Creation of an indoor route planner for airports An exploratory approach

Imke Ines Klatt
Matriculation number: 6028063
September 17, 2018

UTRECHT UNIVERSITY
GEOGRAPHICAL SCIENCES

Supervisors:

CORNE VREUGDENHIL MSc

Dr.-ir. RON VAN LAMMEREN

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place and Date

Signature

Abstract

The main target of the current work is to create an indoor route planner specifically designed for the indoor environment of airports which calculates for the departing passenger a path with intermediate destinations, from entrance over all needed boarding destinations, to the user-relevant gate. Further, the application should provide the option to include additional passenger-preferred tasks into the path, as e.g. to have a café or to go to the toilet. This resulted, besides a suggestion for the system architecture, in a theoretical setup of a hierarchical, directed, accessibility routing graph with partially airport-specific semantics and two path finding algorithms which consider on the one hand boarding destinations in the correct sequence as well as user preferred additional intermediate destinations. Furthermore, vertical passages are considered in the network construction approach as well as in the path finding algorithms.

Contents

1 Introduction **1**

2 Aim and Research Questions **3**
2.1 Constraints . . . 4

3 Literature review **4**
3.1 General concepts for indoor space routing . . . 4
3.2 Current works on indoor modelling . . . 9
3.3 Pathfinding Algorithms . . . 12
3.3.1 A* algorithm . . . 12

4 Method **14**
4.1 Data and Software . . . 14
4.1.1 DXF Drawing Interchange Format . . . 15
4.2 User requirements and corresponding POIs . . . 16
4.2.1 Boarding . . . 16
4.2.2 Positioning: rID0 . . . 18
4.2.3 Integration of additional activities: aIDs . . . 18
4.2.4 Time limitation . . . 19
4.2.5 Accessibility of vertical passages . . . 19
4.3 Routing graph construction . . . 20
4.3.1 Data acquisition and preprocessing . . . 21
4.3.2 Identification of Accessible Corridors, POIs and VPs . . . 21
4.3.3 Space decomposition . . . 21
4.3.4 Generation and classification of Nodes . . . 22
4.3.5 Pathway Generation . . . 27
4.3.6 Edge classification . . . 27
4.3.7 Introduction of a Z-coordinate . . . 28
4.3.8 Topology checking and data validation . . . 29
4.3.9 Generation of network for each floor . . . 29
4.3.10 Generation of one 2.5D Network Dataset . . . 29
4.4 Routing Algorithm . . . 29
4.4.1 Boarding path . . . 29
4.4.2 Activity path: Integration of additional destinations . . . 33
4.5 System requirements and architecture . . . 37
4.6 Testing . . . 39
4.6.1 General Performance . . . 39
4.6.2 Usability and Usefulness . . . 39

5 Results **40**
5.1 Challenges with data preprocessing . . . 41

6 Evaluation **46**

Appendices **51**

A ArcPy helper functions **51**

B Questionnaire Spatial Anxiety **53**

List of Figures

1 Subprocesses of human wayfinding (Downs & Stea 1973) . . . 1
2 Examples of most used current wayfinding support techniques of airports. Front Image: 3D map of Airport Stuttgart (Germany) as presented on airport's website. Back Image: use of map and signage within Düsseldorf Airport, Germany (Solinger Tageblatt, 2016). 2
3 Different Level of Detail(LOD) by (Hagedorn, Trapp, Glander, & Döllner, 2009) . . . 5
4 Taxonomy of indoor space models according to (Worboys, 2011). . . . 6
5 Different types of graphs relevant for routing. Image source: own work . . . 7
6 Extracting the centerline of the interior of a building by applying Voronoi Diagrams. . 8
7 Biased path of centerline within open spaces . . . 8
8 Cells and access points according to (Yuan & Schneider, 2010). . . . 9
9 Access point setting for open cells and path generation according to the approach of (Yuan & Schneider, 2010) . . . 10
10 Mapping indoor structures according to (Lorenz, Ohlbach, & Stoffel, 2006)(A) and (Yuan & Schneider, 2010)(B). . . . 10
11 Mapping indoor structures according to the simple tagging model of OSM. Image source : wiki.openstreetmap.org . . . 11
12 Comparing most common algorithms for shortest path calculations: A* and Dijkstra . . 12
13 Challenge to implement A* in a 2.5D network . . . 14
14 DXF file information storage . . . 15
15 Diagram of the boarding process . . . 17
16 Significant architectural structures within an airport for QR code location setting. Here, the tree structures within the Airport Stuttgart. Image source: Sascha Foerster/airlines.net . . . 19
17 Overview on the requirements on the to-construct routing graph. . . . 20
18 Effect of applying a hierarchy on a graph and on the relevant shortest path calculation. The blue edges indicate the path from node 1 to target node 2. . . . 21
19 Some structures which are not accessible (n.a.) for passengers are excluded from further processing and i.e. will not be integrated in the network . . . 21
20 Inclusion diagram of space decomposition . . . 23
21 Classification of the network data model . . . 24
22 Generation of nodes for clustered structures interchangeable for the passenger . . . 25
23 Node setting for waiting areas at gates . . . 25
24 Depiction of the same vertical building structure, on the left interpretation of the concept of (L. Liu & Zlatanova, 2012), on the right the altered concept used within the current network . . . 26
25 Comparing depiction of a direct-path(blue lines), the top image shows an ordinary direct path while the bottom shows the graph after the inclusion of a hierarchy. . . . 28
26 Different types of vertical connectors and responding edge directions. Abbreviations: S = Stairs, L = Lift, E = Escalator . . . 29
27 Calculation of a short path between two nodes by considering vertical passages . . . 32
28 Calculation of a route considering in both areas (Public and Security) two aIDs (aID_1, aID_2) with each multilocal options(aID_1j, aID_2j). . . . 33
29 Procedure of the algorithm on selecting appropriate aIDs . . . 36
30 Information flow of the suggested route planner . . . 37
31 Suggested system architecture of the route planner . . . 38
32 Algorithm 1 and 2 helper function VCpath: Short path calculation based on A* by considering also vertical passages . . . 40

33 Algorithm 2 helper function findaIDs : Selecting close aIDs by implicitly navigating the user to the next boarding location D 41
34 Suggested steps for data preparation 42
35 Top: Original data for one floorlevel of the airport. Bottom: After cleaning data of the Airport 43
36 Examples of so-called structural gaps 44
37 Examples of visible topological errors 45
38 Generated access points for aID type 'sale'(red). Blue points are corresponding annotation point data. 45
39 Annotation for directions in CAD representation of stairs. Image source: blogs.rand.com 46
40 Questionnaire by Lawton and Kallai (2002). 53

List of Tables

1 Significant topological differences between indoor and outdoor environments 4
2 Graphical presentation included within the different LODs 5
3 Description of the hierarchical levels and corresponding access points 22
4 Description of Network profiles 30

1 Introduction

Modern public buildings are often large, multistorey and their internal structures complex in the way that even persons familiar with the environment are likely to get lost when searching for a particular room, place or service. For people unfamiliar with the inner building structure there is an even stronger need for proper guidance (Raubal & Egenhofer, 1998) (Hölscher, Vrachliotis, & Meilinger, 2006). These circumstances imply the need to provide for effective wayfinding systems in public large-scale buildings. A wayfinding system aims to provide for proper guidance by supporting the process of human wayfinding.

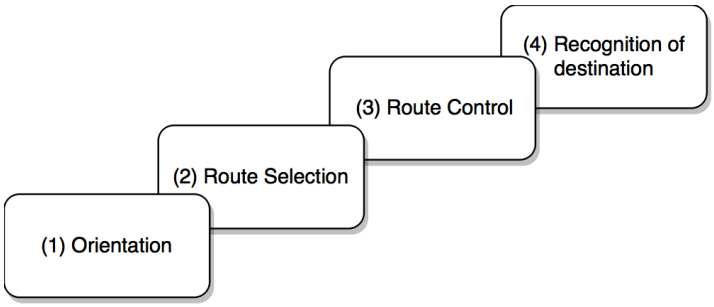


Figure 1: Subprocesses of human wayfinding (Downs & Stea 1973)

The process of human wayfinding can be broken down into four subprocesses: (1) orientation, (2) route selection, (3) route control and (4) recognition of destination (Downs & Stea, 1973) (see ***Figure 1***). (1) Orientation is the process of which a person localizes oneself within the environment, (2) route selection describes the process in which the person chooses a route which will lead to the desired destination, (3) route control implies the constant control and confirmation that the person is following the selected route and (4) is the ability of the person to realize that he/she has reached the desired destination. These processes are supported by a wayfinding system in the way that it provides a person with relevant visual cues within the environment as e.g. maps, paths, directions, symbols but also applications, which should help the person to find a way to the desired destination. For example a good viewable sign stating the actual storey within a building gives a person information on his/her actual location and, as such, supports the orientation within the built environment.

An exemplar highly complex built environment where the benefits of effective wayfinding is especially tangible are airports. The main benefit refers to the improvement of traveller flow which as such reduces airport crowding. As a result it supports passengers getting to their flights easily on time by also decreasing enquiries to airport staff, traveller frustration and confusion, which, ultimately, leads to increased traveller satisfaction (Churchill, Dada, de Barros, & Wirasinghe, 2008; Correia, Wirasinghe, & de Barros, 2008; de Barros, Somasundaraswaran, & Wirasinghe, 2007; Farr, Kleinschmidt, Yarlagadda, & Mengersen, 2012). In sum, an effective wayfinding support service at airports supports travellers reach their destination(s) within the built environment more quickly and easily. Besides, these circumstances increase the traveller's time available for the exploration of the conveniences of the airport (e.g. shops or gastronomy).

