Database management of segmented interior point clouds of buildings

Martijn Meijers
(1007173)
Faculty of Civil Engineering and Geosciences
Delft University of Technology

11th February 2005

Abstract

In this report a thematic, formal model for classification of the interior of buildings is created. Firstly, a description of how objects can be reconstructed from laser scanning point clouds is given. Secondly, the thematic, formal model is described. Thirdly, how to store this model in a database is described.

At the moment, the need for an accurate description of the interior of buildings ('geo-information indoors'), e.g., for as-built information and Location Based Services (LBS), is increasing. However, up to now there are little research efforts on how to structure this interior, so it is usable for querying and storing in a computer. Hence, this research has been conducted.

The result of the research is a description of a model in UML with which the interior of a building can be described and a mapping of this model to a relational Geo-Database Management System (Geo-DBMS). The model is an effort on structuring 3D information of interiors of buildings and storing this information in a Geo-DBMS, ready for querying. Focus of the model is on the way the building is composed, i.e., how all parts of the building compose the complete building. The model can be extended for diverse applications (e.g. LBS or disaster management).

Acknowledgements

This report is the result of research done in the context of the course 'Research Project' at Technical University of Delft. A lot of time and energy have been invested to succesfully finish the project. It could have never be finished without the support of a few people who I would like to mention here explicitly.

First of all, I would like to thank Sisi Zlatanova. Thanks to her enthusiasm I was encouraged to do this research and the discussions we had, were very inspiring to me. I owe thanks to Norbert Pfeifer because of his input and suggestions during our discussions. I am grateful to Tahir Rabbani for his answers to questions on point cloud segmentation. During the laser scanning project at the Aula I collaborated with Rizqi Abdulharis, Jane van Ree and Walter Vroom (from my side with great pleasure).

Last but not least, thanks goes out to a number of people who supported me in a more indirect way: Koos and Jaap (my parents) and Fieke, for their ever present support and understanding even when my mind *again* was occupied with 'points', lines' and 'polygons'.

Thank you.

Naaldwijk, 7th Febuary 2005

Contents

List of Figures

Chapter 1

Introduction

When Maslow tried to describe what people need, he used a pyramid form to describe the hierarchy of these needs. The most basic needs, at the bottom, were physical – air, water, food, sex. Then came safety needs – security, stability – followed by psychological, or social needs – for belonging, love, acceptance. At the top of it all were the self-actualizing needs – the need to fulfill oneself, to become all that one is capable of becoming [Maslow, 1970]. Buildings offer a protection and because they are low in the pyramid of Maslow, they always have satisfied a need for people.

Nowadays, a description of buildings and information based on the form and layout of buildings is getting more and more important. This is not strange, because there is a growing need for this kind of information. As-built information (especially for governmental buildings and industrial installations) and analysis and querying of large, public buildings with respect to disaster management has gotten attention lately. Location Based Services (knowing your position anywhere any time, and interacting with information related to this position) is another push factor for creating indoor models.

In the Geo-information science either field based or two dimensonal (2D) object oriented approach is used mostly to describe objects in the real world. However, advances in technology have made a description with three dimensional (3D) objects feasible since the beginning of the early 1990's [Stoter and Zlatanova, 2003]. Complicated structures of indoors are hard to express in two dimensions (think of stairs on a floor plan of a building) and, stored in a computer in this way, hard to use for analysis and querying. The use of three dimensional models could accompany or even replace two dimensional information. Nowadays visualisation in three dimensions is quite common, but there is still a lack of models supporting storage, analysis and querying of 3D geo-information. The research up to now focusses on the outside world (e.g. creating city models [Zlatanova, 2000]).

The need arises to describe the indoor situation, for diverse applications, so that there is a seamless integration with the outside world (for example: [Gilliéron and Merminod, 2003]). Recent advances of laser scanning techniques allow already fast and economical creating of indoor datasets. A good deal of research has been also conducted on object reconstructing from these point clouds, like plane fitting and CSG fitting (see [Vosselman et al., 2004] and [Rabbani and van den Heuvel, 2004]). However, most of the research efforts are in general on the modelling part and not on the organisation of the point clouds and the reconstructed objects. In this research a formal, thematic model created from

interior point clouds for storage and querying of the interior of buildings is developed. The model is based on the classification of polygons and sections that form a building. As will be explained later (:TODO reference to section definition:), such a section corresponds in many cases to one room of the building. This classification model can be used (and extended) for diverse applications.

1.1 The interior of buildings

In this section the properties of a building that are important for modelling are described. The humans perception of buildings is taken as a starting point of view. Definitions of all the terms used throughout this report are given. Important aspects related to interiors of buildings are discussed:

- *Indoor versus outdoor.* A discussion is given on what to consider indoor and thus what to model. Indoor building models should be able to describe complex building structures. To perform complex (elaborated) analysis and visualisation of buildings interiors in three dimensions are needed.
- *Form and layout.* The classification model should support a way to implement the form and layout of a building. In this respect the concept of accessibility is critical. It should be implemented in such a way to be usable for diverse analysis and querying applications.
- *Mobility of objects.* Movable (tables, chairs) and immovable (walls, doors, etc.) objects are both important for indoor applications. In this research only immovable objects are considered, because of their long lifespan on only one location.

1.1.1 Indoor versus outdoor

Created by people, buildings function as the habitat of people, offering shelter and protection: protection against wild animals, bad weather, other people, etcetera. Most of the time this protection is offered, if you are inside the building, not outside in the open air. Therefore, the interior space of a building has a specific meaning.

Information if something resides inside a building, especially in large buildings, is important for many applications. Therefore, the structure of buildings is important. The information about interior is to be obtained from the structure of a building. To make a distinction between inside and outside a description of the boundary that separates the inside from the outside, is needed. In case of a building, this boundary is created by objects, like walls, doors and windows.

Building interiors have more complex structures compared to the world outside. A 2D or a $2\frac{1}{2}$D (field based approach: a single surface that can be modelled by $z = f(x, y)$) is most of the time sufficient for the outside world, except for some cases, like bridges, tunnels, i.e., multiple land use in general. A building can be seen as an aggregate of different spaces, like rooms, stairs, elevator shafts, etcetera. Inherent to this 'aggregate structure' *and* when a building has more than one floor, is that a $2\frac{1}{2}$D approach is not sufficient for describing interior, vertical relationships that are usable for analysis and querying. A split of the buildings in layers (floors) is needed to give a good description of a building in two dimensions. This is done for

example when floor plans are drawn for a building. Therefore modelling buildings indoor requires three dimensional structures in larger extends than when modelling the world outdoors.

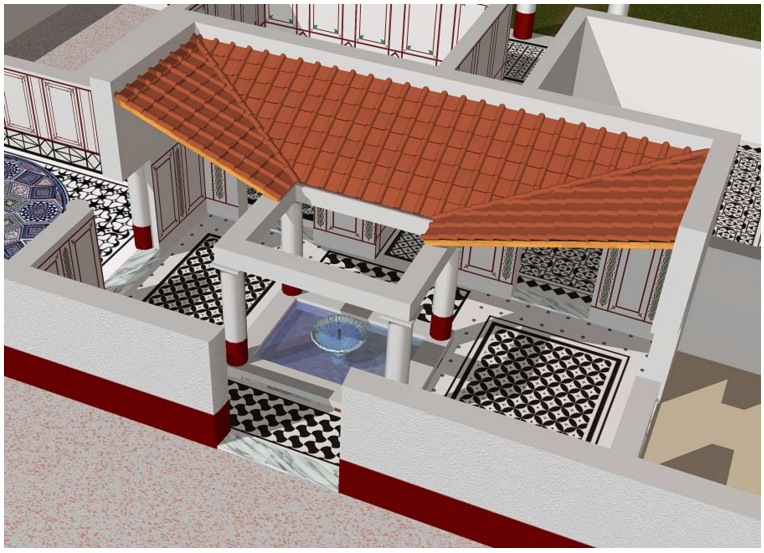


Figure 1.1: The impluvium of the atrium of an ancient building: inside or outside?

Figure 1.1 shows an example of a roman villa. The impluvium can be considered outside, because it is located in the open air, i.e., there is no roof on top. But, it does belong to the building, you can not reach it via another way, only by the interior of the building. This spot in the building also should be described with the model developed in this research. Another example are stairs that run on the outside of the building, for example for escaping the building in case of emergency. These examples show the complex structures that buildings have and that there is a thin line between inside and outside. In some cases it may be appropriate to consider some outer parts of the building also as interior, so they can be described with the same model. In this research a distinction is made between the following possibilities:

- **Outdoor** Located in the open air.
- **Indoor** Separated from the open air. Because this is too strict in some cases also parts that are connected to the building and residing outdoor, but meant to be reached via the inside of the building are considered to be indoor in this research.
- **Interior** A part of a building that is indoor.

1.1.2 Form and layout

Every building is divided into one or more sections. In general, a section is a group of spaces in a building, e.g. the left wing of the building. In this report we define a section differently:

- **Section** The smallest amount of bounded space in a building, that is related to the function this space has in a building. A section should be distinct and non-overlapping with any other section.

It should be noticed that most of the time the human point of view on the building is used. A section then represents a space with a name according to its function, for example a bedroom, a kitchen, an office, stairs, etcetera.

The diverse sections are bound by parts of the building (i.e., surfaces, for example floors, walls, ceilings, windows and doors). These parts are made of a particular material, like stone, wood or glass. The shape of these parts also differs. Here, the shape and size (the geometry), which these parts have and material, which these parts are made of, are called form.

Due to these boundaries the different sections have explicit relationships. The relationships between the sections are important for analysis. Often asked questions are where a room is located. Most of the time this question is answered with something like: "Go to the left here, take the elevator to the fourth floor and when you step out of the elevator the office is at the fifth door at your right hand." It is obvious that the space 'elevator' is related to the fourth floor. The relationships which the sections have are thus important. More generally stated, people are interested in the layout of the building, i.e., the position of the sections indoors and their relationships.

To sum up the terms:

- **Form** The geometry the parts, that form a building, have and the material which these parts are made of.

- **Layout** The relationships the sections have with each other. This can be derived from the form of the parts.

While creating a model for indoor use in buildings a model thus should support the concepts of form and layout *and* it should be usable for diverse analysis and querying applications. These applications are very diverse and can range from shortest path calculation for someone with a wheel chair to analysis on how many people an exit can handle for disaster management or visualising a certain change in layout of the building.

1.1.3 Immovable and movable objects

While creating a building model an interesting question is what is important for modelling. We distinguish between movable (chairs, tables, etcetera), immovable (walls, ceilings, doors, etcetera) and semi-movable (kitchen-cabinets, etcetera) objects. It seems appropriate to use only the frame of the building, i.e., the objects that are directly or indirectly attached to the foundations of the building. These objects will be called immovable objects. Opposed to immovable objects there are also movable and semi-movable objects within buildings.

In this research the immovable objects of the interior of the building are taken into account, for example walls, floors and ceilings. Doors and windows are also classified as immovable, although they can be opened. They can not easily be relocated somewhere else in the building. Immovable parts of the building enclose the interior and divide the interior into separate sections, as described in section 1.1.2.

Furthermore, these objects are likely to stay in place during their lifespan. That is why they are considered to be important from a modelling point of view.

- **Immovable object** Object directly or indirectly attached to the foundation of the building, likely to stay in place during its lifespan and not easily relocatable.

Opposed to immovable objects movable objects (cupboards, tables, plants and chairs) can easily be relocated somewhere else indoor (or even outdoor). Because these objects are movable, their exact location may vary from time to time. Movable objects may be interesting for analysis in certain applications. For example, a shortest path calculation may give different results when a room is filled with tables, compared to when a room is empty. These objects are left out of the scope of this research, because of the limited time span.

- **Movable object** Object not attached to the foundations of the building and easily relocatable.

To make this list complete also semi-movable parts of the building should be considered. A kitchen cabinet is a good example. Objects like semi-permanent walls are important for modelling. Such objects are unlikely to move, in the sense of being located somewhere else in the building, during their lifespan in a building, but they can easier be relocated than the real immovable parts of the interior, like walls and ceilings. Also these objects are considered to be imporant and are taken into account while creating the model.

- **Semi-immovable object** Object directly or indirectly attached to the foundation of the building, likely to stay in place during its lifespan, but reasonably relocatable.

1.2 Research approach

In this section a description of the research objectives and the approach that has been followed to achieve these objectives are explained.

1.2.1 Research objectives

Nowadays, a description of buildings and information based on the form and layout of buildings is getting more and more important. Recent advances of laser scanning techniques allow already fast and economical creating of indoor datasets. A good deal of research has been also conducted on object reconstructing from these point clouds. However, most of the research efforts are in general on the modelling part and not on the organisation of the point clouds and the reconstructed objects. Therefore, the **main objective** of this research is:

To build a formal, thematic model (an abstraction of the real world) for storage and querying of the interior of buildings.

This means that the geometrical parts of the interior of buildings (such as surface patches and sections) will be classified in a thematic way. The following questions will be answered to realise the objective:

- How to classify the interior of buildings?
  - How to classify geometry of interior of buildings with simple types (points, lines, polygons)?
  - How to subdivide the interior of a building?
- How to build a thematic hierarchy of the interior of buildings?
  - How to classify the thematic meaning of sections in buildings?
  - How to describe the interdependence of the formed sections?
- How to store this thematic hierarchy in a Geo-DBMS?

1.2.2 Research strategy

To answer the research questions, the following strategy is used.

Data collection and modelling

Literature study has been conducted on how point clouds can be gathered and how objects can be reconstructed from these point clouds. To store the reconstructed geometry three models have been investigated to show which of these three models suits the storage of the reconstructed objects best.

Indoor abstraction

The terms to describe the interior of buildings with have been described. With these terms a bottom up classification has been developed, herewith the interior of buildings can be described. The complete classification model has been described in the Unified Modelling Language (UML).

