometric Network Model", a network graph representing both topology and geometry of indoor spaces. For the purpose, also an appropriate Data Model has been designed in order to provide the proper abstractions and concepts that may be needed within the context of an Indoor Navigation application. The model, realised as an extension of Indoor Navigation Module of the OGC candidate standard IndoorGML, provides the necessary means for the network generation, in terms of geometry, topology, semantics. Furthermore, since very poor information has been found from current literature regarding automatic derivation of Geometric Network Models from semantically rich 3D digital information Models like IFC and CityGML, it has been investigated how the latter could be beneficial for Indoor Navigation. In fact, even though these models are not specifically designed for this kind of applications, they might provide the information needed within this context. Several conclusions can be drawn regarding the results gained with this research and are discussed in the fortcoming Sections.

8.1 ANSWERING THE RESEARCH QUESTIONS

Returning to the hypotheses/questions posed at the beginning of this study, we can now state that it is possible to automatically extract an Improved Geometrical Network Model from CityGML LOD4 for Indoor Navigation.

8.1.1 *About derivation of geometric, topological and semantic properties from CityGML LOD4*

One of the more significant findings to emerge from this study is that CityGML can substantially provide most of the necessary information for Indoor Navigation, even though this model is not designed ad-hoc for this purpose. The building model is one of the most detailed thematic concepts of CityGML. It allows for the representation of thematic and spatial aspects of buildings and building parts in five levels of detail, LOD0 to LOD4. In LOD4, the highest level of resolution, also the interior of a building, composed of several rooms, is represented in the building model by the class *Room*. This enlargement introduced by LOD4 allows a virtual accessibility of buildings: we proved that geometry (both 2D and 3D) of *Rooms* can be used to infer the concept of floorplans, even if it is not explicitly defined in the model, and navigable (both walkable and drivable) spaces can be extracted. Furthermore we pointed out that topological properties, such as connectedness, between spaces can be desumed both at the semantic level and by performing geometric operations. Thematic objects such as *Door*, *Window* can describe hatches or passages between adjacent rooms. Adjacency of spaces is an essential property needed for implementing the construction of the Network Graph. Semantics yet present in CityGML contribute to the enrichment of the Indoor Navigation Model, providing usefeul information like the presence of Building Installations and Building Furniture that may be interpreted as fixed and movable obstacles respectively. However, this research highlighted the fact that some fundamental concepts for navigation are still missing. For instance, it has been remarked that vertical connectivity concept which is essential for a coarse inter-space path finding, cannot be inferred because there are no classes that directly map this concept into the model. Stairs, elevators, escalators, are generically mapped into *BuildingInstallation* class. Let the reader recall that CityGML is not focused on usage and localization of features (stationary or mobile) in indoor space. Its main concern is to provide representation of sets of 3D urban objects (building architectural components if we are referring to the Building Module), with the purpose of reaching a common definition of the basic entities, attributes, and relations of a 3D city model. More concern on indoors by CityGML working group would help us to establish a greater degree of accuracy on this matter of Indoor Navigation.

8.1.2 *About Indoor Data Modeling*

The evidence from this study suggests that the design of a proper Data Model is essential for supporting different environments/scenarios for specific indoor navigation tasks. This work contributes to existing knowledge by providing a Data Model whose aim consists in filling the deficiencies yet present in the approaches that have been found from current literature, like incomplete concerns about semantics of building spaces, limited consideration of indoor navigation with obstacles, coarse-grained granularity in modeling large, open spaces. Our proposed Data Model has been designed as a thematic extension of the Indoor Navigation Module of OGC's candidate standard named IndoorGML. We proved its flexibility, its modularization, and its capability of being adapted to different requirements as well. The results of this investigation, though, show that IndoorGML has its own limitations: some fundamental concepts for indoor navigation are still missing. Obstacles are not encompassed in the model, yet in emergencies, obstacles are significant as they may change a path and impact the safety of pedestrian's movement. We remarked the necessity of distinguish between navigable and non-navigable space. Moreover, concept of storeys is completely missing. Especially during the process of generating routing directions, it is essential to provide the right, human-comprehensible instruction, especially if the navigation involves traversing vertical spaces that connect different floors.