One response to facilitate wayfinding in airports is the approach of indoor navigation which aims to support people to orientate and navigate from A to B as quickly as possible. Indoor navigation comprises the phases of localization, routing and tracking (Gilliéron, Daniela Büchel, & Ivan Spassov, 2004). With regard to the described human wayfinding processes, indoor navigation can be considered to be an adequate response to support wayfinding as it covers all its subprocesses. However, while outdoor navigation became a more or less standard commodity, indoor navigation is less well investigated and has its practical difficulties. In contrast to indoor navigation, most of the current outdoor

Figure 2: Examples of most used current wayfinding support techniques of airports. Front Image: 3D map of Airport Stuttgart (Germany) as presented on airport's website. Back Image: use of map and signage within Düsseldorf Airport, Germany (Solinger Tageblatt, 2016).

wayfinding applications base on signals of the global navigation satellite system (GNSS). The Global Positioning System (GPS) is the leading location technology which is commonly used to provide for real-time directions or traffic information as used in e.g. the TomTom navigation system. However, GPS is failure-prone in enclosed environments as it requires 'line of sight' to relevant satellites (Fouskas, Giaglis, Kourouthanassis, Pateli, & Tsamakos, 2002). Further, even though GPS offers height information its accuracy is not sufficient for indoor navigation. To overcome the limitations of GPS, in recent years, technologies became available which are specialized for indoor navigation by providing positional techniques other than GPS. Most common positional techniques for indoor navigation are the use of Wi-fi or BLE (Bluetooth Low Energy). Both techniques apply methods as fingerprinting or trilateration to localize a relevant device (e.g. a mobile phone). Wi-Fi positioning systems (WPS) use the strength of Wi-Fi signals and the MAC address for positioning and tracking. Instead, BLE systems (e.g. iBeacon) use Bluetooth Low Energy signals.However, while Wi-Fi systems are relatively inaccurate (5 – 15 m)(Gaudlitz, 2015) and therefore still not optimal for indoor navigation (Smolders & Görtz, 2016), BLE systems are indeed relatively accurate with an accuracy of 5 - 7 m (Smolders & Görtz, 2016) but costly and labor intensive (Dudas, Ghafourian, & Karimi, 2009). As a consequence, not many airports have implemented real-time indoor navigation systems yet nor plan to do so. Instead, most airport wayfinding support services rely on online maps presenting 2D or 3D views on the airport's terminals and a combination of map and signage within the airport building (see ***Figure 2***).

Even though this system supports human wayfinding, it is incomplete as it does not cover the 2nd subprocess of human wayfinding (route selection). As a consequence, the cognitive workload for the passenger is high as he/she needs to find the path and memorize key information (environmental cues) to the destination themselves. This increases the possibility to make erroneous decisions which in this context means a passenger may take a wrong turn with the result to get lost within the building and receive stress. Ultimately, a deficient wayfinding system is considered to lower the benefits for both, the passenger as he/she receives stress as well as for the airport operators, who are interested in the revenues of relaxed passengers exploring the offered conveniences.
However, instead with navigation, instead, wayfinding can also be supported by indoor routing. Indoor routing aims to calculate, select and present routes in order to support localization and finding relevant paths as well as to instruct and guide users to desired destinations within a complex (3D) environment (Hagedorn et al., 2009). In other words, indoor routing provides support to all subprocesses of wayfinding with the main emphasis on route selection. Further, in comparison to indoor navigation, indoor routing is less costly as it does not need costly infrastructures while still providing for navigation guidance. As such it may be a valuable addition to maps and signage and is considered to be a cost-effective alternative to indoor navigation (Dudas et al., 2009). However, indoor routing is less investigated as outdoor routing and therefore is seen as more challenging. One of its challenges refers to the creation of an indoor network model. By comparing indoor and outdoor space, it is noticeable that some objects as e.g. rooms or distinct areas in huge halls do not have counterparts in the outdoor street network which results in a more challenging path network construction process (Goetz & Zipf, 2011; Yuan & Schneider, 2010). Several efforts have been made in creating indoor networks (see e.g. (Goetz & Zipf, 2011; L. Liu & Zlatanova, 2011) but existing approaches to model indoor environments are not specifically designed for the unique environment of airports. This deficit refers not only to spatial indoor models but also to indoor semantics (see for existing semantic models e.g. (Dudas et al., 2009; L. Liu & Zlatanova, 2011) . As a consequence, currently there exist no spatial nor semantic model specifically for the indoor environment of airports which makes the network construction for an airport interesting for exploration.
Further, unlike for outdoor routing, shortest path is usually not the best path indoors, instead a personalized path considering user preference, requirements and interest are considered to be more adequate (Bian, Guo, & Qiu, 2014; Dudas et al., 2009; Yuan & Schneider, 2010) . For example, vertical passages (elevators, escalator, stairways) are a specific feature of multilevel buildings and the user's requirements on how to overcome them needs to be considered in path calculations. For example a person using a wheelchair would not be able to take a path which includes stairways.
A third challenge refers to the path generation for a network with more than two dimensions. To clarify, while roads are commonly defined as 2D structures, buildings need to be regarded as 2.5D (Hölscher et al., 2006) or even 3D due to their multiple storeys. As a consequence, the calculated path bases on a topological model or graph which has an extra degree of freedom in the node labels.
To conclude, indoor routing itself but especially in combination with the complex environment of airports are faced by manifold challenges. With regard to the discussed benefits of indoor routing as a cost-efficient complement to current common wayfinding systems at airports, this topic is considered to be interesting and useful for further exploration.

2 Aim and Research Questions

The main aim of the thesis is to build a prototype web application of a route planner for the built environment of airports. Furthermore, as literature suggests that most visitors at airports aim to depart (Churchill et al., 2008), the route planner will be designed specifically for departing travellers. The main research question is:

How to build a route planner for the indoor environment of airports considering the needs of departing travellers?

The research question can be broken down into the following subquestions:

1. *How to build a routable graph-based network of indoor space with regard to spatial, quantitative as well as qualitative information?*
2. *Which algorithm(s) is(are) adequate for indoor routing?*
3. *What are the systems requirements to build the desired route planner?*
4. *How to build the desired route planner considering previous theoretical research and the needs of travellers departing from an airport?*
5. *How is the general performance of the route planner and how usable and useful for potential travellers?*

2.1 Constraints

Central to the suggested route planner is to support departing passengers in pathfinding within the airport built environment. Accordingly, the suggested routing service will not include areas relevant for other tasks as e.g. arrival or transfer. Further, even though the development of a positional technique is not in the focus, the thesis gives a short indication on how to localize the start position by using QR-codes.

3 Literature review

3.1 General concepts for indoor space routing

In the geographical context, the mapping of an outdoor infrastructure to a graph structure is generally a straightforward task as streets can be modelled as edges and their intersections as nodes. However, although some structures of indoor space show features analogue to outdoor space(e.g. corridors may be comparable to streets) there are indoor structures which cannot be modelled according to outdoor concepts, e.g. rooms or areas (Yuan & Schneider, 2010).(Stoffel, 2009) describes four aspects which considerably distinguish indoor environments from outdoor and which are important to consider in the mapping of indoor space to a graph: shape, movement, granularity and dimension(Table 1).

Aspect	**Outdoor**	**Indoor**
Shape	uniform	diverse
Movement	confined	free
Granularity	coarse	fine
Dimension	2D	3D, 2.5D

Table 1: Significant topological differences between indoor and outdoor environments

Shape Outdoors, pathways as roads and tracks generally have standard properties regarding width, the number of lanes etc. Accordingly, the network structure representing road structures is clearly defined and regular, for example each junction marks a decision point and road segments are generally

linear. In contrast, indoor spaces, often show a high variability in size and shape and, furthermore, in their amount of connections between spaces for which schematic processing often does not suffice.

Movement While infrastructures outdoors imply a network-like structure applying pre-defined movement rules by limiting among others directions and travel speeds, indoor spaces generally do not apply such rules. This difference becomes especially clear within wide indoor spaces as e.g. terminal halls. Within a terminal hall, there is no certain path or speed-limit pre-given for pedestrians, while the use of a motorway presets the drive direction, turn restriction and speed limit of a driver. As such, while the creation of a network for outdoors may take over explicit rules on movement, in indoor space, movement is not specified thereby revealing no explicit pathway structures.

Granularity Pedestrians are moving slower than cars, which influences their perception and perspective of the world around them. Accordingly, they receive the environment more detailed than e.g. a driver of a car. Accordingly, a routing model for a pedestrian needs to be richer in detail in order to adequately represent the environment (s)he receives. Furthermore, (Luebecke, 2003) suggests that the complexity of a model is determined by the level of abstraction adequate for a specific task or requirement. As a consequence, the user task and the relevant requirements should determine the complexity of the representation of the building structure. (Hagedorn et al., 2009) proposes four different levels of detail (LOD) for indoor routing which refer to the thematic, geometric, topological and visual complexity. (Refer to and). ***Table 2*** gives a tabularic overview and ***Figure 3*** the visual analog on the different LODs, which varies from a very coarse representation comprised of a 2D view only of the outer shell of a building at LOD 1 to a detailed 3D view on the internal structure and objects of a building at LOD 4.

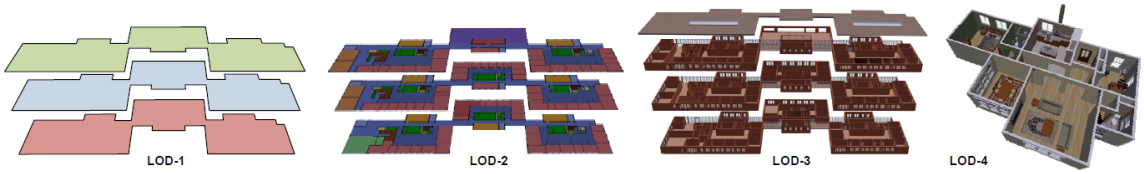


Figure 3: Different Level of Detail(LOD) by (Hagedorn et al., 2009)

LOD	**Graphical presentation**
1	Building's outer shell, it's access points and building parts
2	2D floor plans,various spaces within floors, indoor routes
3	inclusion of space height (3D), general higher realism, shape of doors and windows
4	detailed interior and exterior, highest degree of realism

Table 2: Graphical presentation included within the different LODs

With the differing LODs it is intended to provide for design solutions in order to adapt a geospatial model to its use purpose while also reducing its inherent complexity for a better processing performance. As a conclusion, the approach suggests to fit the complexity of the used model to the task which implies the adaption, simplification and selection of the built structures.

Dimension While navigational models for street networks are usually based on a 2D plane, this is often not the case indoors. Due to the fact that buildings often have multiple floors a 2D representation does often not suffice the requirements for an indoor network.