Database management of indoor abstraction

A short description of what database management contains is the result of literature study. To store the developed model for indoor abstraction, one has to make a mapping between both models to go from a UML model to a relational database management system. Literature study gave insights on how to make this mapping.

Case study

To gather data for building a prototype a part of the Aula of Delft University of Technology has been scanned with a laser scanner. The resulting point cloud has been stored in a Geo-DBMS. Software at the section of Photogrammetry and Remote Sensing (faculty of Aerospace Engineering) has been used to segment the point cloud into surface patches. These surface patches have been stored in the database management system. The model developed for indoor abstraction has been mapped to a database definition language, this mapping and results are described.

1.3 Organisation of the report

Chapter 1 (this chapter) shows the needs and background of the research. Definitions of terms used throughout the report are given.

Chapter 2 gives an overview of data collection and modelling. The topic of point cloud gathering with a laser scanner and how to create point clouds from the separate scans is described. An overview is given on how to segment a point cloud into shapes that describe objects better than the raw point cloud. The chapter concludes with a comparison of different geometric models that can work with the point cloud: Voxelization, Constructive Solid Geometry and Boundary-Representation are discussed.

Chapter 3 introduces the concept of indoor abstraction. A switch is made from (segmented) point cloud to object description, this is thus a step from geometry to thematic classification. Therefore, first the simple types of polygons are defined. After that a model is presented which relies on polygons, sections and complexes of sections. At the end of the chapter a UML Model that describes the developed model is presented.

As described in chapter 4 a geo-DBMS can be used for storage and analysis of the indoor abstraction. An overview is given regarding the topics of Database Management Systems with spatial data. Different architectures (dual, layered, integrated) are described. Discussed is why it is good to integrate spatial data into a mainstream DBMS. Aspects of spatial data that are and are not implemented in mainstream databases are described. To conclude the chapter issues are considered regarding the mapping of the object model for storage in a relational DBMS.

Chapter 5 evaluates the developed model via a case study. For this case study parts of the building of the Aula of TU Delft have been scanned with a laser scanner. The chapter elaborates on the indoor abstraction: segmentation with planes and adaptation of these planes to form sections. Bulk loading the dataset into a Geo-DBMS is described, as is mapping of the object model to relational tables.

Chapter 6 summarises the research and concludes the major findings. The chapter also contains some recommendations for furter research.

Chapter 2

Data collection and modelling

A building model should work with the data that is collected for a building. That is why in this chapter an in-depth look at point cloud gathering and methods for segmenting a point cloud will be established.

As described in section 2.1, point cloud gathering with a laser scanner is suited for indoor data collection. It is a fast method for acquiring point cloud data. Autonomous robots can be fully equiped to do laser scan jobs indoors. There are different ways to create one big point cloud from different scans resulting from laser scanning.

The point cloud itself is not sufficient for many applications. Point clouds often show large data volumes, which makes it hard to work with point clouds. Surfaces are easier to validate and better suited for analysis compared to the raw point cloud. Hence, a point cloud should be segmented in surfaces. This is described in section 2.2. These surfaces can then be used as basis for more applications. Segmentation of the point cloud can roughly be done in two ways. If only planar surfaces are to be looked for in the point cloud a segmentation method based on criteria like proximity of points and or similarity of locally estimated surface normals can be used. If more complex surfaces are to be looked for (like cylindrical surfaces), more advanced methods, like Hough transform, are needed.

In section 2.3 descriptions of different geometric models are given for storing the found surfaces and a comparison is made between the different models. The first method described is voxelization. This method is bound to lose information. With both Constructive Solid Geometry (CSG) based fitting or Boundary-Representation (B-Rep) creation one can end up with planar surfaces from which objects can be reconstructed and thematically classified. B-Rep is the most generic geometric modelling system, in the sense that CSG fitted solids can be converted to a B-Rep modelling system.

2.1 Collection of point cloud data indoor

Indoor point cloud data collection can be done with terrestrial laser scanners (for an example of such a measuring instrument, see figure 2.1). Laser scanning is a fast method for collecting point cloud data. More data can be collected within the same amount for time, compared to using more classical methods, such as measurements with a tacheometer.

However, the data collection process using a laser scanner is still a laborious task for human beings. Therefore, research has been conducted, that the measurements can be done fully automated by mobile robots, that navigate themselves, with a laser scanner mounted on top (see for examples [Nüchter et al., 2003] and [Biber et al., 2004]).

Besides laser scanning there are other ways of gathering point clouds. Van Gool gives a description of measuring a point cloud with video data:

> A grid is projected onto the object by the use of a flash, simultaneously an image is taken, and from the image, a complete surface patch is reconstructed in 3D [Gool et al., 2004].

A disadvantage for use of this technique in buildings is that quite a lot of texture is needed which cannot be found in interiors of buildings. It is thus hard to find appropriate tracking features (such as corners, points, etcetera).

Figure 2.1: Terrestrial laser scanner: a Cyrax 2500 by Leica Geosystems

Laser scanning is a measuring method where distance is derived from the time that a laser pulse (the signal) is on its way. Besides the distance, the angle of the laser ray that was emitted is also known. The location of the point that is acquired, is relative to the position of the laser scanner. Result of the scanning process is a point cloud, called a scan. A point cloud is an unstructured set of point samples [Èmolík and Uller, 2003]. One point sample is an elementary object, specified by its location derived from the distance and angle of the measurement in three dimensional space. A single point sample can be visualised as a small sphere or a point (pixel). The gathering of point cloud data of environments in three dimensions requires multiple scans.

The resolution and scanning time of the wanted point cloud are related. A trade off exists between the resolution and scanning time: a coarse resolution means quick scanning, but less detail. A higher resolution does mean more detail, but also means longer scanning time before the measuring process is finished. The resolution that is chosen is based on the use of the measurements (for which application the data are meant) and the wanted model that is going to be created with the measurements. This is comparable with terrestrial measurements for making maps, where what is measured depends on the wanted scale of the map that is produced. Point clouds

measured with a laser scanner will be large datasets, especially when scanned with high resolution.

To scan the interior of a building several separate scans are required. Each scan has its own local system (e.g., with the origin at the centre of the laser scanner). There are different ways to register all the scans in one large point cloud. Some possible ways are described in [Balis et al., 2004] and [Meijers et al., 2005].

- Different scans can be connected via the scanned points. Therefore, overlap between the two scans to be connected is needed. The Iterative Closest Point-algorithm (ICP), described in [Besl and McKay, 1992], can be used to connect the scans, by pointing out points that are the same in both scans. A necessity for this method is that there is overlap and the resolution of the scans is high enough, so the same features can be distinguished in the individual point clouds.

- Another method is that targets are positioned within the field of view of the scanner. Targets are, for example, half spheres, that can be rotated, so that they are visible from all directions. Half spheres offer the advantage that it are 2 different target types in 1. On one side the half sphere resides, while the other side is flat, here a different target can be placed on. The operator of the laser scanner points out the location of the target (acquired via an optic system accompanying the laser scanner, a photo camera, e.g.) and the laser scanner then tries to locate the target. If the target is recognised by the scanner, it is scanned with higher resolution; some scanners even use more laser beam power while scanning targets. For this approach it is needed that the targets are located in the overlap between the scans (and measured with higher resolution in both scans).

- The last approach described here is that there are targets in the scan area, that are already known in coordinates in an external reference system (measured with GPS or tacheometer, e.g.). For this method each scan is connected to the 'external' reference system, such as a national grid, i.e., the scan is being geo-referenced. For this approach it is needed that the targets are located within each scan (measured with higher resolution in the scan) and are already known in 'external reference'-coordinates.

- Also a combination of the methods above can be used. For example, first connect all scans together with the ICP-method, then do a registration to an external reference system, by usage of some known targets scanned with higher resolution in different scans.

After the measurement process and 'melting' the different scans together, one large point cloud is the result, which can be used for direct measurements in the point cloud. However, a lot of applications require the shape of the objects and therefore the point cloud only is not sufficient. In the next sections object recognition/reconstruction techniques will be explained. First, the process of point cloud segmentation will be explained, second, some models for storing objects reconstructed from the segmented point cloud will be described.

2.2 Point cloud segmentation

After creating the point cloud, the shape of the recorded objects is needed. The objects described by the point cloud have a certain shape. Because the shape of the objects is known, some assumption can be made to extract the surfaces out of the point cloud.

Planes are important for modelling the indoor parts of a building, because they can be used as a base for describing the interior of buildings. In man-made objects a plane is the most frequent used surface shape. Indoor walls can be assumed to be planar surfaces, for example.

A plane is a flat surface extending infinitely in all directions [Wikipedia, 2005]. Mathematically described, it is a plain, infinite surface, which has an infinite area. One plane divides a three dimensional space in two parts. In a three-dimensional x, y, z-coordinate system, one can define a plane as the set of all solutions of an equation:

$$ax + by + cz + d = 0 \tag{2.1}$$

In equation (2.1) a, b, c and d are real numbers such that not all of a, b, c are zero. Planes can also be described with normal vectors. For a point $P_0 = (x_0, y_0, z_0)$ and a vector $\vec{n} = (a, b, c)$, the plane equation for the plane passing through the point P_0 and perpendicular to the vector $\vec{n}$ is:

$$ax + by + cz = \underbrace{ax_0 + by_0 + cz_0}_{-d} \tag{2.2}$$

Vector $\vec{n}$ in equation (2.2) is called the normal vector of the plane passing through point P_0.

Besides planes, also other surfaces can be used to reconstruct objects. Certain objects are cylinders and elbow shaped pipes. In the petroleum industry, where objects are constructed from CAD models that use cylinders, it is very common to try to find cylinders in the point cloud.

According to [Vosselman et al., 2004] extraction of surfaces out of the point cloud roughly can be divided into two categories:

- Segmentation of a point cloud based on criteria like proximity of points and/or similarity of locally estimated surface normals. Generally, a measure of homogeneity, e.g. being similarity of curvature, is introduced to segment the point cloud.

- Those that directly estimate surface parameters by clustering and locating maxima in a parameter space. This one is more robust, but only can be used for shapes like planes and cylinders, that can be described with a few parameters, and there should not be too many shapes in the point cloud.

An assumption can be made, that the interior immovable objects can be represented, or at least approximated, with planar faces. Then segmentation of the point cloud is straightforward. Extraction of planes from the point cloud is wanted. The first case of possible segmentation methods can be used.

Segmentation is an effort to find structure in the point cloud. Surfaces are extracted by grouping nearby points that share some property, like the direction of

a locally estimated surface normal. A point cloud then is segmented into multiple groups of points that represent surfaces. The points that apparantly belong together after the segmentation process, can be used for fitting a planar surface through this group of points. A boundary on the fitted plane bounds the wanted surface. This boundary can be the convex hull created around the outer points projected on the plane. These planar surfaces, called polygons from now on, form the basis of the wanted model for describing objects and will be used in the following section (3D model). There are some other ways to come to the boundary of the surfaces, e.g., meshing the laser points or overlaying with a grid can be used to generate a better boundary, instead of a convex hull, although a convex hull is the most simple form to detect points on the outside of the polygon.

If a building is not describable by planes only, but the building should be described with a mixture of different models, like planes, cylinders and other complex objects, then segmentation just creates subsets based on some assumption. As a result the point cloud is either over-segmented, where each object is splitted across multiple segment or the point cloud is under-segmentated, where each segment of the point cloud contains more than one object. A method from the second mentioned option above is needed.

In the case of undersegmentation 3D Hough Transform [Vosselman et al., 2004] can be one way of separating points belonging to different models. Hough Transform is a model based voting scheme that tries to find a given model in the input data using a voting for each possible instance of the given model. This is of use in case of a point cloud, containing interior structures that can be described with different models.

2.3 Geometric modelling systems

In this section a description of three geometric modelling systems is given, because the segmented surface patches should be stored in a way that analysis and querying with the surface patches is possible. Information in the first three subsections is based on [Aguilera, 1998]. Although it is possible to measure directly in a point cloud, it is not comfortable to work with it directly. It is possible to segment the point cloud and describe objects with related surfaces. First, the models will be described in the following sections (2.3.1–2.3.3). A discussion of the best suited model to work with segmented point clouds is given in section 2.3.4.

2.3.1 Voxelization

Spatial-occupancy enumeration (also called exhaustive enumeration) is a special case of cell decomposition in which the solid is decomposed into identical cells arranged in a fixed and regular grid ([Requicha, 1980] quoted in [Aguilera, 1998], p. 2-20). The primitive cells can be used to describe objects. Each cell-decomposition system defines a set of primitive cells that are typically parametrized. These cells in 3D are often called voxels (volume elements), in analogy to pixels (which are 2D picture elements), as described in [Kong and Rosenfeld, 1989] (quoted in [Aguilera, 1998], p. 2-20).

Each cell may be represented by the coordinates of a single point, such as the cell centroid, the vertex with minimum coordinates, etcetera, and the cell size is given by the grid size. Usually a specific spatial scanning order is imposed.

2.3.2 Constructive Solid Geometry

Objects can be modelled as solids. Constructive Solid Geometry (CSG) is a scheme where simple primitive solids are combined by means of regularized boolean set operators that are included directly in the representation. An object is stored as an ordered tree with operators at the internal nodes, and simle primitives (such as cubes and cylinders) at the leaves. Some nodes represent boolean operators (union, subtract and intersect), whereas others perform translation, rotation or scaling [Aguilera, 1998], on p. 2-13 quoting [Requicha, 1980].

2.3.3 Boundary Representation

With a Boundary Representation (B-Rep) objects are described in terms of their surface boundary elements. Some boundary representations are restricted to planar and polygonal boundaries. The representation is a directed graph containing object, face, edge and vertex nodes.