8.1.3 *About the automatic derivation of Improved Geometric Network*

The findings from this study make another important contribution to the current literature. We presented a novel approach for the automatic derivation of Geometric Network Graph. The proposed strategy aims to comprehensively convey the major features and benefits of both S-MAT and Door-to-Door, the two widely used approaches for automatic network generation. The algorithm is geometry-based, but semantics like obstacles, constraints, openings and passages are encompassed. Section 7.3.4 extensively illustrates its advantages and its major limitations, as well. Also recommendations are given for future research on this topic. At present, the Improved Geometric Network Graph is not validated yet by elaborated tests. In the next stage, further experimental investigations are needed to estimate the behaviour of the algorithm especially in the case of inset polygons having their centroid outside the contour.

BIBLIOGRAPHY

[1] Imad Afyouni, Cyril Ray, and Christophe Claramunt. Spatial models for context-aware indoor navigation systems: A survey. *Journal of Spatial Information Science*, (4):85–123, 2013.

[2] Maneesh Agrawala and Chris Stolte. Rendering effective route maps: improving usability through generalization. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 241–249. ACM, 2001.

[3] Srikanth Bandi and Daniel Thalmann. Space discretization for efficient human navigation. In *Computer Graphics Forum*, volume 17, pages 195–206. Wiley Online Library, 1998.

[4] Thomas Becker, Claus Nagel, and Thomas H Kolbe. A multilayered space-event model for navigation in indoor spaces. In *3D Geo-Information Sciences*, pages 61–77. Springer, 2009.

[5] CGAL. Computational geometry algorithms library - official website. URL http://www.cgal.org/.

[6] L Paul Chew. Constrained delaunay triangulations. *Algorithmica*, 4(1-4):97–108, 1989.

[7] Youness Dehbi and Lutz Plümer. Learning grammar rules of building parts from precise models and noisy observations. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(2):166–176, 2011.

[8] Mohamed El-Mekawy, Anders Östman, and Ihab Hijazi. An evaluation of ifc-citygml unidirectional conversion. *International Journal*, 2012.

[9] Cacciola Fernando. A survey of polygon offseting strategies. URL http://fcacciola.50webs.com/Offseting%20Methods.htm.

[10] Marcus Goetz and Alexander Zipf. Formal definition of a user-adaptive and length-optimal routing graph for complex indoor environments. *Geo-Spatial Information Science*, 14(2):119–128, 2011.

[11] Marcus Goetz and Alexander Zipf. Towards defining a framework for the automatic derivation of 3d citygml models from volunteered geographic information. *International Journal of 3-D Information Modeling (IJ3DIM)*, 1(2):1–16, 2012.

[12] Christopher Gold. Problems with handling spatial data-the voronoi approach. *CISM JOURNAL ACSGC*, (1):65–80, 1991.

[13] Open Geospatial Consortium Inc. Opengis city geography markup language (citygml) encoding standard. version 1.0.0., August 2008. URL http://www.citygml.org/.

[14] ISO 19107. Geographic information - Spatial schema, 2003.

[15] Marcelo Kallmann. Path planning in triangulations. In *Proceedings of the IJCAI workshop on reasoning, representation, and learning in computer games*, pages 49–54, 2005.

[16] Ismail R Karas, Fatmagul Batuk, Abdullah E Akay, and Ibrahim Baz. Automatically extracting 3d models and network analysis for indoors. In *Innovations in 3D Geo Information Systems*, pages 395–404. Springer, 2006.

[17] Aftab Ahmed Khan and Thomas H Kolbe. Constraints and their role in subspacing for the locomotion types in indoor navigation. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–12. IEEE, 2012.

[18] Thomas H Kolbe, Thomas Becker, and Claus Nagel. 1 st technical report discussion of euclidean space and cellular space and proposal of an integrated indoor spatial data model.

[19] Thomas H Kolbe, Gerhard Gröger, and Lutz Plümer. Citygml: Interoperable access to 3d city models. In *Geo-information for disaster management*, pages 883–899. Springer, 2005.