Another difference between indoor and outdoor routing is pointed out by (Fellner, Huang, & Gartner, 2017) who suggest that route instructions for outdoor routing is usually metric, however, indoors it is suggested that instructions relying on significant structures('landmarks') may be more useful for pedestrians than metric instructions used within navigation systems for cars.

Indoor space models (Worboys, 2011) developed a top level taxonomy of indoor space models (***Figure 4***).

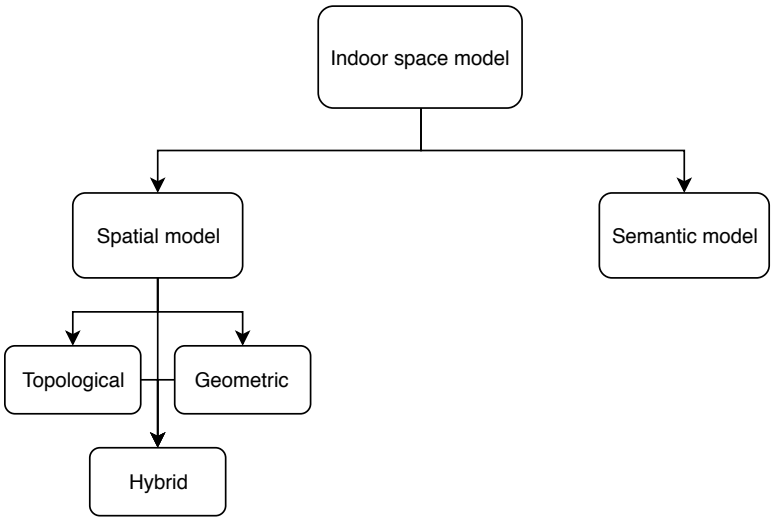
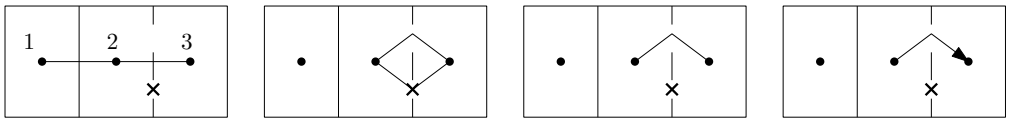


Figure 4: Taxonomy of indoor space models according to (Worboys, 2011).

According to this, indoor space models differentiate into semantic models and spatial models. While semantic models represent the variety of spatial entities, their properties and relationships in indoor space, spatial models refer to models considering whether topology, geometry or a combination of both. The latest one are defined as hybrid models. Topological indoor space models are concerned with the connectivity properties of indoor space which further divides into primal space (also 'structural space') and the path space. Topology of the structural space refers to the connection properties of structural components in building as e.g. rooms, walls, doors etc. The path space refers to the space of (implicit) pathways within a building. The path space is usually presented as the routing graph G = (V, A), where V is a set of nodes and A is a set of edges.

Graphtypes (Worboys, 2011) differentiates between three types of graphs relevant for routing: adjacency, connectivity and accessibility.(***Figure 5***)

In an adjacency graph (***Figure 5***a) nodes represent spatial entities, and edges between those nodes represent a shared physical boundary between them. Two rooms sharing a wall would be an example of two adjacent rooms and would be regarded in an adjacency graph as 'connected'.
Instead of adjacency, a connectivity graph(***Figure 5***b)takes a physical opening between two entities of



(a) adjacency

o \ d	1	2	3
1	0	1	0
2	1	0	1
3	0	1	0

(b) connectivity

	1	2	3
1	0	0	0
2	0	0	2
3	0	2	0

(c) accessibility

	1	2	3
1	0	0	0
2	0	0	1
3	0	1	0

(d) directed accessibility

	1	2	3
1	0	0	0
2	0	0	1
3	0	0	0

o = origin

d = destination

Figure 5: Different types of graphs relevant for routing. Image source: own work

space as e.g. a door as the precondition for a connection, while the accessibility graph(***Figure 5***c)further includes information on the accessibility of the physical opening and accordingly stores a connection only if the opening is also accessible. For example, considering the creation of a route planner for passengers of an airport, within the airport building may be a room which may have a physical opening as a door to the terminal but the access to the room is limited to the airport staff. In such a case, the room is not accessible for the passengers and as such would not be included in the routing graph.
A graph may further be specified with regard to its direction (***Figure 5***d). Indoors this is rather unusual as in most case people can walk freely within indoor spaces. However, in some environments direction may become important. One example for the requirement of a one-way within indoors are escalators, which only go into one vertical direction, up or down. However, specific for airports would be crossing from the public to the security zone. Here, once crossed the security control people are not allowed to return to the public area even though a physical opening would be available.

Hierarchical graphs To the graph types explained above a hierarchy can be added in order to create containment or partitioning. In a general definition, hierarchical graphs are a series of different graphs at different levels of detail or granularity (Stoffel, 2009). (Stoffel, 2009) describes two approaches, top-down and bottom up. While the top-down approach refers to the partitioning of one large graph into several pieces according to some specified criteria(s), the bottom-up approach describes the integration from a number of smaller graphs into one single one. In other words, the first type of spatial network graph requires integration to receive a hierarchical structure of space and the other approach describes a migration.

Pathway approaches Early indoor navigation models were exclusively symbolic without considering the geometry of indoor spaces. A path description basing entirely on the topology would be a list of entities needed to be taken in order to reach destination without considering distances, angles or obstacles. However, for routing a network with the goal to find a short path and give out a distance it exist two approaches to support metrical route planning in indoor spaces: the grid-based approach and the path-based approach. While the first approach decomposes spaces into cells and explores the connectivity of cells, the latter constructs implicit paths in an indoor space that are directly based on the architectural structure of the relevant building. The latter is generally used for routing.
However, in essence there are two approaches generally used to construct the pathway of a hybrid

network, on the one hand the centerline approach, and on the other the Visibility graph approach.

Medial Axis (Centerline) Per definition the Medial axis is a set of points which are equidistant to the two closest boundaries which, when connected, result in straight or parabolic edges, depending on the properties of the relevant boundaries. The medial axis originates from the creation of Voronoi diagrams (***Figure 6***) however, without considering vertices of concave corners in the network.

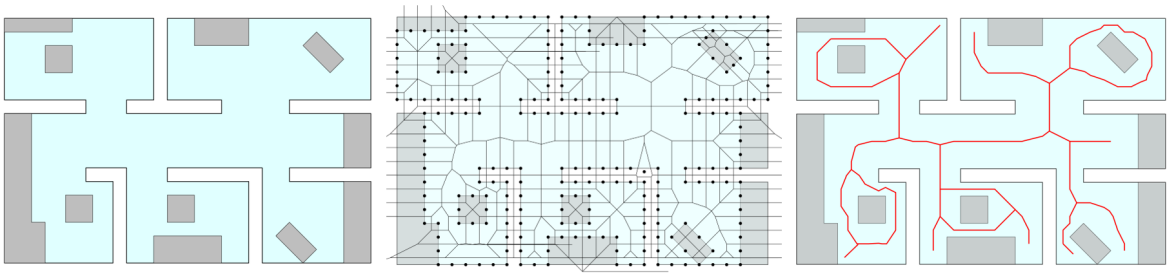


Figure 6: Extracting the centerline of the interior of a building by applying Voronoi Diagrams.

For the extraction of the centerline from the Medial axis it is convenient to cut off all branches that lead to convex corners. Accordingly, the generalizing steps from Voronoi diagrams to the centerline need considerable time. The centerline approach is well-suited for routes within closed spaces as e.g corridors in which the shape of the path is implicitly given. Instead, open spaces are a weakness of the centerline approach when it comes to useful path description, this as paths are biased towards the centre of the open space. For example, in ***Figure 7*** one would expect a path which guides directly from A to B as the locations are adjacent. However, due to the bias of the Medial Axis to the middle of space, the suggested path would guide a user to first go off from that area into the middle of the space and then return to the same area in order to reach B.

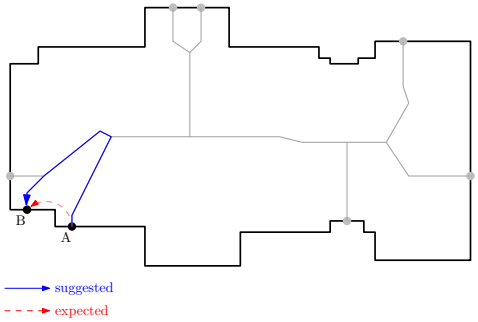


Figure 7: Biased path of centerline within open spaces

Visibility graph Another approach for path generation is the construction of a visibility graph. Originally created for robot path planning, the goal of a visibility graph is the creation of path segments between each pair of nodes which are considered to be visible from each other. In the visibility graph, nodes represent convex corners of regions including corners of all obstacles and additionally all openings to the region. The resulting geometric paths are optimal. However, even though the resulting path description is not suitable for humans (for example “the next entrance is within a distance of 7.4 m at

an angle of 128 degree")it finds several adaptions for human path planning in scientific works, which will be considered in more detail in section 'Spatial models'. A general major disadvantage of this approach is the intensive storage space needed for edges and nodes. For example, a multilevel building with many rooms of which each pair of nodes will be needed to be connected by an edge would need a significant amount of storage space which may not be acceptable regarding performance.

3.2 Current works on indoor modelling

Spatial models To support shortest path routing (Yuan & Schneider, 2010) propose a hybrid spatial model by constructing a spatial network on the basis of a visibility graph. For this, they divide the architectural indoor space into different basic units, which include rooms, corridors and lobbies. Those units are named *cells* which can be accessed by *access points*. Cells are further subdivided into *simple cells*, *complex cells*, *open cells* and *connectors*(***Figure 8***).