Many B-Rep systems support only objects with a two manifold boundary, i.e., a boundary containing only two-manifold edges and two-manifold vertices. A two-manifold edge is adjacent to *exactly* two faces and a two-manifold vertex is the apex of only one cone of faces. However, for buildings non-manifold edges and vertices are needed, if for example a wall is represented with only one surface. In this case an edge can be adjacent to more than two faces and thus is non-manifold.

2.3.4 A comparison in relation to point cloud segmentation

All models above offer the possibility to convert from the point cloud to the given model. Which model is suited best for this conversion, i.e., can be used in case of analysis and querying, storing it in a database and is easy to convert from a point cloud? For analysis and querying the model should be easily translatable to a node-edge graph structure. In this graph the nodes will represent rooms and edges represent doors that grant access. In this way graph theory can be applied to the building and shortest path algorithms can be used for navigational purposes. It is thus an advance if a geometric model offers the possibility to convert to a node-edge structure.

Voxelization is a process of rasterization in three dimensions. The main aim is to simplify access to the measured data. With this process a 3D point cloud is converted into the three-dimensional grid domain. The cells are small cubes, which are called voxels. The size of the grid cells determines the resolution of the 3D grid. Usually, if a grid cell contains a laser point, it gets a value of 1, otherwise, if it does not contain any laser point it will get a value of 0. Somewhat more advanced is it when the number of laser points inside a voxel is assigned to that voxel [Vosselman et al., 2004].

However, when using this rasterization process with a point cloud indoors, it is bound to lose information. For this rasterization process the assumption that the

point density is the same everywhere in the point cloud is made. A problem herein resides, what is an appropriate bin size to use for the voxels? When the point cloud has no uniform point density on the surfaces, a loss of information appears on spots where density was low already: No voxels are created on these spots. Further more, if a too small bin size is used, access is not simplified and the goal of voxelization not met. If a too large bin-size is used, then one risks to lose the structure of the object that is wanted from the point cloud.

With Constructie Solid Geometry solids are used to represent the wanted objects. Combinations of basic solids are fitted within the point cloud. This is a nice effort, especially when the objects that one is looking for are designed with CSG based CAD systems. These objects (such as cones, cylinders and spheres) are not supported by B-Reps. Examples of these are found in the petroleum industry where pipes are designed in this way. However, it is difficult to segment an interior point cloud based on CSG solids, compared to the assumption of only finding simpler surfaces. Indoor spaces are not that easy to reconstruct from primitive shapes used with the CSG base method. A solution to this can be to take a step in between. First, find planar surfaces, and second, fit solids to the found planar surfaces, instead of manually editing and creating a B-Rep with the fitted planar surfaces. From CSG the boundary of the objects formed with the solids can be converted to a B-Rep, a Triangular Mesh or a Point Cloud [Rabbani and van den Heuvel, 2004].

In case of Boundary Representation the segmentation ends up with parts of faces of polyhedrons. An adaptation of the found surfaces is needed to accurately and completely describe the form of the objects. For this adaptation it is not possible to give an accuracy measure, especially not if there are gaps in the point cloud and the planar surfaces are extended manually (to their intersection boundaries, for example) so gaps are closed. The adapted surfaces form faces of polyhedrons. The faces can also originate from CSG based fitting, when a conversion from solids to a B-Rep model has taken place.

To sum up, the voxelization method is bound to lose information and not easy to validate, as is the same with the point cloud itself. Both CSG based fitting and B-Rep creation out of point cloud are possible. With both methods one can end up with planar faces. These faces can be taken as simple feature types (i.e. polygons) with which the interior of the building can be classified.

Chapter 3

Indoor abstraction

After measuring and segmenting the point cloud, a segmented point cloud (groups of points that do belong together to one object) is obtained. The first step of object reconstruction has been taken: planar surfaces or CSG objects have been fitted to the point cloud that describe the objects under study (described in chapter 2)

The geometry (i.e., points, lines and polygons, see page 36) found consist of several simple feature types (point, curves and surfaces) which represent only simple parts of the building (like doors, walls and windows). These simple parts can be combined into complex feature types to represent sections (like rooms or hallways) or even complexes of sections (e.g., the complete building). How to use the simple feature types to represent parts of the building can be seen as a classification problem: what parts should be represented with what sort of geometry? How to combine those simple types together into complex feature types? To cope with the complete building a hierarchy of the complex feature types has to be build, and other complex feature types are needed for this.

This chapter concentrates on two major questions. The questions are how to classify the interior of buildings and how to build the thematic hierarchy with this classification? To answer these questions, they are broken into smaller subquestions. In section 3.3 an answer will be found to the subquestion on how to classify the geometry of interior of buildings with simple feature types (points, lines, polygons)? Also the subquestion on how to classify thematic attributes of the interior of spaces of buildings is treated. In section 3.4 the subquestion is how to relate classified simple types to form complexer objects, describing the sections inside a building? How to describe the interdependence of the formed complex objects is described in section 3.5. The complete model is given in section 3.6.

3.1 Point cloud

A clear relationship exists between the segmented point cloud and the surface patches that can be formed after segmentation. The interior of buildings, and more important the immovable parts of a building, can be described with those surface patches. It is more appropriate to describe those parts of buildings with surfaces, compared to the point cloud, because of reasons like validity (validness of surfaces is easier to check than the validness of point clouds) and computation power: surfaces are easier to work with, than a point cloud with high density describing the same interior.

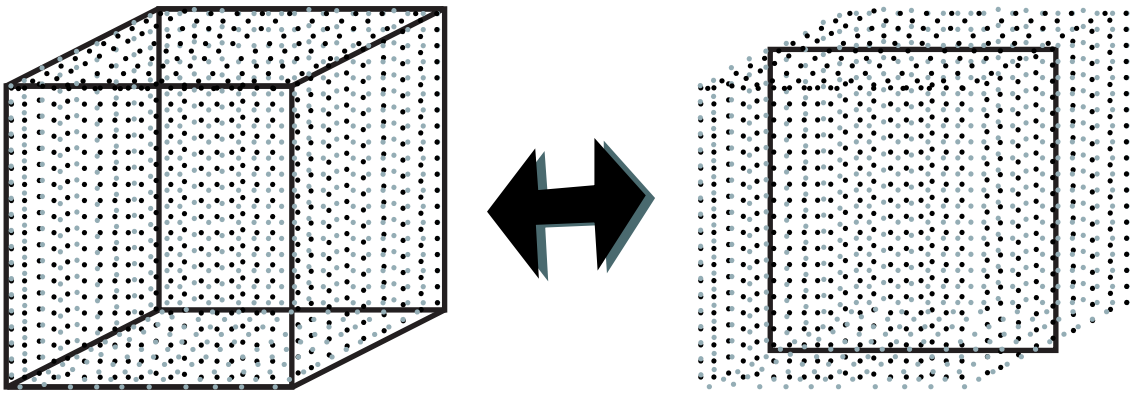


Figure 3.1: Different representations can be generated from the same point cloud

To classify the interior objects within buildings the granularity that fits analysis and querying applications is thus to choose for surface patches. However, the relationship between the points and surface patches should be maintained. A more detailed representation can be obtained easier, if a relationship in the model is maintained between the original point cloud and the surface patches. For example, when modelling thick walls this is the case. How to model those walls depends completely on the application. For a visualisation application one probably wants to model the thick wall with 6 separate polygons. Contrary, for a shortest path calculation application a choice for one polygon that is estimated in the middle of all the points is sufficient (figure 3.1). If the relationship in one model with the original points and the surface patches is maintained, this relationship can be used to select the original points and then generate other representations. This new representation is then still based on the original points.

3.2 Classification

As described above, the steps to create an object from a point cloud are as follows: points are measured, then the point cloud is segmented and surfaces are adapted to describe the objects (be it via a CSG approach or not). What remains is the classification of the surfaces of the sections that should be considered in the model. This classification can be done two ways: via deduction or via induction.

With deduction the classification is structured top down. The general principle is taken as a starting point for classification: from known terms unknown terms are derived. The complete collection is divided into classes based on criteria which were set beforehand.

The inductive way of classifying is a more pragmatic approach. Induction, discovering a general principle from a set of facts (bottom up approach), is used to come to the classification. For each item of a collection similarities to the other items are tried to be found. If there are similarities between items a group of those items is formed, if there are no similarities comparison goes on with the remaining items or a completely new group is formed.

In the next sections a bottom up classification is described. First, examples are given from which the general terms for classification are derived.

3.3 Polygons

As mentioned in section 2.3.4 when a point cloud has been segmented, either via planar surfaces that are manually adapted, or with CSGs that are converted to a B-Rep, we can end up with planar surface patches. These surface patches can be adopted as a base for modelling.

3.3.1 Definition

With the choice to adopt a planar surface patch as base for modelling, an important question arises on what geometry is allowed for such a surface patch. A planar surface patch is called a polygon in the field of geometry. Several definitions of polygons have been given, for example by ISO, the Open Geospatial Consortium (OGC, a non-profit, international, voluntary consensus standards organisation) and by researchers at the Geo-Database Management Center (GDMC) at TU Delft.

Although it is somewhat dangerous to quote without background from the ISO specification, because the ISO geometry specification is a model in itself, a polygon in this specification is defined as follows:

> A GM_Polygon is a surface patch that is defined by a set of boundary curves and an underlying surface to which these curves adhere. The default is that the curves are coplanar and the polygon uses planar interpolation in its interior [ISO/TC 211/WG 2, ISO/CD 19107, 2003], p. 78.

The implementation specification of the OGC is less dangerous to quote from. Although focus is on two dimensions, there are some pointers for use of polygons in three dimensions:

> The OpenGIS Abstract Specification describes a simple Surface as consisting of a single 'patch' that is associated with one 'exterior boundary' and 0 or more 'interior boundaries'. Simple surfaces in three-dimensional space are isomorphic to planar surfaces. Polyhedral surfaces are formed by 'stitching together simple surfaces along their boundaries, polyhedral surfaces in three-dimensional space may not be planar as a whole. The boundary of a simple Surface is the set of closed curves corresponding to its 'exterior' and interior boundaries. The only instantiable subclass of Surface defined in this specification, Polygon, is a simple Surface that is planar [Open GIS Consortium, Inc., 1999], p. 2-7.

A polygon is a subclass of the class of surface objects. Six assertions are mentioned in the specification that should be followed to make valid polygons.

> A Polygon is a planar Surface, defined by 1 exterior boundary and 0 or more interior boundaries. Each interior boundary defines a hole in the Polygon [Open GIS Consortium, Inc., 1999], p. 2-8.

In [van Oosterom et al., 2003] a polygon is defined a little different, because the above definitions or assertions given together with the definition are ambiguous. Their goal of giving a definition of a polygon is to make the definition unambiguous, so polygons can form a stable foundation of spatial modelling. The definition reads:

> A polygon is defined by straight line segments, all organized in rings, representing at least one outer (oriented counterclockwise) and zero or more inner boundaries (oriented clockwise). This implies that all nodes are at least connected to two line segments and no dangling line segments are allowed. Rings are not allowed to cross, but it is allowed that rings touch (or even partially) overlap themselves or each other, as long as *any point inside or on the boundary of the polygon can be reached through the interior of the polygon from any other point inside the polygon*, that is, it defines one connected area. As indicated above, some conditions (e.g. 'ring touches other ring') require a tolerance value in their evaluation and therefore this is the last part of the definition.

The orientation of the rings is important, because traversing the boundaries gives knowledge on where the interior of the polygon resides. If a 2D polygon in 2D space has been structured according to the definition, the outer ring is oriented counterclockwise. If the ring is processed clockwise, the interior of the polygon is always on the right side of the ring. In 3D the notion of orientation, clockwise or counterclockwise, is not sufficient. A convention is needed to specify these directions. A convention can be to order all the vertices of outer boundaries counterclockwise, seen from the outside of an object and the vertices of inner boundaries clockwise. Then, the normal vector is pointing to the outside of the object.

The last definition is adopted in this research and polygons are used as simple feature types to describe the interior of buildings.

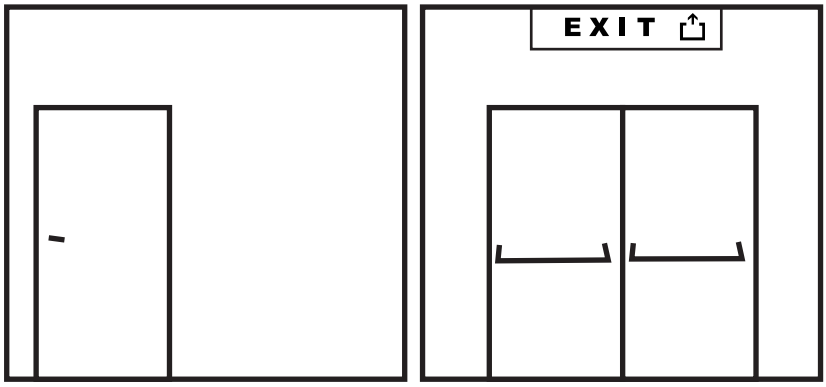
3.3.2 Examples

One can come up with a rather large list of functions that objects inside buildings have. Different meanings that a polygon that resides indoors can have are for example: wall, window, door, door opening, portal, exit, emergency-exit, entrance, ceiling, stair, side of a lift shaft, entrance of an air cooling shaft, ..., and so on. All these polygons serve different functions. With the diversity of those simple examples it is already clear that the classification of the polygons is quite complex.

Figures 3.2(a) shows a typical example of a part of a building; a wall with a door in it. The door makes it possible to enter, while the wall prevents from entering the section. If there is no lock on the door, this is always the case. But if the door can be locked (and it is), then it is not possible to enter the other section, except when one has been granted access (has the right key or access code) of the door. The door is a two-way opening, i.e., you can use it bi-directional.