[20] Fabrice Lamarche and Stéphane Donikian. Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. In *Computer Graphics Forum*, volume 23, pages 509–518. Wiley Online Library, 2004.

[21] J Lee and Sisi Zlatanova. A 3d data model and topological analyses for emergency response in urban areas. *Geospatial information technology for emergency response*, 143:C168, 2008.

[22] Jiyeong Lee. *3D GIS for Geocoding Human Activity in Microscale Urban Environments*. 2004. doi: 10.1007/978-3-540-30231-5_11.

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

[24] Xiang Li, Christophe Claramunt, and Cyril Ray. A grid graph-based model for the analysis of 2d indoor spaces. *Computers, Environment and Urban Systems*, 34(6):532–540, 2010.

[25] L Liu and S Zlatanova. A" door-to-door" path-finding approach for indoor navigation. In *Proceedings of GeoInformation For Disaster Management Conference 2011*, pages 3–8, 2011.

[26] Liu Liu and Sisi Zlatanova. A semantic data model for indoor navigation. In *Proceedings of the Fourth ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, pages 1–8. ACM, 2012.

[27] Liu Liu, Sisi Zlatanova, and George Sithole. A conceptual framework of space subdivision for indoor navigation. In *Proceedings of the Fifth ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*. ACM, 2013.

[28] Bernhard Lorenz, Hans Jürgen Ohlbach, and Edgar-Philipp Stoffel. A hybrid spatial model for representing indoor environments. In *Web and Wireless Geographical Information Systems*, pages 102–112. Springer, 2006.

[29] Daniel R Montello. Scale and multiple psychologies of space. In *Spatial information theory a theoretical basis for gis*, pages 312–321. Springer, 1993.

[30] Ivin Amri Musliman, Alias Abdul Rahman, Volker Coors, et al. Implementing 3d network analysis in 3d-gis. *International archives of ISPRS*, 37(part B), 2008.

[31] OGC IndoorGML. OGC® IndoorGML - Draft v.0.6.4, 2013.

[32] Alexandra Stadler and Thomas H Kolbe. Spatio-semantic coherence in the integration of 3d city models. In *Proceedings of the 5th International Symposium on Spatial Data Quality, Enschede*, 2007.

[33] Edgar-Philipp Stoffel. *Hierarchical graphs as organisational principle and spatial model applied to pedestrian indoor navigation*. PhD thesis, lmu, 2009.

[34] Edgar-Philipp Stoffel, Bernhard Lorenz, and Hans Jürgen Ohlbach. Towards a semantic spatial model for pedestrian indoor navigation. In *Advances in Conceptual Modeling–Foundations and Applications*, pages 328–337. Springer, 2007.

[35] S Taneja and E William East. Transforming an ifc-based building layout information into a geometric topology network for indoor navigation assistance. ASCE, 2011.

[36] Wouter Van Toll, Atlas F Cook, and Roland Geraerts. Navigation meshes for realistic multi-layered environments. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3526–3532. IEEE, 2011.

[37] A Vanclooster and Ph De Maeyer. Study of the applicability of cognitive path algorithms for 3d indoor navigation.

[38] Emily J Whiting. *Geometric, topological & semantic analysis of multi-building floor plan data*. PhD thesis, Massachusetts Institute of Technology, 2006.

[39] Li Yuan and HE Zizhang. 3d indoor navigation: A framework of combining bim with 3d gis. In *44th ISOCARP Congress*, 2008.

[40] Wenjie Yuan and Markus Schneider. inav: An indoor navigation model supporting length-dependent optimal routing. In *Geospatial Thinking*, pages 299–313. Springer, 2010.

[41] Wenjie Yuan and Markus Schneider. Supporting 3d route planning in indoor space based on the lego representation. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, pages 16–23. ACM, 2010.

[42] Lingli Zhao, Shuai Liu, Junsheng Li, and Haicheng Xu. Rapid acquirement and visualization of citygml documents. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 2, pages 1182–1185. IEEE, 2008.

[43] Sisi Zlatanova. On 3d topological relationships. In *Database and Expert Systems Applications, 2000. Proceedings. 11th International Workshop On*, pages 913–919. IEEE, 2000.