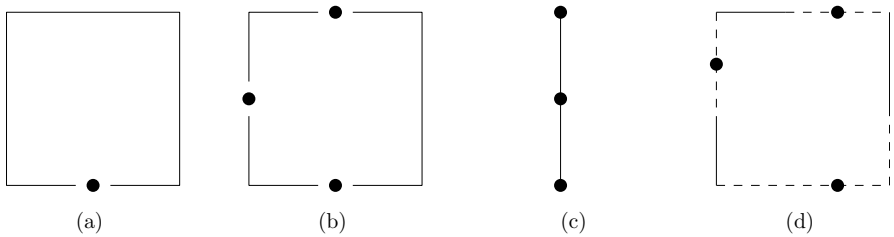


Figure 8: Cells and access points according to (Yuan & Schneider, 2010).

According to their approach a simple cell (***Figure 8***a) is a cell with only one access point, e.g. a room which is accessible by one door. Accordingly, the room cannot function as a passage but instead functions as start or target object. A complex cell (***Figure 8***b) can be assessed by multiple access points and as such can function as start and target object as well as passage. A connector cell (***Figure 8***c) is an object that connects different floors in a building by using a vertical unit as stairways, elevators and escalator. An open cell (***Figure 8***d) is a cell which provides for the most parts no solid boundaries as e.g. walls, which makes it difficult to determine access points. Examples of open cells in airports are areas as e.g.a waiting area in front of a check-in counter. To connect the interior of cells (Yuan & Schneider, 2010) consider implicit path segments often taken by people within cells. This is, according to them, the optimal path between a pair of access points and as such follows a visibility graph approach. According to this approach, in a complex cell, the implicit path between two access points would be a straight line between them. In an open cell this is less intuitive as the openings are not pregiven by physical structures. Alternatively, they select the center position of each open boundary as its access point, and construct path segments between the created access points.However, as geometry needs to be considered this approach needs an adjustment in order to handle concave shaped structures adequately. ***Figure 9*** shows an example of the creation of the so-called Direct-path, which is a variation of the visibility graph. ***Figure 9***a shows a complex cells which is further decomposed into into five open cells (A-E). In ***Figure 9***b all path segments between the openings are created. The red vertices are implicit access points. The blue line indicates a path segment intersecting with the boundary due to the concave shape. (Yuan & Schneider, 2010) select the vertex on the concave boundary as an intermediate point and accordingly segments the straight line into two segments. This process of segmenting continues until no generated segment between two openings intersects a boundary.

However, in wide halls with a complex internal structure as e.g. within a terminal hall of an airport, the path graph resulting from connecting each pair of access points as according to the approach of (Yuan & Schneider, 2010) is considered to be unacceptable regarding performance and storage (Goetz

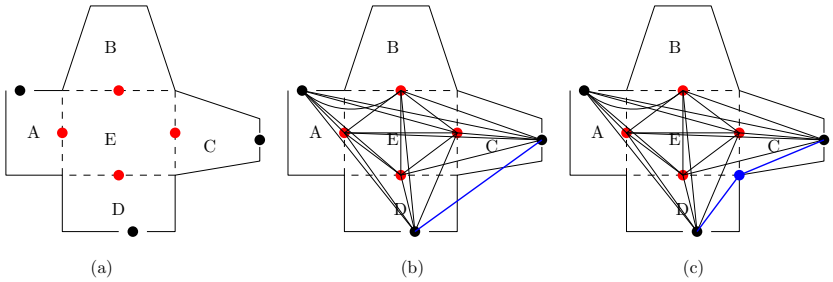


Figure 9: Access point setting for open cells and path generation according to the approach of (Yuan & Schneider, 2010)

& Zipf, 2011) due to its complexity as described in 3.1. Lorenz et al. (2006) proposes a hybrid spatial model in which small building structures as e.g. rooms or lobbies are directly mapped with their centerpoints while long spaces are partitioned into sectioned. Accordingly, long corridors are split into shorter sections and each section is represented in the graph with its centerpoint(10a). Consequently, one single floor is mapped with the amount of nodes equal to its sections which are connected by edges. Furthermore, they map structures by their centerpoints which is in contrast to the approach of (Yuan & Schneider, 2010) who base their approach on the assumption that openings as e.g. doors to certain structures are the destination of a user, rather than the centerpoints. Figure 10 shows the two different approaches of (Lorenz et al., 2006) and (Yuan & Schneider, 2010) to map the building structures of a corridor to a connectivity graph with the purpose of routing.

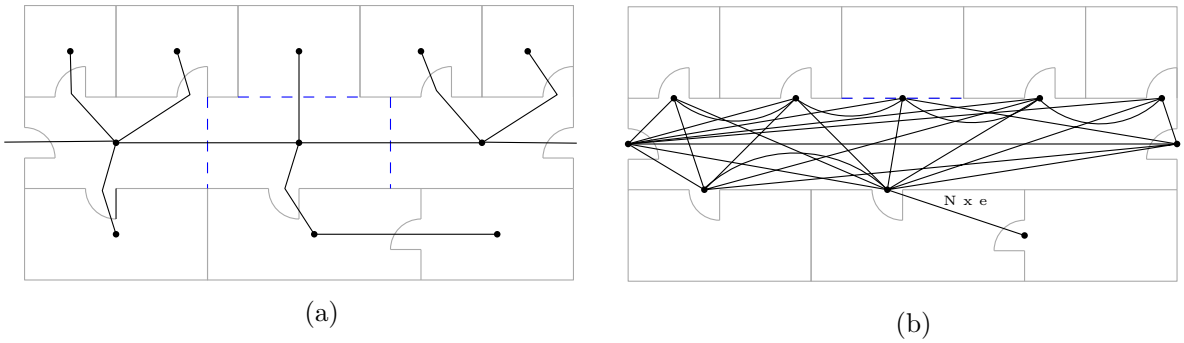


Figure 10: Mapping indoor structures according to (Lorenz et al., 2006)(A) and (Yuan & Schneider, 2010)(B).

However as explained in 3.1Visibility Graph, it is suggested that the optimal path may not always be applicable for wide halls as e.g. for inside terminals, but instead it may be useful to further decompose wide cells into smaller cells and to consider the crossings explicitly in the network in order to provide for a more informative path description. In other words, not only accessibility needs to be explicitly modelled but also containment by including a further information to nodes and edges. (Lorenz et al., 2006) suggests the approach of *interface nodes* whereas areas in buildings are partitioned into a hierarchy of different levels of implicit cells which nodes need to labelled accordingly. For example a terminal would be partitioned into west wing and east wing. Nodes and edges in the graph are labelled with qualitative information (e.g. a node labelled as 'door') and edges with qualitative information (e.g. distance). However, this approach does not include specific nodes at the crossing of implicit cells but instead suggests to label edges and nodes connecting two levels accordingly.

Semantics A simple schematic structuring of indoor space is suggested by OSM. It aims to provide a simple tagging model for the construction of a connectivity graph. It structures a building in levels, corridors, walls, areas and rooms which are connected by doors. Those elements can be provided with more information as e.g. name, height or the suitability for user of a wheelchair.

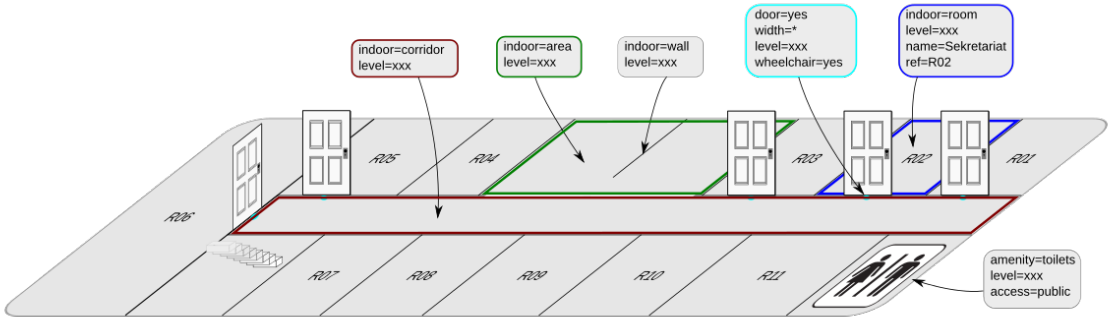


Figure 11: Mapping indoor structures according to the simple tagging model of OSM. Image source : wiki.openstreetmap.org

(L. Liu & Zlatanova, 2012) suggests the Indoor Navigation Space Model (INSM) which is a semantic data indoor model designed for the support of an automatic derivation of the topology of a building by suggesting a specification for nodes and edges considered useful for routing. In general, they suggest to represent spaces as nodes and connections between spaces as edges. According to their approach, indoor space is basically classified into three types of indoor spaces, which are obstacles, openings and navigable space cells. Obstacles are defined as space which cannot be entered by pedestrians while openings are classified as transition space for moving between spaces. Navigable space cells (NSC) can be translated as the space in which people can move freely. Further, vertical NSCs are classified into (1)vertical space (VS), which is space in which people can move freely in vertical directions (up and down) and (2)an abstraction of the aggregation of a group of vertical units (VU), which is a generalization of vertical connectors (VCs) as besides others stairs, elevators and escalators. Besides VCs they introduce the horizontal analog, which they call horizontal connectors(HC). According to their approach, VUs, VCs and HCs are the basis for path calculations.

Further,in association with the above described model, the same author differentiates space into two concepts. Liu(2017) makes a distinction between Space of Interest (SOI) and Point of Interest (POI) for the purpose to make a distinction between the structural building component which the user wants to visit (SOI) and the abstraction of it, namely the target location, represented by a geometric structure as e.g. a single point defined by coordinates (POI)(Liu, 2017) . While a SOI would be a region or area of interest for a user, a POI is a location which is at, close to or contained in a SOI. Examples of an SOI are e.g. areas at coffee machine or the front area at registration desks. Applying this on the airport environment, waiting areas in front of check-in counters may be a SOI and the corresponding check-in counter the POI. As such, a POI is the abstraction of an SOI and adds the most relevant and detailed information, while the SOI is on a coarser level.

Another semantic model is suggested by (Dudas et al., 2009). Their model not only takes environmental structures of a building into account, but the model considers various needs in its semantic, as in the case of e.g. visually impairment or paraplegia. They apply differing use cases with the intention to show even though origin and destination may be the same for different individuals their route may differ due to different user requirements. For example, in an airport a fastest route may differ from a barrier-free route with regard on how to take the vertical passage even though original position and the

desired destination are the same. A barrier-free route may exclude stairways and escalators as those transportation facilities are unusable for people using a wheelchair. Accordingly, in order to calculate a route fitting to the needs of a user, a semantic classification of indoor structures corresponding to the user's needs is advised when building a graph-based network.