In figure 3.2(b) an emergency-exit is shown. This exit is only meant to be used, in case of an emergency. The door grants a two-way opening. The rest of the time the object is locked. If it is open, it is supposed to only leave the building via such a door. However, it is possible that one enters a section via the emergency exit if it is open. A slightly different case of a one-way opening is a door that can only be opened on one side with an access code. An alarm sounds if motion going the wrong way is detected, while the door is open.

Figure 3.3 shows two other cases. Figure 3.3(a) shows a wall with a door and a fixed window. This window is fixed, and cannot be opened. Only if it is demolished it grants access to the section adjoined to the window. If it has been demolished it grants access in two ways. Figure 3.3(b) shows instead of the fixed window an



(a) Door, that is openable

(b) One way openable emergency exit

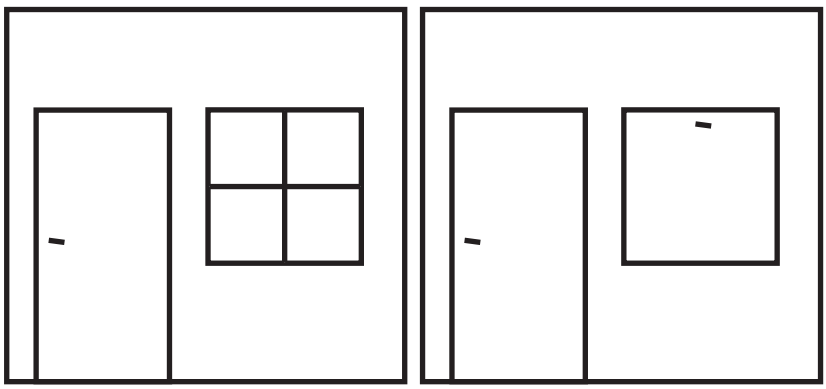
Figure 3.2: Some doors

openable window. Although the window is openable, it is not meant to function as an entrance or exit. However, in case of emergency the window can grant access in two ways.

Up to now, the examples showed real object properties. In figure 3.4 a wall with a hole in it is shown. This case can be compared with the case shown in figure 3.2(a). Compared to the door there is the same problem of accessibility. However, there is no way to lock the section behind the portal. To make the relationship of the different sections explicit, the the hole must be modelled as well.

3.3.3 Classes

In this section we give definitions for the classes of polygons (and their properties) that can be used for classification of the different objects in the interior of the build-



(a) Door in wall, besides a fixed window

(b) Door in wall, besides an openable window

Figure 3.3: Doors and windows in a wall

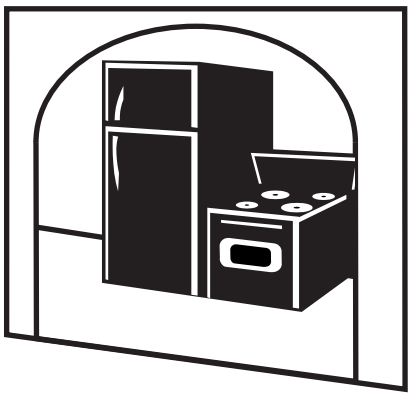


Figure 3.4: Virtual polygon for portal

ing. Only interior, immovable and semi-immovable objects that create the layout of the building, are taken into account for classification as described in sections 1.1.1 to 1.1.3. The form (i.e., the material the object is made of) can be stored as a property with a polygon. The following criteria are used for creating a classification of these objects:

1. persistence;
2. existence;
3. granting access and
4. types of passing.

Persistence

Important for constructions that are semi-permanently available in a building is persistence of these constructions. An example is a sliding door that can divide a section in two parts, so that the section becomes two separate sections. This notion is only necessary for polygons that are not persistent their whole lifetime. To distinguish the two classes of polygons we define the concept of persistence:

- **Persistent polygon** The polygon is there all the time.
- **Non-persistent polygon** The polygon can be removed from time to time.

For non-persistent polygons we need a notion if the polygon is there at the moment. This can be stored in a property of the polygon on which functions (e.g. `SetThere()` and `GetThere()`) will work. These functions can change the state of the polygon to true, if the polygon is there at the moment, or to false, if the polygon is not there at the moment.

- **Being there** The polygon is there at the moment.
- **Not being there** The polygon is not there at the moment.

Existence

To make relationships between sections explicit, it is not sufficient to only model objects. Also some parts of space, not taken by any objects forming the sections, should be modelled. The reality of the polygon does create two classes of polygons:

- **Real polygon** A polygon being there in reality *and* in the model.
- **Virtual polygon** A polygon *not* being there in reality, but being there in the model only.

This way relations sections have, can be distinguished and functions sections have can easier be separated and described.

Granting access

On basis of the criteria that a polygon grants access to a certain section, the polygons can be split in two classes:

- **Non-granting polygon** The polygon does not grant access to any section. It even prohibits entering the section which it is adjoined to.
- **Granting polygon** The polygon does grant access to a section.

For example, walls are to be represented with non-granting polygons. The door in the wall, not having a lock is of the class of granting polygons.

Further, the granting class can be split up in 3 different classes, because the sorts of granting acces can be different from polygon to polygon. First, there are full-granting polygons. A full-granting polygon is a polygon that grants access under all conditions. Doors having no locks, or virtual polygons are to be represented with such a polygon. Besides full-granting polygons there are semi-granting polygons that grant access to a section under common conditions. Doors having locks are to be represented with semi-granting polygons. The condition is that it only grants access when it is unlocked, or when it is locked and one has a possibility to unlock it. The last subclass of granting polygons is the class of limited granting polygons: there is a barrier to use this kind of polygons, only in special situations. Emergency exits and windows belong to this class. Under normal operation these polygons will not be used as entrance or exit, but under uncommon conditions, e.g. an emergency, these polygons can grant access to sections. To sum up:

- **Full-granting polygon** The polygon grants access under all conditions.
- **Semi-granting polygon** The polygon grants under certain common conditions access to a section.
- **Limited-granting polygon** The polygon does only in special situations grant access: a limitation condition has to be fulfilled before these polygons can be used.

Next to the subclasses of granting polygons, there is a relationship with the reality of the polygons. A non-granting polygon can only be real. A polygon in the model and not in reality will never disable access. It thus only makes sense to use a real polygon for the class of non-granting polygons, while with the class of granting polygons real *and* virtual polygons can be used.

Types of passing

Some notions on the polygons are only applicable to the class of granting polygons. The case of the one-way exit shows that the direction of granting access is important. Because it is nonsense to have such a property on a non-granting polygon (it would be always zero-way) this property is only allowed on the class of granting polygons.

- **One-way** Only from one side to the other; uni-directional.
- **Two-way** From one side to the other *and* vice versa; bi-directional.

3.4 Sections

In the previous section terms for classifying polygons were defined. The question is now to classify thematic attributes of the interior sections of buildings and how the objects that form the sections relate to the sections.

3.4.1 Definition

The sections in a building can also be classified. From a human being point of view, one way of classifying can be to classify the sections with the function they have. The function is a property of the sections we talk about. This function relates to what the section is used for in the real world. This can be a rather large list, and not very suited for a general way of storing information on buildings in a computer, because it is very heavily depending on the application which functions are there in the model. However, if a section is defined as the smallest amount of bounded space in a building, that is related to the function this space has in a building, it is clear what to consider a section (and is in accordance with section 1.1.2)

A more general way for classification of the sections seems to be appropriate, although the function a section has should not be forgotten. A distinction that can be made easier, and thus can function as a criterion for classification, is based on the function that the polygons have that create the section instead of the function the section has itself.

From the reconstructed polygons the sections are formed. The sections are represented with polyhedrons. A polyhedron is a bounded subset of three dimensional Euclidean space (E^3) enclosed by a finite set of plane polygons such that every edge of a polygon is shared by exactly one other polygon (adjacent polygons) [Preparata and Shamos, 1985] quoted by [Aguilera, 1998], p. 2-5. The vertices and the edges of the polygons are the vertices and edges of the polyhedron; the polygons are the faces of the polyhedron. The faces of the polyhedrons are thus formed by the classified polygons.

3.4.2 Examples

As described in section 3.2 the building section shown in figure 3.5 should be represented with 6 real, non-granting polygons (the walls, the ceiling and the floor) and 1 real, full-granting polygon (the door), if assumed the door cannot be locked. The section can only be entered and left via the full-granting polygon, another possibility simply does not exist. The situation would change, if there was in the opposite wall

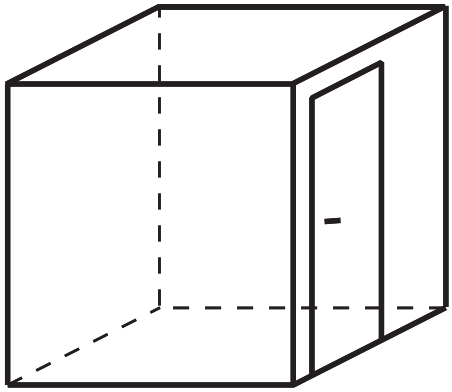


Figure 3.5: A section being a meeting room with a door

of the door (on the left side) another granting polygon. The section could then be entered via one door and be left via the other.

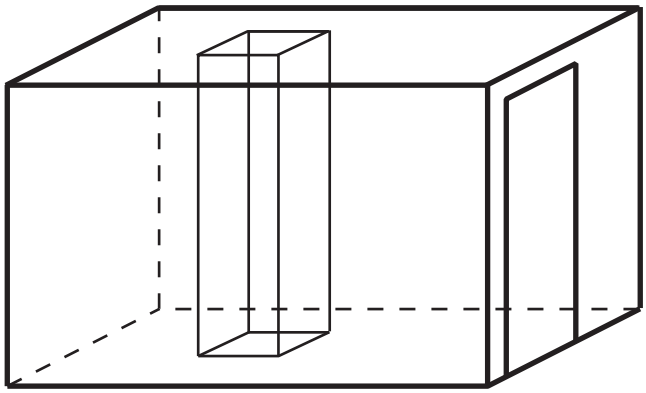


Figure 3.6: A pillar in the room

In figure 3.6 a section is shown with another section in it. The section that is contained in the other section is a pillar. The interior of the pillar can be solidly filled or filled with air. However, it is not possible for a human being to enter the inside of the pillar. This is also related to the polygons that create the pillar. In this case the pillar is bounded by 6 real, non-granting polygons, which explains why this section is not enterable by human beings.

Figure 3.7 shows an example of two rooms connected to a hall way. The hall way (on the right side) can be described as follows: there are 5 real, non-granting polygons (2 left walls, 1 right wall, 1 ceiling, 1 floor), 2 virtual, full-granting polygons (both sides are open) and 2 real, full-granting polygons (the doors granting access to both rooms). This resembles the example of figure 3.5 expanded with one door on the other side. The granting polygons make it possible to enter the hall way via one way and leave it via another, different way.

In the examples mentioned above, the access is only granted on the same floor. However, the approach using the granting polygons can be extended for a situation with multiple floors. A stairs can be modelled as is shown in figure 3.8(a) and 3.8(b). Two virtual, full-granting polygons allow one to access the section of the stairs and leave it. The first virtual polygon is on the front, the second is on the top. On the

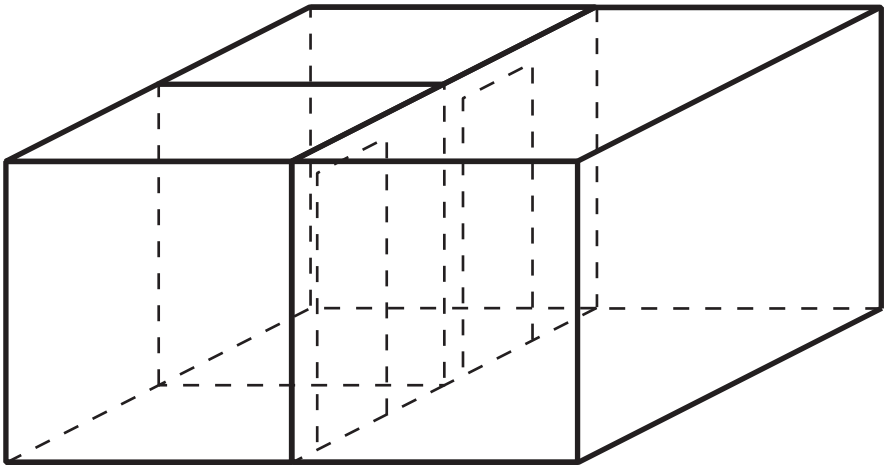
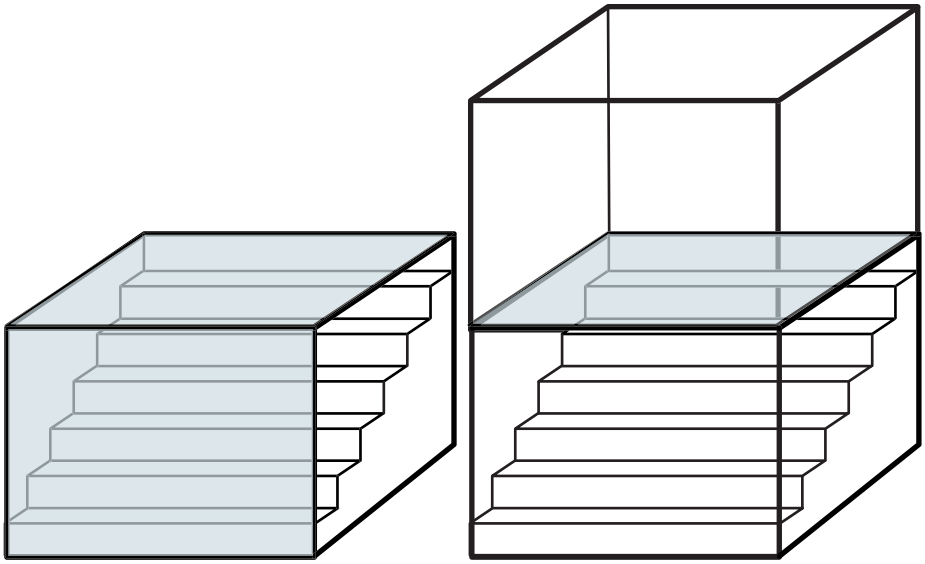


Figure 3.7: The hall way connects both rooms



(a) Stairs on one floor (b) Stairs below a floor

Figure 3.8: Stairs

upper floor (where one is connecting to with the stairs) there should be a horizontal polygon (the same as the top polygon) and a vertical polygon, both virtual (as is shown in the top part of figure 3.8(b)). The stairs in this way is a separate section. Because there is more than one granting polygon, be it real or virtual, it is possible to leave the section via another way than the one via which one entered the section.