3.3 Pathfinding Algorithms

Most studies focus on the shortest path problem with single destination for the outdoor situation. The shortest path problem describes the problem of finding the shortest path between two points in a graph. Most commonly used algorithms for networks with positive weightings are Dijkstra and A*. In fact Dijkstra is a special case of A*. Most studies referring to shortest path algorithms for graphs with positive weights use the shortest path algorithm of Dijktra(Dijkstra, 1959) even though the algorithm of A*(Hart, Nilsson, & Raphael, 1972) outperforms Dijkstra in most cases. ***Figure 12*** gives an overview comparing A* and Dijkstra in general.

Aspects	Dijkstra	A*
Efficiency	-	+
Time complexity	-	+
Optimality	+	+
Implementation complexity	+	-

Figure 12: Comparing most common algorithms for shortest path calculations: A* and Dijkstra

3.3.1 A* algorithm

The A* star algorithm is classified as an informed algorithm which means that it applies a heuristic for a targeted search in order to shorten the run time of processing. The algorithm is complete and optimal which means the algorithm always finds the optimal path, if one exists. A* visits those nodes first which are most possibly closest to the destination. To find the most promising node, each known node x gets a value f(x) which specifies the length of the path from the start to the destination by visiting the relevant node. The node with the lowest f-value is chosen to be visited next. The f-value for each node x calculates at follows:

$$f(x) = g(x) + h(x)$$

where $g(x)$ denotes the true costs from the start node to reach node x and $h(x)$ denotes the estimated costs calculated by applying the chosen heuristic. The A* algorithm calculates the costs of the cheapest paths ensuing from the starting vertex to all other vertices in a graph. It does this by greedily choosing the next vertex. The term greedy here means that the next vertex will be chosen according to its beneficing value. In the case of shortest path finding, the most beneficial next vertex would be the shortest one. Further, the algorithm applies a priority queue. This means that all vertices already visited are stored in a queue, ordered in the way, that the most beneficial vertex, which is the vertex with the lowest cost, is stored as first vertex, followed by the vertex with the second lowest costs etc. During the search, nodes can be in one of three conditions:

unknown nodes Unknown nodes are not found yet which means that the path to the node is still unknown. At the beginning of the algorithm each node is unknown.

known nodes Known nodes are those which are stored in the so-called 'open list' (priority queue) with their f-value. The path to this nodes is already known but it may possibly not yet the optimal. Accordingly, from this list the vertex to be visited next is chosen. At the beginning, only the start vertex is known and stored in the open list.

already visited nodes The most optimal path to these vertices is already known and the vertices are stored in the so-called 'closed list' to exclude them from further inspection. The list is empty at the beginning of the algorithm.

Each known or already visited node includes a reference to its predecessor. By this reference, the path from the currently viewed node to the start vertex can be retraced. During the final visit of node x the succeeding nodes will be added to the Open List and the node x will be registered in the Closed List (and classified as 'already visited node'). Further, the references of the succeeding nodes will be set on node x. If a succeeding node is already registered in the Open List and if the costs of the newly ascertained path is more optimal, then both attributes of the node will be actualized, the f-value and the predecessors. However, if a succeeding node is already registered in the Closed List, neither will it be re-registered in the Open List nor its references changed. The algorithm terminates when the target node is finally visited and the path will be constructed by means of the predecessors and given out.

In the initialisation, the path to the starting vertex has a cost of '0' as the distance to itself is 0. All other vertices get the costs of infinite, as they are unknown yet. During the process of searching, the costs from the starting vertex to each other vertex in the graph shall be improved, by storing the actual shortest path to each vertex. The starting vertex will also be included in the queue. After the initialisation, the first vertex from the queue is considered, which is the starting point as with zero costs it has the lowest costs (while all other vertices have infinite costs). Subsequently, all neighbouring vertices and their edges are considered and checked on the conditions described above. First, if the considered vertex is already in the queue (was it already visited before?), and second, if the costs of the relevant edge are lower than the actual costs to/of the vertex. If the costs are lower, the costs of the vertex will be reduced accordingly. If the vertex was not in the queue, it will now be included, to be able to consider the edges originating from it. Further, the edge which resulted in an actualisation of costs of a vertex is stored as predecessor of that particular vertex. The algorithm will continue to consider vertices from the queue and check their neighbouring vertices and corresponding edges on the above described two conditions until costs cannot be further reduced. Accordingly, the algorithm discontinues when all vertices are visited and when no shorter path can be found.

The used heuristic is only valid if the estimated cost for each node are lower than the true costs. For pathfinding, the Euclidean distance (ED) is considered as a valid heuristic as the costs are in each case lower than the true costs. The Euclidean distance is calculated as :

$$Euclidean\_Distance(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

in which p and q are two points defined by their coordinates $(p_1, p_2, ...p_i)$ and $(q_1, q_2, ...q_i)$.
As stated above, in most cases A* is preferable due to its performance efficiency, however, indoors A* is complex to implement due to the heuristical approach in combination with the additional 3rd dimension which (Bian et al., 2014) defines as *cross storey problem*.

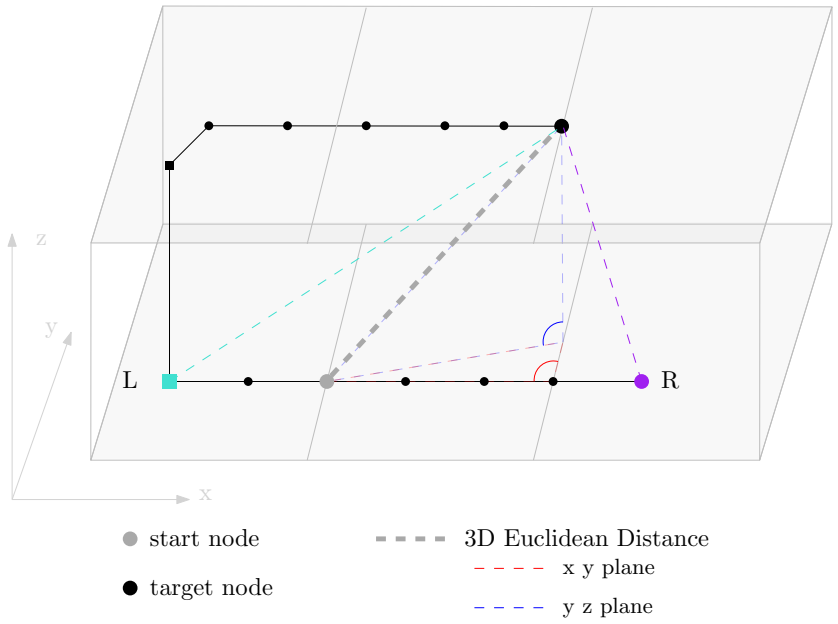


Figure 13: Challenge to implement A* in a 2.5D network

Challenge to implement A* into a multilevel network ***Figure 13*** depicts the challenge which arise when applying the heuristical approach of the Euclidean Distance in A* within a 2.5D network.

As A* bases the estimation on distances between two points in space with the Euclidean Distance, it does not consider topology/geometry. A such, the vertical passages may in some cases constitute a kind of bottleneck, as they are the only connections between two levels thereby only at a few locations available. In ***Figure 13*** the heuristic is misleading as in such a case A* first traverses into the direction of node R (seen from the start node) as the Euclidean distance becomes smaller the more it traverse to node R. However, as only the vertical connector(L) on the left would provide for a pathway from floor 1 to floor 2 the algorithm needs to traverse to the left to find a path. In this case A* with ED would first traverse all nodes on the right side of the red node until it does not find a path. Only then it starts to search the left side, into the direction of node L, which represent a vertical connector. Accordingly, in this situation the heuristical approach would not enhance performance but, instead, may perform similarly to the general Dijkstra, or maybe worse. Therefore, in a 2.5D network, if applying A*, the algorithm needs to consider vertical passages in order to take advantage of the heuristical approach.

4 Method

4.1 Data and Software

The network will be built by using multiple software, proprietary software as ArcGIS, as well as open source solutions as QGIS with the built-in library GRASS, PostgreSQL with the extensions PostGIS and pgRouting. Further, for the evaluation, SPSS, a proprietary statistical software is used for the statistical evaluation.

The network is built from floor plans of the airport Stuttgart. The airport Stuttgart is an international airport in the German province Baden-Württemberg. Measured by passenger volume it is

on rang 8 of the biggest airports in Germany (*Einblicke.Informationen über den Flughafen Stuttgart.*, 2017). The airport area has an area of about 400 ha of which 6000 m^2 is used as shop area and 3800 m^2 for gastronomy (*Einblicke.Informationen über den Flughafen Stuttgart.*, 2017). The airport has five levels which divide into four terminals with 106 check-in counters in the public area and 70 gates in the security area(*Einblicke.Informationen über den Flughafen Stuttgart.*, 2017).

4.1.1 DXF Drawing Interchange Format

The data of the airport Stuttgart is provided in the format DXF, which is classified as a CAD file. CAD files are purely geometric drawings in 2D or 3D which are produced among the field of architecture and in a wide range of other industries. The DXF format is an open source 2D vector graphic file format which was originally specified from the proprietary software Autodesk for CAD data exchange. The introduction of DXF aimed to ensure external interpretable data exchange between CAD systems. Generally DXF stores a combination of lines, polygons, circles, arcs, bezier curves and text. More precise, DXF files store instructions to render drawings, accordingly, the file includes information on how objects should be modelled expressed in mathematical formulae. Further, each data point is plotted using Cartesian coordinates, which are XY coordinates on a grid. Information is encoded in plain text, i.e. each element of the drawing is spelled out in plain text or ASCII format(see ***Figure 14***). Due to this specification, complex DXF files can be up to hundreds of MB in size. For the airport Stuttgart the size of the four files summarizes to 217 MB, of which the size per floor varies between 6 MB and 111 MB. Accordingly, in a GIS, rendering times may be increased.