A somewhat difficult example is shown in figure 3.9. Question is how to model the two rooms, while they also can be one room, if the wall in between can be removed, i.e., it is a non-persistent, real, non-granting polygon within the section. The two rooms from the example should be modelled as one room. This one room is to be modelled with persistent polygons. One or more non-persistent polygons only can be contained in a section that is bounded by persistent polygons: this way

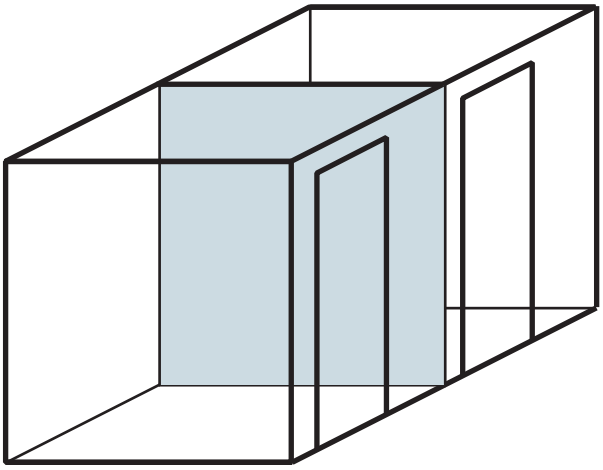


Figure 3.9: Two rooms, or one room?

it can divide a section into multiple sections. The section thus can be dynamically changed from one sort of section (with only one entrance, also being the exit) to another section (that has two distinct entrances).

3.4.3 Classes

Classification of the sections based on the function the sections is really depended on the application and can not be used in general for storage of the information of a building in a computer, although the function a section has should not be forgotten. What to consider a section is defined by its function. After stating what to consider a section, classification of the sections based on the relationship of the sections with their parts looks very promissing for storing information on buildings in a computer. Three sort of sections can be distinguished: not accessible, end and connector sections.

Some sections are not accessible at all for humans. For example, the interior of a pillar, can not be accessed by human beings. It can be solidly filled, or even open, but still not accessible. These sections are bounded by non-granting polygons only.

Some sections in a building are only accessible via one door and should be left through the same door. No alternative ways are there for entering or leaving this section. This means that the section is bounded by non-granting polygons plus exactly one full-granting or one semi-granting polygon.

Some sections in a building are accessible via more doors, so they can be entered via one door and be left via another. It is possible that such a section creates a connection between other sections. This connection can be horizontal, then the connected sections are on the same level, yet it also can be a connection with sections that are on different levels (a vertical connection).

To give a summary of the classes:

- **Not accessible section** Section bounded by only non-granting, persistent polygons, thus no granting ones.
- **End section** Section bounded by non-granting, persistent polygons, plus exactly one granting polygon (be it a granting or semi-granting polygon).

- **Connector section** Section bounded by non-granting, persistent polygons, plus at least two granting polygons.

A section is to be modelled with persistent polygons. Non-persistent polygons only can be contained in a section that is bounded by persistent polygons: this way it can divide a section into multiple sections. The section can be dynamically changed from one sort of section (end section) to another section (connector section).

3.5 Complexes of sections

When one speaks of buildings, often functions of sections are mixed to form new notions of parts of buildings. For example, the 'left wing' of a building can exist of several different sections, that have their own function. Also the notion of a floor is an aggregate of different sections. A building model should be advanced enough to work with those concepts, besides the sections alone. A question arises on how to build a thematic hierarchy with the basic sections.

3.5.1 Definition

It should be enough to have sections to describe all the parts of a building. If subsections are needed, the defined functions do not fit the granularity of the application. In this case the subsections should be translated to sections, because the definition of a section is not used correctly (a section is related to a function and uses the smallest amount of non-overlapping space valid for this function).

With sections groups of sections can be formed. Such a group is called a complex. This grouping can be done in two ways: grouping by function the sections have, or by location the sections have.

A complex of sections (i.e., a group of sections) can be formed by sections having the same function, like the function of 'meeting room' e.g. Although selection of sections with a certain function is useful, it is not a good criterion for grouping of sections. Grouping can be done better based on the location the sections have and the relationships they show. The grouping then has a relation with the location and the relationship the individual sections have.

A restriction on grouping of sections is thus that the sections are adjacent, before they can be grouped. Adjacency can be defined by a function on the geometry of the polygons that form a section, with a certain tolerance value. It also can be defined on the relationships sections have, through there parts.

For storing the sections and groups of sections that belong to a building, a rooted tree can be used. The term tree stems from graph theory. A **tree** is defined as:

> An undirected simple graph G that satisfies any of the following equivalent conditions: G is connected and has no simple cycles; G has no simple cycles and, if any edge is added to G, then a simple cycle is formed; G is connected and, if any edge is removed from G, then it is not connected anymore; any two vertices in G can be connected by a unique simple path. :TODO wikipedia reference:

A tree is a **rooted tree** if one vertex has been designated the root, in which case the edges have a natural orientation, towards or away from the root.

Terms we need for creating a hierarchy are:

- **Complex of sections** A group of sections, in which the sections are adjacent.
- **Tree** A tree is a graph in which any two vertices are connected by exactly one path.

3.5.2 Examples

Lets look at the example of an elevator shaft as is shown in figure (:TODO picture of elevator shaft:). If you need to split up the elevator shaft in different parts, because you want to visualise only a part of the building, then the function of elevator shaft did not fit the granularity of the application. Parts of the elevator shaft are needed for the application, so the model should give a way to handle those parts. It does, via the way of using sections, which are defined to be the smallest amount of space that is needed for a certain function. In this case an elevator shaft can be split into elevator shaft parts, and the parts can be grouped together into one elevator shaft, while the parts itself can be used with the notion of a floor.

The term elevator shaft relates to a vertical grouping of elevator shaft parts. Also stairs relate to a vertical group of 'stairs-parts'. The term floor relates to a horizontal grouping. Besides those two sorts of grouping, we can also group in both directions (horizontal as well as vertical). An example is when the sections on the left side of the building with two floors are grouped together and are called 'the left wing of the building'.

3.5.3 Classes

There is a need for groups of sections, such as floors, stairs, elevator shafts, etcetera. The building thus is a composite of sections and groups of sections. This composite should be non-overlapping, i.e., all sections should only appear once in the complete composite.

However, some applications may require that a building can be represented by more than one composition. To be able to do this another class is needed. In this case, a building can be represented with a rooted tree. The root node represents the complete building, while the other nodes can be formed by complexes of sections or sections. Each tree has a relationship with the building.

Overlap in the tree is not allowed, a section can only show up once in the complete tree. The leaves of the tree should be sections (the smallest units having a function in a building). With the notion of a tree, a section can appear in multiple trees that belong to a building. For each building there should be at least one tree, which contains all the sections.

3.6 The complete model

A class diagram from the Unified Modelling Language (UML) has been used to depict the classification that has been described in this chapter. A class in UML encapsulates all the objects that share common properties in the context of an application. In a UML diagram a class is drawn with a rectangle divided in three parts. The top of the rectangle shows the name of the class, the second part shows

attributes (data) and in the third part operations (behaviour) of the class are shown. The relationships that the classes have are depicted with associations. These associations are drawn with a line, with a certain ending. For a more indepth intro to UML see :TODO reference:.

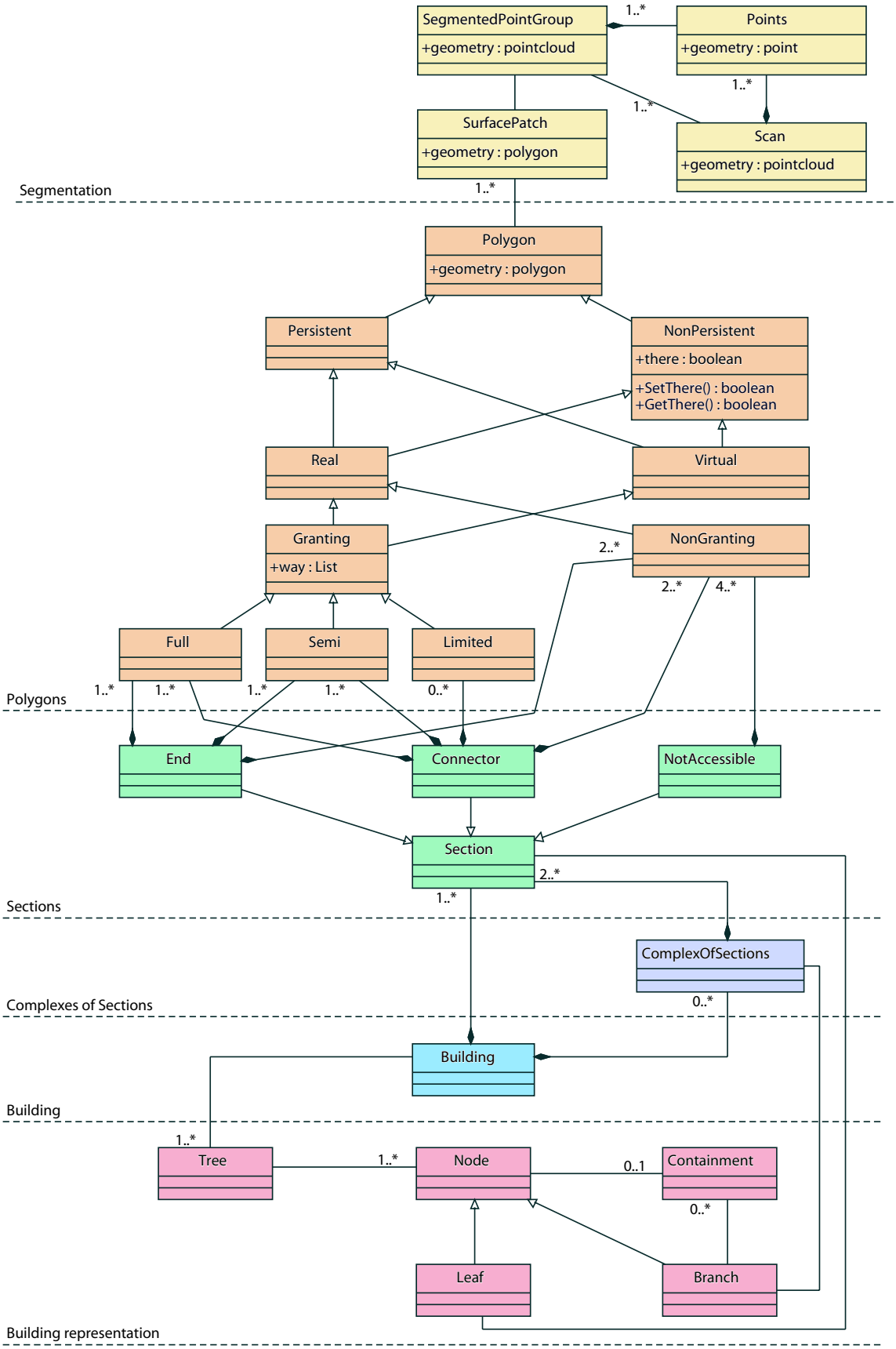


Figure 3.10: UML Model showing abstraction indoor

Chapter 4

Database management of the indoor abstraction

After classifying indoor surface patches, sections and complexes of sections, storage for the classified objects and the point cloud is needed. A Geographical Information System (GIS) is suited. Because database management systems (DBMSs) offer certain functionality, GIS-functionality has been integrated into mainstream database management systems, so this functionality has not to be implemented twice. This integration has evolved via diverse architectures, which is described in section :4.1 TODO: Also the mathematical concepts that are used for storage of geometry are discussed, as well as the current state of implementation of these mathematical concepts in Geo-DBMSs.

Then possible approaches for storing the indoor abstraction model are discussed in section :4.2 TODO: An implementation of storage of geometry in a mainstream DBMS (Oracle with Spatial Cartridge) is described, because this DBMS is used in the case study. Rules for mapping of an object model to a relational implementation is at the end of section :4.2 TODO:.

4.1 Spatial in DBMSs

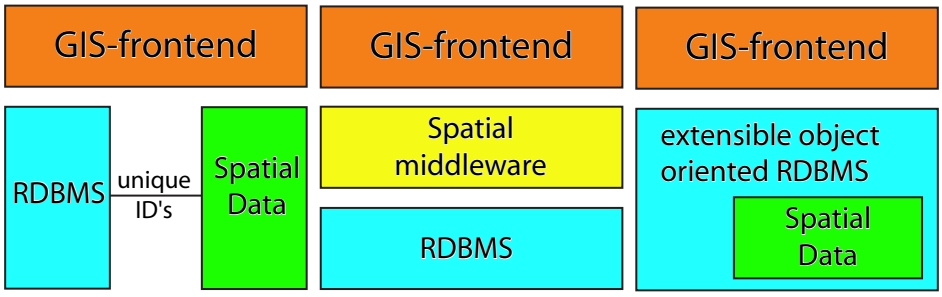
A Geographic Information System (GIS) is defined by Worboys as a computer based information system that enables capture, modelling, manipulation, retrieval, analysis and presentation of geographically referenced data [Worboys, 1995].