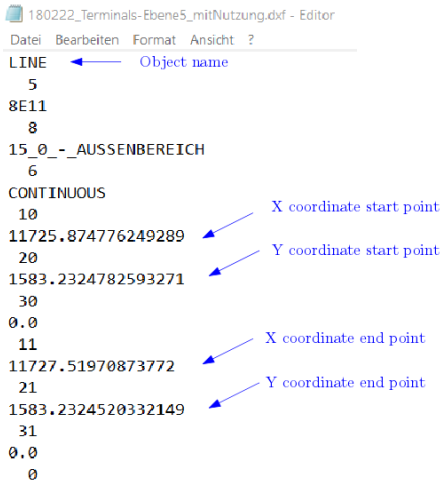


Figure 14: DXF file information storage

Further, typically, CAD files use 2D or 3D Cartesian coordinate systems that locate data at fixed coordinates, however X- and Y-coordinates in DXF are exclusively 2D and further not inherently geographic locations but instead are locations relative to an arbitrary geometric origin (0,0,0). The x-axis can be thought of as an easting direction and the y-axis as a northing direction, but they do not necessarily translate to grid directions in relevant spatial coordinate systems. Although it is possible to create DXF data that correspond to the X-, Y- coordinates of a projected grid zone, most CAD data is created without the consideration of a spatial reference system. Further, in general, CAD drawings are drawn at full scale (1:1), but the decision on the scale can also be determined by the level of detail of the drawing seen as appropriate for the model and as such is up to the decision of the author. In GIS systems as ArcGIS and QGIS the geometry within a DXF file is not a GIS feature class, but instead,

the geometry is translated 'on the fly' to a virtual feature class that resemble the geodatabase schema. In ArcGIS the information is sorted into 5 feature classes: point, polyline, polygon, annotations and multipatch.

4.2 User requirements and corresponding POIs

The route planner should provide the user with a path which considers the task of departing and a such needs to include features of the building corresponding to the task as well as other needs, as e.g. additional user-preferred tasks and requirements. One design approach is the approach of ‚User-centred design' which describes design processes in which potential users are integrated in the design consideration, typically during requirements gathering and/or usability testing(Wickens, Gordon, & Liu, 2004). This with the intention to ensure that the needs of a user are taken into account. Following this approach, prior to constructing a routing service, the requirements of the target user needs to be collected and understood. Accordingly, here the user requirements are listed and structured to guide creation of network and path algorithm for the airport route planner.

4.2.1 Boarding

Im most cases, routing a person through space is a matter of finding some shortest path from an origin to a destination. However, in the environment of airports, destinations within the building depend on strict procedural requirements which need to be completed before arriving at the relevant gate. As a consequence,the intermediate destinations before arriving at the final destination (which is the relevant gate) need to be understood and considered in the routing service. Accordingly, the process needs to be understood prior constructing the relevant network as it determines which building structures are user-relevant and need to be included and emphasized. Accordingly, this section structures the process and relates the subprocesses to the related functional locations, or POIs, within the airport building. The POIs of the boarding process are in the course of the work defined as *required intermediate destinations*(rID). See ***Figure 15*** for a schematic depiction about the boarding process and relevant locations. However, this procedure may slightly differ between different airports.

Within Public Area

Check-In Desk (rID_1) In most cases, check-In counters are the first rID within the public area of terminals after the start location. In general, each airline is represented with more than one check-in counter which are usually adjoining arranged. As it does not matter for the passenger which of the check-in counter of the relevant airline to visit, check-in-counters are considered as *clustered building structures*, of which one cluster represents multiple counters per airline. Some airlines offer self-check-in machines for those passengers who do not have baggage. As the machines appear in the same way as check-in desks (clustering per airline) they are also considered as clustered structures.

Security Control

Ticket Control (rID_2) After Check-In and the reception of the boarding ticket, passengers need to go to the security control. The first step in the security control is the counter for the boarding ticket control. Also here, for the passenger it does not matter which ticket control counter to visit, as long as (s)he is in the correct terminal. As such counters are seen as clustered structures of which the individual counter does not matter. Further, here the passenger leaves the public area and enters the security area without the permission to go back. As such, a relevant path originating from the ticket control is considered to be one-way.

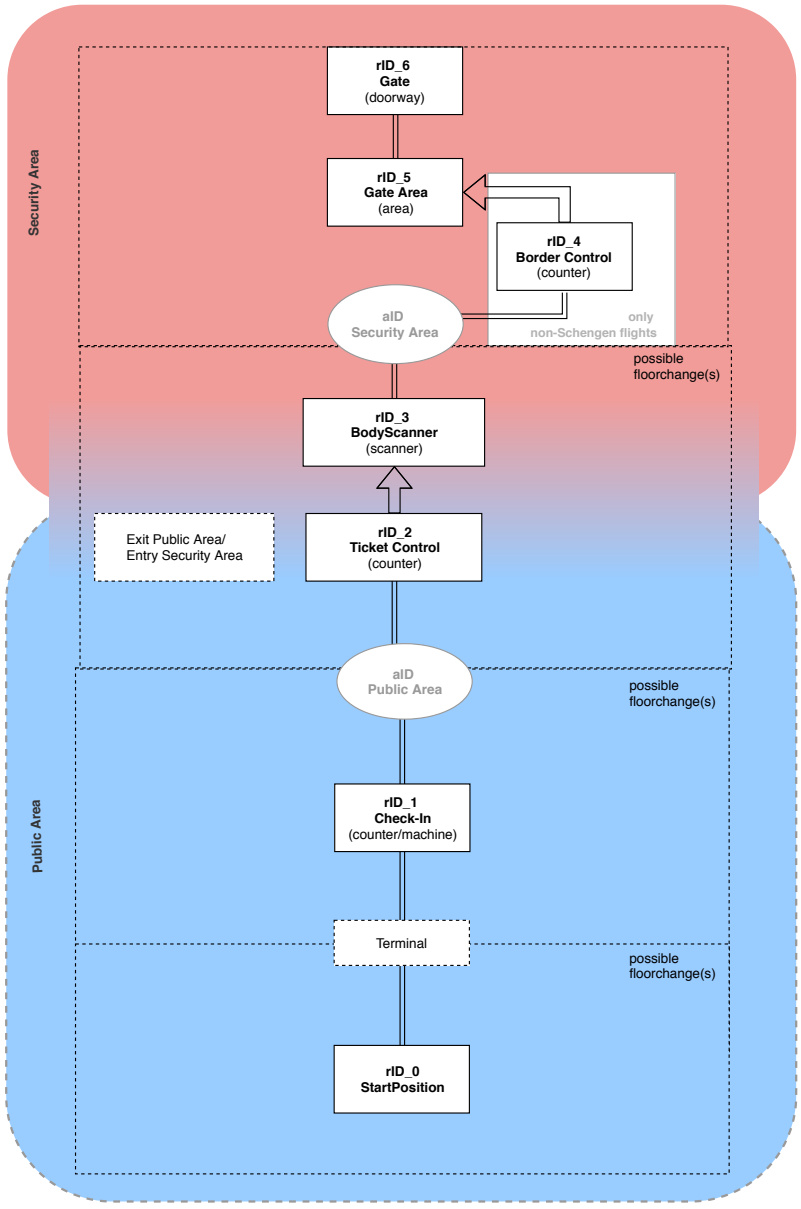


Figure 15: Diagram of the boarding process

Scanner (rID_3) After the Control desk, travellers go through the body scanner. Scanners are usually aligned next to each other forming a sort of chain with separate items. Also here,it does not matter for the procedure which of the scanner to take. Accordingly, also here the scanners are seen as cluster.

Within Security Area

Border Control (rID_4, only for non-Schengen) Passengers taking a non-Schengen flight need to get controlled on their identity, i.e. their pass ports. As a consequence, before entering the gate area of their relevant flight they need to pass the border control. Generally, the border control is represented by two (or more) counters adjoining arranged forming a kind of bottleneck between the corridor and gate areas. As the counters share the same function, they show the same attribute as check-in counters, ticket control counters and scanners, namely as a cluster of structures.

Waiting Area Gate (rID_5) After the security control (respectively border control for non-Schengen flights) passengers go to the waiting area of the gate relevant for their flights. The waiting area is generally a partial open space with on one side a physical boundary which is the airport wall, often a glass wall, aligned to the airfield. The area provides seats for passengers to sit down and wait for their call to board the aircraft. Waiting areas for the different gates are usually adjoining arranged and implicitly separated by an obvious space between the seat rows which connote the two waiting areas for different gates.

Gate (rID_6) Travellers will leave the airport building by the gate for the relevant flight. The gate is generally a doorway through which passengers leave the airport building in order to board the aircraft.

4.2.2 Positioning: rID0

Travellers within the building need to orientate in order to understand their actual position. Accordingly, even though the main focus of the route planner is neither localization nor real-time navigation it needs to provide for start locations to calculate a useful path for the passenger.

For start locations, it is suggested to provide QR codes at main entrances and at significant internal building objects. A QR code ('Quick Response Code') is an optical label, comprised of a two-dimensional barcode which contains machine-readable information. Information can be read by a relevant small software program installed on a smart phone. Often such a program is already preinstalled by standard. For the route planer, QR codes are used to store coordinates from a location x where x is a significant building object which may be considered as a landmark. To identify adequate objects for the provision of QR codes it is oriented on an approach of (Fellner et al., 2017) to identify landmarks. (Fellner et al., 2017) suggests to verify two attributes of potential building objects to identify relevant objects as qualified to be considered as landmarks: recognizability and availability. Recognizability refers to the need that certain characteristics of the structure are visible within the certain area, for example due to shape or size. The second refers to the availability of the structure within the certain area. If a structure is not available within the area while in others it may not be appropriate as landmark. In the airport Stuttgart, next to significant openings as entrances, the architectural trees (***Figure 16***) are considered as significant locations. This as they widely visible due to their size and shape ('recognizability') and relatively regularly spread within the entire public building space ('available'). As such, each tree within the public zone would be provided with a QR code to scan and to give the route planner the current location which to take as rID0 for the path calculation. Consequently, the structures of the trees need to be considered in the network.