Nowadays, the representation of spatial objects has been integrated from separate GIS systems into mainstream database technology. Without pretending to be complete, the functionalities that DBMSs offer contain: concurrent updates, multiple views on the same data, security and access level for different users and a multi-user environment (with locking possibilities of data to ensure data integrity). An integration of GIS and DBMS will make it possible to use all the benefits of DBMSs functionality in a GIS, without implementing features twice. A DBMS with spatial functionality integrated is called a geo-DBMS.

According to the definition of GIS, it has to be capable of representing spatial objects. This is also the case for a geo-DBMS. The first step is by having data types and operators for simple features (i.e. geometric primitives) [van Oosterom et al., 2002]. Therefore a data definition and a query language for spatial data are needed in the database engine. With these languages data can be inserted, updated and deleted

and spatial querying (e.g., are two objects overlapping) and validation of data is possible. To make the query processing run fast, it must be possible to create spatial indexes and use clustering techniques for the spatial data (several spatial indexing techniques are shown in [van Oosterom and Vijlbrief, 1996]). The database engine should make use of these indexes and clusters. To optimise the speed of the system a query is mostly split in two parts: a filter operation, where a rough answer is found solely based on the index placed on top of the objects, and second, an operation based on the exact geometry of the objects giving a more accurate answer.

The adoptation of GIS in mainstream DBMS technology meant gradually changes in architecture of GIS systems. Different architectures have been adopted and are shown in figure 4.1. The architectures evolved since the second half of the 1980's from hybrid systems (dual architecture), to mid 1990's spatial cartridges (layered architecture) and, since the late 1990's and beginning of the 20th century, to an integrated architecture.



(a) Dual architecture

(b) Layered architecture

(c) Integrated architecture

Figure 4.1: Three different GIS architectures [van Oosterom, 2001]

With a dual architecture spatial object representation is broken into two pieces. A spatial subsystem is used for storage and querying of spatial data, besides a separate DBMS for administrative data. The link between the two systems is maintained through the use of unique id's. This architecture means freedom to use efficient data structures and algortihms in the spatial subsytem. However, queries for such a system must be decomposed into two parts, so it is complex to do the query processing and no global query optimisation is possible. Either the spatial or the standard index can be used, but those two can not be used together, because they reside in separate subsystems. Examples of commercial systems and research prototypes are: ESRI's ARC/INFO, SICAD and Ooi (:TODO references:).

A layered architecture stores all data in a single DBMS with the spatial engine responsible for generating spatial knowledge contained in a middleware layer between the application and the DBMS, separate from the DBMS itself. ESRI's ArcSDE is an example of this architecture.

An integrated Spatial DBMS Architecture means that spatial data types are integrated within a normal database system. With such an approach there is no difference between 'standard' and spatial data, because both data types are available. Indexing of both data types is possible, because for normal data a B-tree is available and for spatial attributes an R-tree. Operations for spatial data are em-

bedded within the database. Prototypes that have been built based on extensible database systems are: Gral, Monet and Geo++. Commercial solutions offering spatial functionality via the integrated way are Oracle Spatial Cartridge, Informix with Geodetic Datablade and IBM its DB2 with Spatial Extender.

Different mathematical concepts can be adopted for storage of spatial data. The concepts of geometry, the concepts of topology, or a hybrid approach of both concepts can be used for storing. The concepts of geometry and topology can be defined as follows:

- **Geometry** Geometry (from the Greek words geo = earth and metro = measure) is the branch of mathematics dealing with spatial relationships. It was the first field to be put on an axiomatic basis, by Euclid. A significant development was analytic geometry. Analytic geometry imposes a coordinate grid on the space, making it possible to study geometric objects (e.g., lines, parabolas, and circles) by means of algebra (e.g., linear equations and quadratic equations) and vice versa. :TODO after Wikipedia:

- **Topology** A branch of mathematics concerned with the study of topological spaces. Topology is concerned with the study of the so-called topological properties of figures, that is to say properties that do not change under bicontinuous one-to-one transformations (called homeomorphisms). Two figures that can be deformed one into the other are called homeomorphic, and are considered to be the same from the topological point of view. For example a solid cube and a solid sphere are homeomorphic. (:TODO after wikipedia:)

With both concepts a distinction between several primitives has to be made. The primitives used in geometry and topology have different names, so they can easily be distinguished. In this research the following conventions are used:

- Geometrical concepts: point, curve, surface and solid.

- Topological equivalent concepts: node, edge, face and volume.

Both concepts offer their own strong and weak points. Storage via geometry is good when features are going to be used for metric and specialisation operations, i.e. (metric) calculate an area, length or distance, (specialisation) calculate a buffer, the centroid, convex hull, intersection or difference of an object or between objects. In contrast, storage via topology should be used when data integrity is of concern (e.g. features must form a non-overlapping space partition in 3D) and spatial queries (e.g. adjacency of certain features) relying on topological references are used.

Standardisation of the geometrical terms has resulted in having simple features and operators for these features. In 2D the geometry standards have quite matured. Standardisation has not been done for 3D, although some specifications also give pointers on how to store 3D geometry. However, storage of 3D elements is possible (for example, systems that offer storage for 2D primitives in 3D space are PostGIS and Oracle). Albeit possible to store 3D elements, often still not all functionality, like querying in three dimensions, is working correctly; this is work in progres.

In for topology in 2D there is no implementation standard yet. Although recently some implementations have come available in commercial systems focussing on topology in 2D (Oracle and the Radius Topology engine on top of Oracle), there

are no commercial solutions available for storing 3D topology. At the moment user-defined solutions by using the relational system or extending the database engine with scripts offer a possibility for storing 3D topology (see [Stoter, 2004], p. 159).

4.2 Usage of a relational DBMS

The indoor abstraction is described in terms of objects. Database management systems store and manipulate information. To store the data often a relational approach is used, which has strong mathematical foundings. The relational model of database managament was proposed in [Codd, 1970] at IBM Research Laboratory. A relational DBMS represents all information in the database as tables and it supports three relational operations: selection, projection and join. With these operations it is possible to specify exactly which data one wants to see.

This approach however differs from the object oriented approach used for describing the indoor abstraction. A translation has to be made between the object model, described in the previous chapter and a relational implementation, for storing the information in a relational database.

For the indoor abstraction some choices to store the model in the rDBMS are:

- Use a full relational approach, not using the geo-part of the DBMS, where all parts of the model described in previous chapter are stored via a relational way, even the geometry, this is not a good idea, because native spatial functions then can not be used from the DBMS but have to be programmed in a higher level language, like PL/SQL.

- Use a hybrid approach, using a Geo-DBMS, where the polygons are stored as geometry (as a separate type, specified in the DBMS) and the hierarchy of the model be implemented in a relational way.

The second approach will be choosen in the case study, to take the advantage of the re-use of functionality. The case study will make use of Oracle 10g with the Spatial Cartridge.

4.2.1 Storage of geometry in Oracle

The way geometry is implemented in Oracle is different from international standardisation initiatives. In Oracle a new objecttype, like string and integer, has been defined for geometry (see listing 4.1). Within this type all sorts of geometry can be stored, it thus acts like a container for storing geometry. This is, compared to standardisation, a different approach, because standardisation efforts define different objecttypes for each sort of geometry, such as a point type, a line type, etcetera. The standardised approach makes it less confusing what is stored in a geometry field.

Listing 4.1: Definition of the sdo_geometry type

```
1 CREATE TYPE sdo_geometry AS OBJECT (
2  SDO_GTYPE NUMBER, -- gives the type and dimension of the geometry
3  SDO_SRID NUMBER, -- defines coordinate system
4  SDO_POINT SDO_POINT_TYPE, -- defines object with x, y and z, e.g.
       for use as centroid
```

```
5  SDO_ELEM_INFO MDSYS.SDO_ELEM_INFO_ARRAY, -- how to interpret
       ordinates
6  SDO_ORDINATES MDSYS.SDO_ORDINATE_ARRAY -- summation of all the
       ordinates
7 );
```

In Oracle the object has the property GTYPE to distinguish the type of geometry that is stored and gives the dimension of the geometry. Values that are allowed are: $d000$ for unknown geometry, $d001$ for point, $d002$ for line, $d003$ for polygon, $d004$ for collection, $d005$ for multi-point, $d006$ for multi-line and $d007$ for multi-polygon, where d should be replaced by 2 for the second dimension (only x and y-coordinates in SDO_ORDINATES), 3 for the third dimension: x, y and z-coordinates stored in SDO_ORDINATES or even 4 for a fourth dimenstion. A point in three dimensions has a GTYPE of 3001.

Listing 4.2: A rectangle in the RD-system

```
1 SDO_GEOMETRY (
2  SDO_GTYPE = 2003,
3  SDO_SRID = 90112,
4  SDO_POINT = (13.5,19.5,NULL),
5  SDO_ELEM_INFO = (1,1003,3),
6  SDO_ORDINATES = (12,15,15,24)
7 );
```

The example in listing 4.2 shows a rectangle in the RD-system. It has a GTYPE of 2003. The 2 at the front shows that the polygon is in 2D and the 003 shows that it is a polygon. It has a centroid of $(13.5, 19.5)$. The SDO_ELEM_INFO shows how to interpret the SDO_ORDINATE-array. For the polygon the lower left and upper right coordinates are stored in het SDO_ORDINATE-array, which is shown by the 1003 in SDO_ELEM_INFO.

4.2.2 Mapping of objects to a relational DBMS

The object oriented approach differs from the relational approach. The relational approach, if the relationships between the tables are normalised, does not show any redundant data storage but uses links, while the object oriented approach uses the object paradigm where redundancy of properties is quite common. It is not uncommon to map the object oriented approach to a relational implementation, because of this storage requirement. Object-relational mappers are implemented in diverse programming languages (like Java and Python), where one can use objects in the programming language, while, if needed, objects can be made persistent by storing them in a relational DBMS.

If one has to map a UML model to a relational DBMS, there are several hints for this mapping process. In :reference to a quick summary guide to data modelling in UML: an excellent overview has been given for mapping objects to the relational implementation. In this research the approach described will be used. Below, only the two most important points of the mapping process will be tapped.

First, certain classes will be mapped into tables. These classes will be mapped to a relational table, while behaviour of these classes can be programmed in a programming language:

> ...the object model is based on discrete entities having both state (attributes/data) and behaviour, with access to the encapsulated data gen-

> erally through the class pulbic interface only. The relational model exposes all data equally, with limited support for associating behaviour with data elements through triggers, indexes and constraints.
>
> You navigate to distinct information in the object model by moving from object to object using unique object relationships (similar to a network data model). In the relational model you find rows by joining and filtering result sets using SQL using generalised search criteria.

Classes of objects do have a lot of different relationships which can be depicted quite easily in an object oriented model, but not so easily in a relational model:

> In the object model we have a rich set of relationships: inheritance, aggregation, association, composition dependency and others. In the relational model we can really only specify a relationship using foreign keys.

Quite often, the relationships in the object model depict m:n-relationships. This is the case for example with a polygon that can be used for more than one section, for example. These relationships, which are drawn in the object model with lines between the diverse classes, should be stored explicitly in a relational database, in a separate table used for storing this relationship.

Chapter 5

Case study

The building of the Aula has been taken as a case study, because the first International Symposium on Geo-information for Disaster Management will be held there. During the symposium some experiments will be done with mobile devices and evacuation. Hence, data for the interior and a model that describes the interior of the building is needed.

Figure 5.1: The Aula building

The building of the Aula has an impressive and difficult exterior (as shown in figure 5.1). Also the interior of the buiding has a certain degree of complexity. Because of this, only the front part of the building has been measured. The ground floor is a big hall with some counters in it. On the first floor there is a large space, in use as lobby. On both sides there are two hall ways, separated from the lobby by a wall. On the second floor there are some small meeting rooms and the 'Senators room'. The floors are connected with open stairs. On the second floor one can gain access to the large Auditorium room.

This chapter is structured as follows. In section 5.2 the data collection process and segmentation of the point cloud is described. In section 5.3 the indoor abstrac-

tion is described and in section 5.4 is shown how the data has been stored in Oracle (a Geo-DBMS).

5.1 Data collection and modelling

During two weeks some parts of the Aula has been scanned. This has been done with a Cyrax 2500 laser scanner, from Leica Geosystems. The laser scanner has an opening of 40 degrees in horizontal and verital direction. The resolution, which the scanner scans with, can be changed, as well as in horizontal or in vertical direction. The laser scanner was accompanied by a battery, a laptop and a tri-pod, which made it hard to move the scanner through the building. A simple cart under the tri-pod resolved this problem. Now, only two people were needed to relocate the laser scanner. It was the intention to achieve a resolution better than 2.5 cm in both directions. Because the distance between the laser scanner and the objects was about 10 meter, about 85,000 points were scanned in each scan. To achieve the same resolution in bigger rooms (where the distance between the laser scanner and object was larger) more time was needed to scan. Taking a scan this way took about 4 minutes.

The building is far from a standard three floor building. This has positive and negative effects on the measuring process. With the open stairs it was quite easy to connect the different floors together. However, due to the big amount of stairs and the symmetry of the building there were some ambiquity problems while connectig the scans manually. It is hard to see if the stair in the first scan also is the same stair in the second scan. The Aula has very diverse rooms. The meeting rooms have an area of 15 to 50 square meters, while the lobby has an area of about 100 square meters and the hall on the ground floor is near 200 square meters. Also the auditorium is an impressive room and a challenge to scan.

The result of the measuring process is 237 individual scans. These scans were made with overlap. By appointing same points within the overlap of the two scans, the scans could be connected together to one cloud. It is necessary that three ponts in the left and three points in the right scan are appointed manually. With the ICP algorithm a new point cloud can be created, in which the points of both scans are contained and in which the scans are positioned relative to each other. The method gives an accuracy measure of the connection in terms of standard deviation. The final result after the data acquisition and after the connecting of the scans a point cloud with more than 25 million points is the result: a huge dataset.

The laser scan point cloud with 25,134,871 points did reside in a text file of 1.1 Gigabytes. The file with point cloud points contains on each line either a tuple of numbers or only one number. If only one number is on the line, the number indicates that the following next lines belong to this scan. In listing 5.1 the number on the first line shows that the next 62,308 lines belong to the first scan. The second line shows the x, y and z-coordinates of the measured points. The fourth value is a reflection measure. Besides, a rectified photograph was draped onto the points, and the colour from this photograph was stored with the point as a red, green and blue value, that is what the fifth to the seventh values represent.

Listing 5.1: A part of the text file with point cloud data

```
1 62308
```

```
20.325 29.380 11.512 -383 25 6 25
21.393 29.841 10.543 -394 25 6 25
21.436 29.875 9.990 -357 25 6 25
21.371 29.840 9.912 -384 25 6 25
21.404 29.855 9.919 -386 25 6 25
21.436 29.868 9.912 -383 25 6 25
21.339 29.827 9.920 -355 25 6 25
21.396 29.853 9.985 -371 25 6 25
21.433 29.868 9.944 -369 25 6 25
...
```

The file with point cloud data also served as input for a segmentation program, developed at the section of Photogrammetry and Remote Sensing (faculty of Aerospace Engineering). The process of segmentation was based on a measure of homogeneity, being normal vectors pointing in the same direction. The segmented output consists of 4,564 polygons. For each point in the point cloud is stored which segmented polygon it belongs to.

5.2 Indoor abstraction

With the segmented polygons an indoor abstraction of the Aula can be formed. Each polygon has to be classified and with all the polygons a thematic hierarchy of the building can be build.

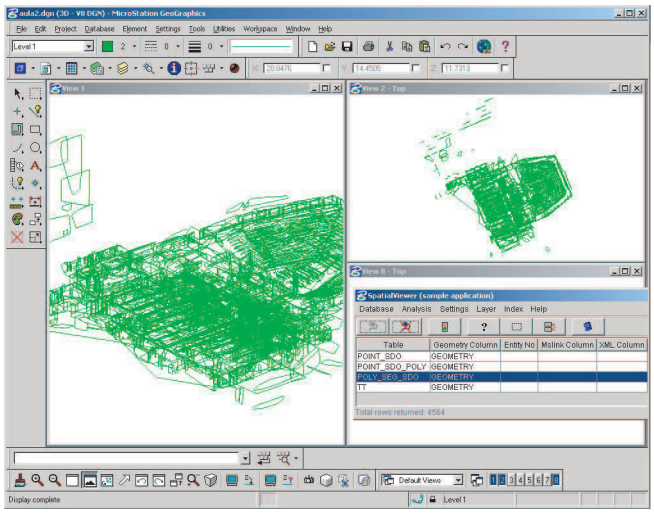
All segmented polygons are shown in figure 5.2(a). Because for each point out of the point cloud a relationship with a segmented polygon has been kept (if there is any relationship; Some points do not relate to any polygon), it is possible to select those points and segmented polygons that belong to a certain section, through the location of the points.

For each polygon a classification of the indoor abstraction has to take place, i.e., a polygon should end up in the correct database table with the right attributes. There are two possibilities to do this classification. In the graphical frontend (i.e. MicroStation) the segmented polygons are adapted to polygons. These adapted polygons are posted back to the database. Before the posting takes place, the type of the polygon is classified. After this, the polygon is posted to and stored in the DBMS. Another possibility is that only the geometry is posted to the database and the classification takes place within a client connecting to the database (e.g. with `sqlplus`). There are some known problems of the versions of software used in the case study, when using the first option. That is why the second option should be choosen.

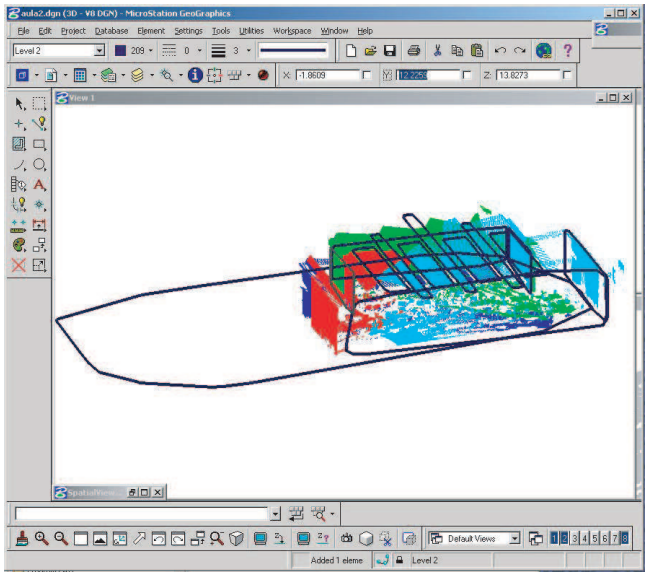
With the classified polygons the thematic hierarchy of the building can be build and stored in the DBMS, through the use of a DBMS client. The functions of the diverse sections are clear and good to distinguish. All sections in the Aula building have been given names, like 'Commissiekamer 1', 'Senaatszaal' and 'Auditorium'. The sections can be formed from the segmented polygons as is shown in figure 5.2(b).

5.3 Database management of indoor abstraction

Before data in a DBMS can be managed it should be loaded into the database. After loading a spatial index has to be created, before spatial functions can work with the



(a) All segmented polygons



(b) Section formed with segmented polygons

Figure 5.2: Segmented polygons

data. When this is all done, the data can be visualised with a graphical frontend. The object model developed in secion :TODO: has to be mapped to a relational implementation, before it can be used in a relational DBMS.

5.3.1 Data loading

To load the point cloud data in a DBMS bulk loading is to be used. A separate program (`sqlldr` in case of Oracle) that acts as a special client to the database reads data from a file. This data has been formatted according to certain rules and then can be inserted with a high speed into the database. A Python script

(shown in appendix A) was needed to translate the data in listing 5.1 to a suitable format for bulk loading. After transformation, a file was resulting, as is shown in listing 5.2. Each line contains a unique number per scan (scan id), a unique number within this scan for each point (point id), the type of geometry (i.e., a 3D point), the coordinates for each point (x, y, z), a reflection value, and the colour after draping an image over the point cloud (colour expressed in a red, green and blue-value).

Listing 5.2: Point cloud data suitable for bulk loading

```
'1' '1' '3001' '20.325' '29.380' '11.512' '-383' '25' '6' '25'
'1' '2' '3001' '21.393' '29.841' '10.543' '-394' '25' '6' '25'
'1' '3' '3001' '21.436' '29.875' '9.990' '-357' '25' '6' '25'
'1' '4' '3001' '21.371' '29.840' '9.912' '-384' '25' '6' '25'
'1' '5' '3001' '21.404' '29.855' '9.919' '-386' '25' '6' '25'
'1' '6' '3001' '21.436' '29.868' '9.912' '-383' '25' '6' '25'
'1' '7' '3001' '21.339' '29.827' '9.920' '-355' '25' '6' '25'
'1' '8' '3001' '21.396' '29.853' '9.985' '-371' '25' '6' '25'
'1' '9' '3001' '21.433' '29.868' '9.944' '-369' '25' '6' '25'
...
```

These data were loaded with the bulkloader program into the table `point_sdo`. This table is described in listing 5.3.

Listing 5.3: Description of the table that contains the laser scan points

```
SQL> DESC point_sdo;
 Name          Null?  Type
 ------------ ------ --------------------
 SCAN_ID              NUMBER(38)
 POINT_ID             NUMBER(38)
 GEOMETRY             MDSYS.SDO_GEOMETRY
 REFLECTION           NUMBER(38)
 RED                  NUMBER(38)
 GREEN                NUMBER(38)
 BLUE                 NUMBER(38)
```

For data loading with the bulkloader a description is needed how to interpret the data in the file that is going to be loaded into the database. This description is called a control file, because it gives control over how the bulkloader loads the data into the DBMS. The control file for the point data is shown in listing 5.4. It shows that each row in the file of listing 5.2 corresponds to a tuple to be inserted in the table shown in listing 5.3.

Listing 5.4: Bulk loading of point data (control file)

```
LOAD DATA
 INFILE "file"
 INTO TABLE point_sdo
 FIELDS TERMINATED BY ' ' OPTIONALLY ENCLOSED BY '\''
 (
  scan_id,
  point_id,
  geometry COLUMN OBJECT (
  SDO_GTYPE INTEGER EXTERNAL,
  SDO_POINT COLUMN OBJECT
  (
   X FLOAT EXTERNAL,
   Y FLOAT EXTERNAL,
   Z FLOAT EXTERNAL
```

```
15   )
16  ),
17  reflection, red, green, blue)
```

Because laser scanning delivers large point clouds, the amount of time needed to process the files is also substantial. The bulkloading process of the points took approximately $1\frac{1}{2}$ hour. After this, all the point data was loaded into the database. Now, spatial indexing techniques have to be used to make access to the data faster. Listing 5.5 shows the statement for creating a spatial index, a R-tree.

Listing 5.5: Creation of R-tree index

```
1 CREATE INDEX geometry_idx
2 ON point_sdo (geometry)
3 INDEXTYPE IS mdsys.spatial_index
4 PARAMETERS ('sdo_fanout=54 sdo_indx_dims=3 layer_gtype=point
    tablespace=indx');
```

The process of index creation took nearly 8 hours. The parameters are choosen this way, so that entries for the R-tree can be contained inline in a (physical) database block. Further the index is placed on another diskset, via the tablespace parameter. This makes physical access to the data faster.

The index has been created in 3 dimensions. Because of the 3 dimensional index, spatial operations of Oracle have been limited to the `sdo_filter` operations, where only the index, but not the exact geometry will be used for finding answers.

Before adaptation of the polygons could take place, the polygons were stored in the database. Output of the segmentation algorithm program was already in the format for the bulk loading client, so the bulkloading client could be used directly (no translation step was needed this time). The definition for the table that stores the segmented polygons is shown in listing 5.6.

Listing 5.6: Table definition for segmented polygons

```
1 CREATE TABLE poly_seg_sdo (
2     poly_id NUMBER,
3     geometry MDSYS.SDO_GEOMETRY
4 )
```

For bulkloading the information on segmented polygons into the database the control file was specified as in listing 5.7. In this case, the control file is integrated with the data file. Data starts on line 19. First, an unique identification number is given (1), then the interpretation of the geometry (3003) is given: it is a polygon in 3 dimensions. The triplet '1, 1003, 1' shows how to connect the ordinates that follow on the next line. On line 20 the first 3 ordinates are shown (`-3.01`, `13.904`, `-1.585`), on line 21 the polygon boundary is closed by the same values. On line 22 the next polygon definition starts.

Listing 5.7: Bulk loading segmented polygon data

```
1 LOAD DATA
2  INFILE *
3  TRUNCATE
4  CONTINUEIF NEXT(1:1) = '#'
5  INTO TABLE POLY_SEG_SDO
6  FIELDS TERMINATED BY '|'
7  TRAILING NULLCOLS (
8    poly_id  INTEGER EXTERNAL,
```

```
    geometry    COLUMN OBJECT
     (
       SDO_GTYPE         INTEGER EXTERNAL,
       SDO_ELEM_INFO     VARRAY TERMINATED BY '|/'
         (elements       FLOAT EXTERNAL),
       SDO_ORDINATES     VARRAY TERMINATED BY '|/'
         (ordinates      FLOAT EXTERNAL)
     )
)
 BEGINDATA
 1|3003|1|1003|1|/
#-3.01|13.904|-1.585|-3.45|13.801|-1.585|...
|-3.01|13.904|-1.585|-3.01|13.904|-1.585|/
 2|3003|1|1003|1|/
#-26.303|17.429|2.841|-35.853|12.567|2.835|...
|-26.303|17.429|2.841|-26.303|17.429|2.841|/
...
```

5.3.2 Visualisation

After loading the polygons, the polygons can be visualised in a frontend that can connect to the Geo-DBMS. Besides the whole dataset, also selections can be retrieved from the database. This has been done in figure 5.2(b). In figure 5.3 it is clearly shown that the segmentation process creates convex hulls. These convex hulls have to be manually adapted, so they accurately and completely describe the objects. An example of a manually adapted polygon is shown in figure 5.4.

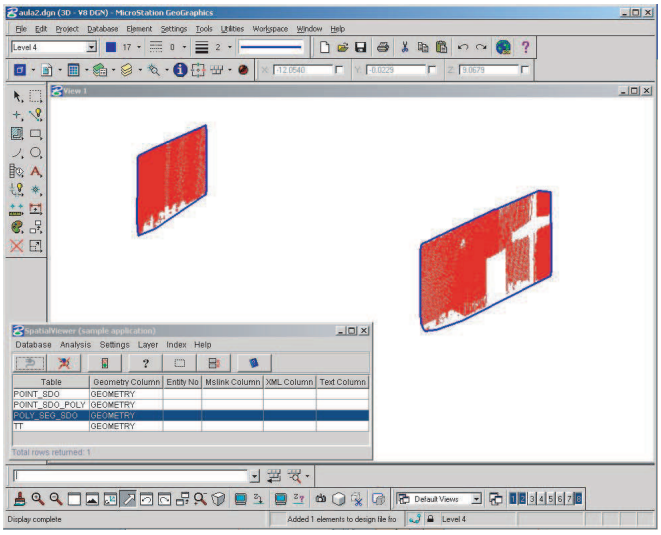


Figure 5.3: Convex hulls are created by segmentation process

5.3.3 Object model to relational implementation

To store the object oriented model described in chapter :TODO: a mapping process has to take place. As raised in section :TODO: a decision has to be taken which classes will be represented by tables, which will be represented by attributes and how relationships between classes will be mapped to the relational model. In the

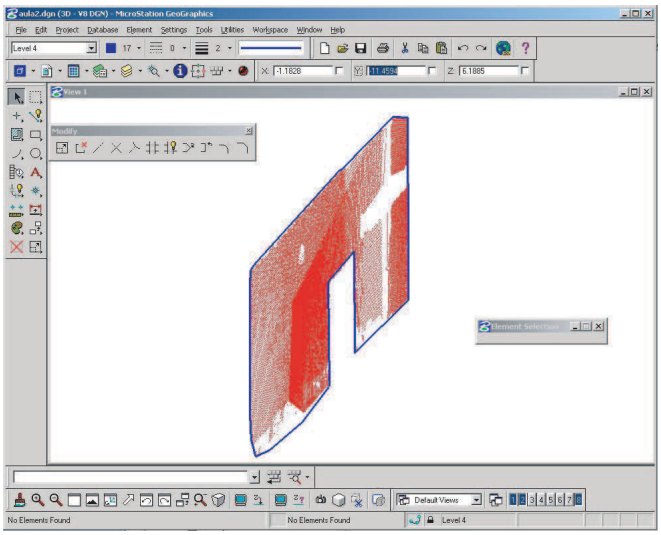


Figure 5.4: Adaptation of door

mapping that follows all relationship tables, used for m:n-relationships, start with 'join'. Classes that are mapped to relational tables are: 'Points'-class to 'point'-table, 'SurfacePatch' to 'surface_patch', 'Polygon' to 'polygon', 'Section' to 'section', 'Building' to 'building' and the 'Tree'-class to 'tree'-table.