4.2.3 Integration of additional activities: aIDs

Passengers may like to use the time between certain boarding steps available for accomplishing other tasks within the airport halls. This may include activities like buying a present at a store, having some drink in the cafe or going to the WC. Accordingly, the corresponding building structures need

Figure 16: Significant architectural structures within an airport for QR code location setting. Here, the tree structures within the Airport Stuttgart. Image source: Sascha Foerster/airlines.net

to be considered in the network, which are called in the current work *additional intermediate destinations*(aIDs).

4.2.4 Time limitation

The period of stay at the airport is limited by the departure time. Accordingly, with arrival at the airport, a passenger enters a time window in which interval he/she needs to accomplish a variety of tasks and activities, most important the boarding procedure. As a consequence, from the perspective of the passenger, time plays a crucial role in the environment of the airport as it affects his/her acting within the building. In dependence of the available time one may consider to do additional task. However, in order to plan additional activities the passenger would need to know the time available. For this the travel time needs to be considered in the routing service. When referring to estimations for the available time, individual differences regarding the accessibility of building structures need to be taken into account as explained in the following paragraph.

4.2.5 Accessibility of vertical passages

A traveller may have requirements or preferences on how to take vertical passages. For example, a person using a wheelchair will need to take a barrierfree route and consequently elevators only, whereas other travellers may prefer to take e.g. only stairs in order to stretch ones legs before the long flight. Accordingly, to take the preferences of the user into account, the route planer should consider

individual differences on how to take vertical passages. The accessibility further plays a significant role with regard to the estimation of the available time. This as a person using a wheelchair will need to take a barrierfree route and consequently elevators only, whereas a person not depending on a certain connector type will be able to take the metrically closest.

4.3 Routing graph construction

Figure 17 shows the requirements on the routing graph which need to consider besides accessibility, geometry and semantics also hierarchical information (containment).

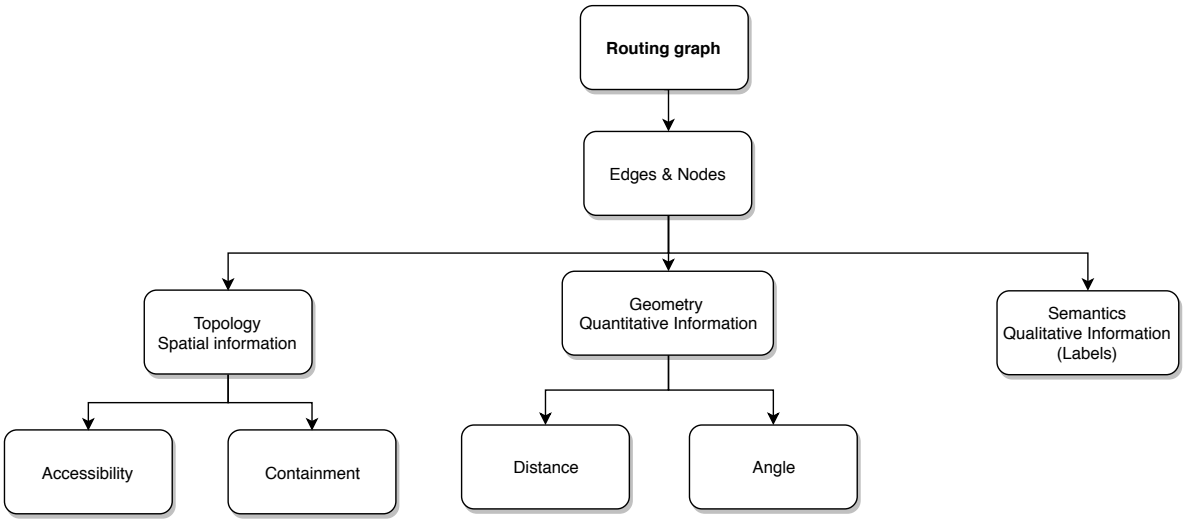


Figure 17: Overview on the requirements on the to-construct routing graph.

From the literature review it was clarified both pathway-generating approaches have their weakness to present both building structures, narrow corridors and wide halls, equally adequately. As airport buildings show both types of structures it is suggested to combine edge-generating approaches to create a spatial network which puts together the complementary strengths of both approaches. Accordingly it is suggested to create a hybrid spatial model based on a combination from visibility graph and centreline, enriched with partially airport-environment specific semantics. While within corridors the centerline approach is applied, for the wider terminal halls a door-to-door path should be created including a hierarchical classification by applying cell decomposition on the internal structure of the terminal halls. ***Figure 18*** shows the expected effect of applying a hierarchy on a visibility graph. Image (a) shows the approach of (Yuan & Schneider, 2010) explained within section 3.2 and (b) shows the current suggested approach, which includes a hierarchy. The path explanation from 1 to 2 in ***Figure 18***a) may be 'Go from 1 to 2.', whereas the path description in ***Figure 18***b) would be 'First go to E, then go to C, within C you reach target 2'. The second explanation follows a hierarchical approach and as such includes more information for a path description.

The addition of information is intended as the graph should provide the basis for the calculation of a path conceptualized to guide passengers not only through the airport building but also through the departure procedure by considering furthermore user-specified needs and requirements.

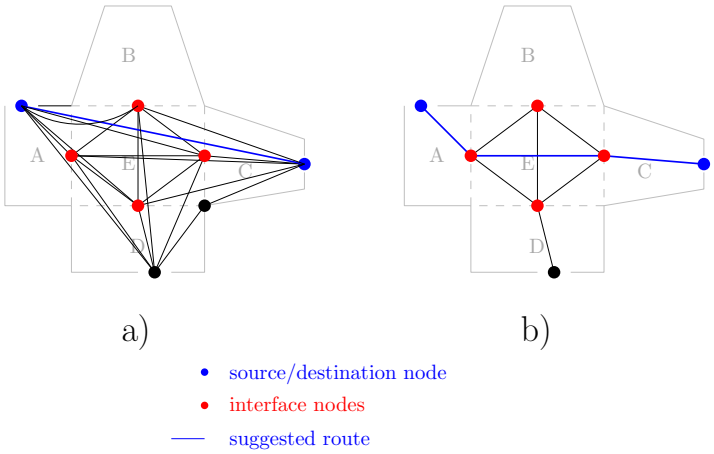


Figure 18: Effect of applying a hierarchy on a graph and on the relevant shortest path calculation. The blue edges indicate the path from node 1 to target node 2.

4.3.1 Data acquisition and preprocessing

The first step in data processing involves georeferencing the DXF files received from the airport Stuttgart.In ArcMap, this can be done by setting control points and creating a 'world file' which contains the coordinates of the relevant two control points. The world file needs to be saved in the relevant folder which contains all floor plans in order to guarantee that each floor plan is georeferenced in one go. Arcmap supports the setting of only two control points for CAD files. Accordingly, the control points need to be selected properly, which is assumed to be one control point of the airport building from the top-right, and one from bottom left.

4.3.2 Identification of Accessible Corridors, POIs and VPs

First,relevant data needs identification a some data may not be accessible for passengers and as such are not included in the network as a accessible structures. (***Figure 19***)

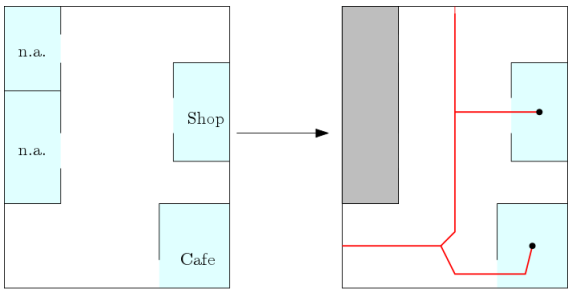


Figure 19: Some structures which are not accessible (n.a.) for passengers are excluded from further processing and i.e. will not be integrated in the network

4.3.3 Space decomposition

For the network creation, the airport space is divided and classified into 4 levels : Floors, Airport Building Spaces, Airport Zones and ActivityAreas (***Table 3***). Level 1 are floors. Floors are divided into

terminals and corridors (Level 2). Further, terminals are divided into public and security zones(Level 3). Zones further contain activity areas(Level 4). Activity areas include adjacent building structures which share the same type of activity. This may be an aggregation of check in counters or, for aIDs, the aggregation of adjacent shops. For example, if multiple different shops are adjacent, they will form one activity area according to their aID function, which in this case may be 'shopping'. Consequently, not all target nodes will be part of an activity area, but instead for lone-standing building structures which represent a certain activity type without any adjacent building structures of the same type, there will be no need to form a corresponding area. Further, within activities areas there may be more than one activity area with the same type. These may be further defined in e.g. wings or just identifiable by a number translating to e.g. 'activity area nr 2 within terminal 3'.

Granularity	**SOI**	**Access Points**
Level 1	Floors	Entrances and vertical connectors
Level 2	Airport Building Spaces	Implicit entries to terminals and corridors
Level 3	Airport Zones	For Security zone: Entry to the Security Control
Level 4	Activity Areas	Implicit entries to clustered structures representing a certain activity, e.g. check-in area

Table 3: Description of the hierarchical levels and corresponding access points

Figure 20 shows an inclusion diagram for the SOIs and POIs for the differing hierarchical levels. For the work at hand, the concepts of SOI and POI (Liu, 2017) are taken up and for the network creation slightly adapted as follows: In general SOIs are those spaces which define the hierarchy and contain POIs. As such, SOIs are spaces nested into each other from the lowest level representing a big spatial extend to a higher level with a small spatial extend. The highest level contain POIs. In this way, SOIs circumscribe POIs. Furthermore, SOIs are represented within the network with corresponding access points marking a junction whether from one hierarchical level to another (for example from a terminal to a waiting area within the same terminal) or between two SOIs within one hierarchical level (e.g. the crossing between two terminals). Further, while POIs inherently can have the function of target and/or connector, this in the case a POI is e.g. a complex cell, nodes to SOIs function exclusively as connector. This as the inclusion of representing SOIs within the network have the only purpose to support the pathfinding process to a certain POI and as such SOIs are in no case the main target but represent areas of differing spatial extend according to their hierarchical level, circumscribing POIs.