Each table has a caption, which represents the tablename. In the left column names for each column in the table are specified. The right column shows the type of information stored in a column.

point (poi)	
poi_id	int
poi_scan_id	int
poi_geom	sdo_geometry
poi_r	int
poi_g	int
poi_b	int
poi_reflection	int

All laser scan points are stored in the 'point'-table. For each point information on from which scan the point resides, the geometry (x, y and z-coordinate), a colour (in r, g, and b-value) and a reflection measure is stored.

surface_patch (sur)	
sur_id	int
sur_geom	sdo_geometry

join_sur_poi (jspt)	
jspt_id	int
jspt_sur_id	int
jspt_poi_id	int

From the points groups of segmented points can be formed, from which surface patches can be reconstructed. Each surface patch has thus a relationship with the points. This relation is stored in the table 'join_sur_poi'. Each surface patch can be used as a base for an adapted polygon. This relation is stored via the table 'join_pol_sur'.

join_pol_sur (jps)	
jps_id	int
jps_pol_id	int
jps_sur_id	int

polygon (pol)	
pol_id	int
pol_per_ds	varchar
pol_exi_ds	varchar
pol_gra_ds	varchar
pol_there	boolean
pol_geom	sdo_geometry
pol_seg_id	int

Each adapted polygon is stored in the 'polygon'-table. For each polygon is stored: the persistence, existence and granting type. Also it is possible to store the there property of non-persistent polygons. The classes are stored in one table, because only the entries that are made in the table will be different (i.e., certain combinations of attributes are not allowed to be stored). Otherwise the modelling of the relationships with polygons and section would be much harder to implement and store.

section (sec)	
sec_id	int
sec_function_ds	varchar

join_sec_pol (jsp)	
jsp_id	int
jsp_sec_id	int
jsp_pol_id	int

A section is constructed from polygons that carry a classification. This link is stored through the 'join_sec_pol'-table. All sections are stored in the table 'section'. Each section has a function, this function can be stored in 'sec_function_ds'.

building (bui)	
bui_id	int
bui_nm	varchar

join_bui_tre (jbt)	
jbt_id	int
jbt_bui_id	int
jbt_tre_id	int

A building has to have one entry in the 'building' table. A building is a composite of sections and groups of sections, which can be represented by one or more tree structures. Each building thus has a relationship with the 'tree'-table, this relationship can be stored in the table 'join_bui_tre'.

tree (tre)	
tre_id	int
tre_type	enum('group', 'section', 'root')
tre_item_id	int
tre_parent_id	int

All information on the trees for representing the composition of sections and groups of sections within buildings can be stored in the 'tree'-table. Via 'tre_id' and 'tre_parent_id', via the use of a relationship, a tree structure can be stored based on relational identifiers. For each tree there should be a tuple with tre_type equal to root. This is the root node. Via 'tre_id' and 'tre_parent_id' branches of the tree can be stored. In 'tre_item_id' a relational link to the 'group'- or 'section'-table is made.

group (gro)	
gro_id	int
gro_nm	varchar

join_gro_sec (jgs)	
jgs_id	int
jgs_gro_id	int
jgs_sec_id	int

A node itself can be a group, or a section. The information for this is stored in the tables 'group' and 'section'. The containment that a group has is stored through the table 'join_gro_sec'.

As described in this chapter, the complete object oriented model for the interior of buildings is mapped to a relational Geo-DBMS.

Chapter 6

Conclusion and further research

The main objective was to build a formal, thematic model (an abstraction of the real world) for storage and querying of the interior of buildings. This objective has been reached.

It is important to create such a model, because nowadays, a description of buildings and information based on the form and layout of buildings is getting more and more important. Recent advances of laser scanning techniques allow already fast and economical creating of indoor datasets. The focus of this research has been on the organisation of the point clouds and the reconstructed objects.

The interior of buildings, reconstructed from point clouds, can be classified with polygons to form faces of polyhedrons. The interior of a building can be subdivided based on the function the sections have. Based on the polygons that form faces the thematic meaning of these sections can be classified. Interdependence of the sections can be based on geometric adjacency and topological adjacency through the faces. This creates a composition of a building, which can be represented with a tree. A translation from the object model, with which all the terms have been described, to relational model is needed to store the thematic hierarchy in a Geo-DBMS.

Further research on the interior of buildings can be conducted on several topics. For certain applications, like disaster management, the question on if it is possible to integrate movable objects, such as tables, chairs, plants, cupboards, etc., i.e. temporal assets of indoor building modelling, in the model described in this report, is an important one. Also an interesting question raised during this research is if automatic generation of a graph derived from this classification model can take place, and if so, how this can be done. Perhaps this is possible by transforming the sections to nodes and polygons of the granting class to edges. A database topology model, like Oracle's network model, can then be used for storing this derived graph.

Bibliography

[Aguilera, 1998] Aguilera, A. (1998). *Orthogonal polyhedra: study and application.* PhD thesis, Universitat Politécnica de Catalunya, Barcelona, Spain.

[Balis et al., 2004] Balis, V., Karamitsos, S., Kotsis, I., Liapakis, C., and Simpas, N. (2004). 3D Laser Scanning: Integration of Point Cloud and CCD Camera Video Data for the Production of High Resolution and Precision RGB Textured Models: Archaeological Monuments Surveying Application in Ancient Ilida . In *Proceedings of FIG Working week 2004, The Olympic Spirit in Surveying.*

[Besl and McKay, 1992] Besl, P. and McKay, N. (1992). A method for registration of 3d shapes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 14(2), pages 239–256.

[Biber et al., 2004] Biber, P., Andreasson, H., Duckett, T., and Schilling, A. (2004). 3D Modeling of Indoor Environments by a Mobile Robot with a Laser Scanner and Panoramic Camera. In *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS 2004).*

[Codd, 1970] Codd, E. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387.

[Èmolík and Uller, 2003] Èmolík, L. and Uller, M. (2003). Point Cloud Morphing. Presented on 7th Central European Seminar on Computer Graphics.

[Gilliéron and Merminod, 2003] Gilliéron, P. and Merminod, B. (2003). Personal Navigation System for Indoor Applications. In *Proceedings of the 11th IAIN World Congress.*

[Gool et al., 2004] Gool, L. V., Waelkens, M., Mueller, P., Vereenooghe, T., and Vergauwen, M. (2004). Total Recall: A Plea for Realism in Models of the Past. In *International Society for Photogrammetry and Remote Sensing (ISPRS), XXth ISPRS Congress, Commission V*, pages 332–343. GITC.

[ISO/TC 211/WG 2, ISO/CD 19107, 2003] ISO/TC 211/WG 2, ISO/CD 19107 (2003). Geographic Information – Spatial Schema.

[Kong and Rosenfeld, 1989] Kong, T. and Rosenfeld, A. (1989). Digital Topology: Introduction and Survey. *Computer Vision, Graphics and Image Processing*, 48:357–393.

[Maslow, 1970] Maslow, A. (1970). *Motivation & Personality.* New York: Harper and Row, 2nd edition.

[Meijers et al., 2005] Meijers, B., van Ree, J., and Vroom, W. (2005). Met 3D GIS is voor de drukte uit al een mensenmassa te navigeren. In Dutch, will be published in VI Matrix 93.

[Nüchter et al., 2003] Nüchter, A., Surmann, H., and Hertzberg, J. (2003). Planning Robot Motion for 3D Digitalization of Indoor Environments. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR '03)*.

[Open GIS Consortium, Inc., 1999] Open GIS Consortium, Inc. (1999). OpenGIS Simple Features Specification For SQL. Revision 1.1, Open GIS Project Document 99-049.

[Preparata and Shamos, 1985] Preparata, F. and Shamos, M. (1985). *Computation Geometry: an Introduction*. Springer-Verlag.

[Rabbani and van den Heuvel, 2004] Rabbani, T. and van den Heuvel, F. (2004). Methods for Fitting CSG Models to Point Clouds and their Comparison. In *Tenth Annual Conference of the Advanced School for Computing and Imaging*.

[Requicha, 1980] Requicha, A. (1980). Representations for Rigid Solids: Theory, Methods and Systems. *ACM Computing Surveys*, 12(4):437–464.

[Stoter, 2004] Stoter, J. (2004). *3D Cadaster*. PhD thesis, Technische Universiteit Delft.

[Stoter and Zlatanova, 2003] Stoter, J. and Zlatanova, S. (2003). 3D GIS, where are we standing? In *Proceedings of the ISPRS Joint Workshop on "Spatial, Temporal and Multi-Dimensional Data Modelling and Analysis"*.

[van Oosterom, 2001] van Oosterom, P. (2001). De geo-database als spin in het web. Delft. In Dutch.

[van Oosterom et al., 2003] van Oosterom, P., Quak, C., and Tijssen, T. (2003). Polygons: the unstable foundation of spatial modeling. In *Proceedings of the ISPRS Joint Workshop on "Spatial, Temporal and Multi-Dimensional Data Modelling and Analysis"*.

[van Oosterom et al., 2002] van Oosterom, P., Stoter, J., Quak, W., and Zlatanova, S. (2002). The Balance Between Geometry and Topology. In *Symposium on Geospatial Theory, Processing and Applications*. Ottawa.

[van Oosterom and Vijlbrief, 1996] van Oosterom, P. and Vijlbrief, T. (1996). The Spatial Location Code. In *The Seventh International Symposium on Spatial Data Handling*.

[Vosselman et al., 2004] Vosselman, G., Gorte, B., Sithole, G., and Rabbani, T. (2004). Recognising structure in laser scanner point clouds. In *International Conference on Laser-Scanners for Forest and Landscape Assessment and Instruments, Processing Methods and Applications, Freiburg*. Germany.

[Wikipedia, 2005] Wikipedia (2005). Plane (mathematics). `http://en.wikipedia.org/wiki/Plane`.

[Worboys, 1995] Worboys, M. (1995). *GIS: A Computing Perspective*. Taylor & Francis.

[Zlatanova, 2000] Zlatanova, S. (2000). *3D GIS for Urban development*. PhD thesis.

Appendix A

Transforming point cloud data suited for bulkloading

The python script that has been used to transform to bulkloading format for a point cloud.

Listing A.1: transform.py

```
# Copyright (c) 2004 Martijn Meijers
#
# Permission is hereby granted, free of charge, to any person
# obtaining a copy of this software and associated documentation
# files (the "Software"), to deal in the Software without
    restriction,
# including without limitation the rights to use, copy, modify,
# merge, publish, distribute, sublicense, and/or sell copies of the
# Software, and to permit persons to whom the Software is furnished
# to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
# included in all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
# EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
# OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
# NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
    HOLDERS
# BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN
    AN
# ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.

import sys

try:
    f = open('aula_ref_local_not_unified.pts')
    point = open('point20040929.txt', 'r+')
    file = 0
    i = 0 # scan_id
    for line in f:
        l = line.rstrip('\r\n').split(' ')

        # figure out what to do
```

```
34          if len(l) == 11:
35              #
36              # number of how many points the coming scan contains
37              #
38              i = i + 1
39              j = 1
40
41          elif len(l) == 7:
42              #
43              # rgb value is known
44              #
45              x = l[0]
46              y = l[1]
47              z = l[2]
48              reflection = l[3]
49              red = l[4]
50              green = l[5]
51              blue = l[6]
52              file = 1
53
54          elif len(l) == 4:
55              #
56              # rgb value is unknown -> set color to white
57              #
58              x = l[0]
59              y = l[1]
60              z = l[2]
61              reflection = l[3]
62              red = 255
63              green = 255
64              blue = 255
65              file = 1
66
67          # write eventually to file
68          if file == 1:
69              # write points to file
70              #
71              # E.g.
72              # -4.655947 -0.442396 -3.169199 -196 149 116 113
73              try:
74                  nl = "'" + repr(i) + "' '" + repr(j) + "' " + "
                        '3001' " + repr(x) + " " + repr(y) + " " + repr(
                        z) + " " + repr(reflection) + " "   + repr(red)
                        + " " + repr(green) + " " + repr(blue) + " " + "
                        \n"
75                  point.write( nl )
76                  j = j + 1
77              except IOError, e:
78                  print "Error %d: %s" % (e.args[0], e.args[1])
79                  sys.exit (1)
80
81  except IOError, e:
82      print "Error %d: %s" % (e.args[0], e.args[1])
83      sys.exit (1)
84
85  point.close()
86  f.close()
87  sys.exit (0)
```