4.3.4 Generation and classification of Nodes

Figure 21 shows the classification of nodes for the airport network. Each node in the network besides x, y and z coordinate gets a label in which floor it is located, which area, which terminal (or corridor) and which type of function/activity it serves (whether target or connector).The distinction of nodes into target nodes and connectors is defined according to their purpose. Target nodes are defined as nodes which are POIs or really the main point of interest in order to do some activity at the location of which the target node x represents (the entrance to) the functional location. While connector nodes

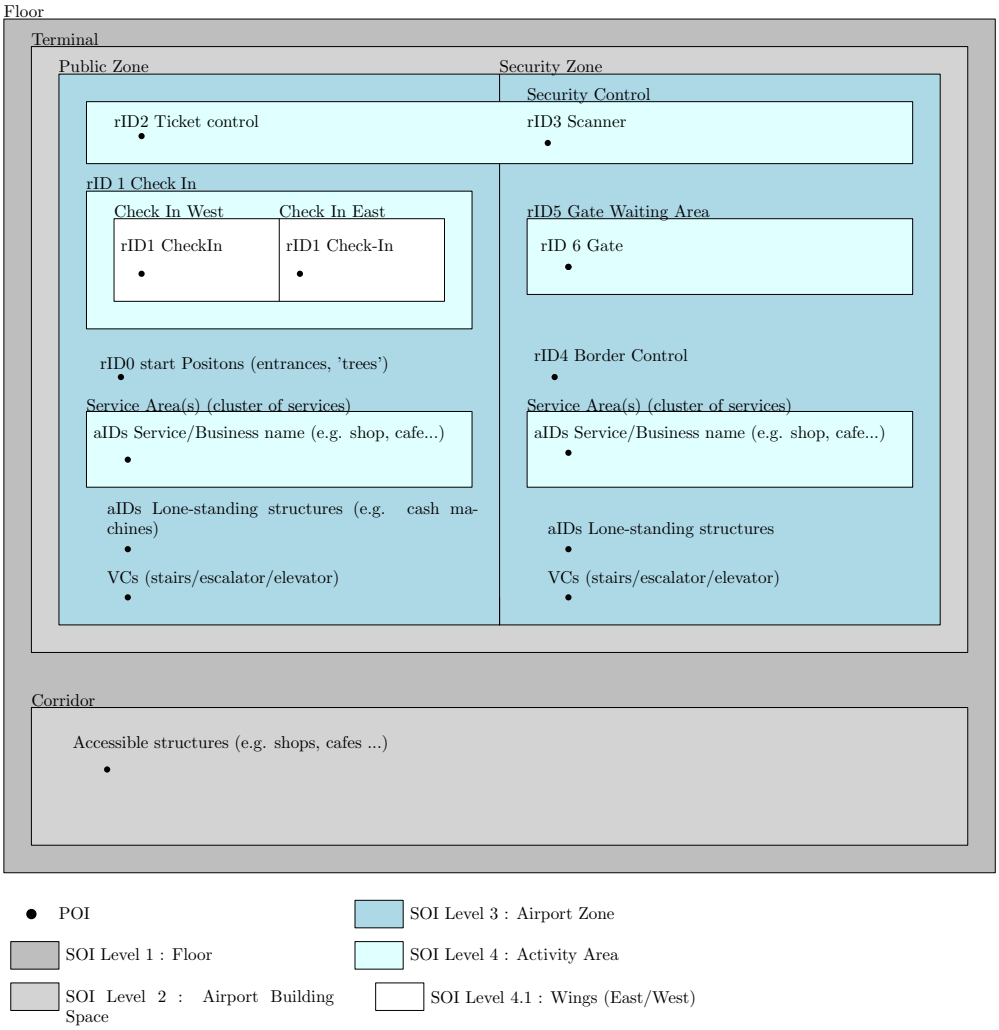


Figure 20: Inclusion diagram of space decomposition

serve the purpose to connect two SOIs.
In genereal the following rules are applied within the generation of nodes from building structures:

Clustering Clustered structures which serve the exact same function and as such for which the individual structure is not of interest are represented as one single node (see ***Figure 22***).

Opening vs centerpoint Room structures are represented according to the approach of Yuan and Schneider (2010), who assume that the entrance, so the accessible opening, is of main interest, not the centerpoint of a room. This approach is followed as it is assumed that the search for a particular room structure (e.g. a WC) will end as soon as a user has found the entrance.

Target nodes: rIDs and aIDs Target nodes further divide into rIDs and aIDs, which are considered the two main classifications of POIs.

Boarding destinations : rIDs

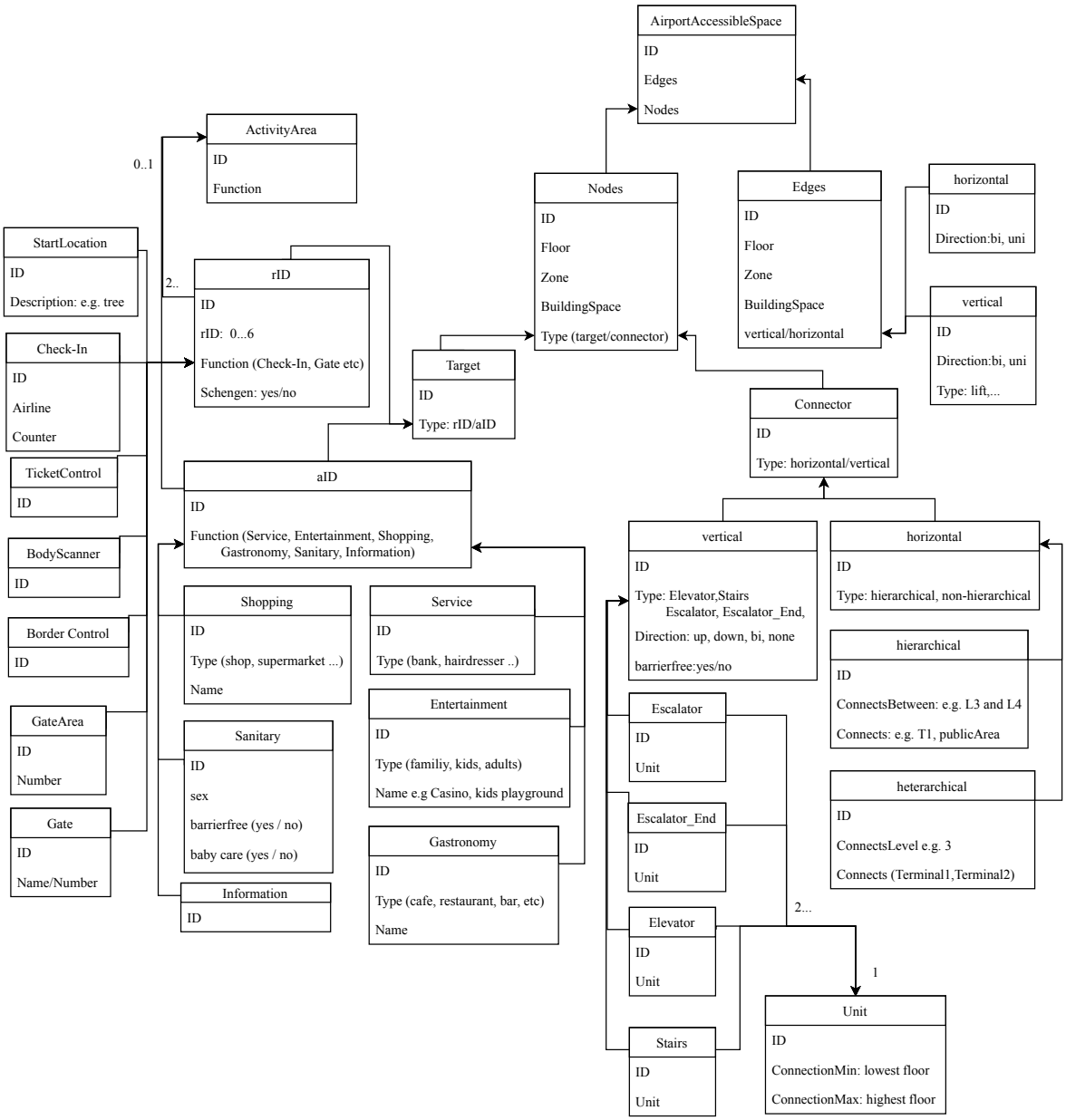


Figure 21: Classification of the network data model

rID0 Each entrance from outside into the airport is considered in the network as a node and labelled as 'rID0'. Entrances are represented by a node at an accessible opening to the airport building as explained. Further, all architectural trees are represented in the network on each relevant floor level. Each tree is represented as a node by its centerpoint.

rID1,rID2, rID3 and rID4 For rID1, rID2, rID3 and rID4 clustering is applied, however the process slightly differs between rID1, rID4 and the rIDs within the security control. Check-in counters

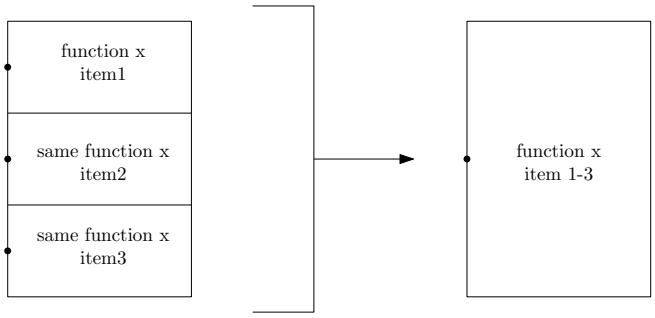


Figure 22: Generation of nodes for clustered structures interchangeable for the passenger

are clustered structures and as such one node represents multiple counters in this case per Airline. For example, if the airline KLM has four adjoining counters, the four structures of the counters are represented as a single node with the label e.g. Airline = KLM, Counter Nr = 001-004. For this representation, a minimum bounding box around the counters is created and a node set at the long side of the rectangle which overlaps with the openings to the counters. As such, the access point is set on the side of the opening of the structures.

rID5 For the waiting area the approach of the open cell of (Yuan & Schneider, 2010) is applied in the case a waiting area is not bounded by physical walls. As such, around the seats of each area first a bounding box is created, and the entrance defined as the centerpoint of the polygon edge opposing the only physical wall aligning the flight field (see ***Figure 23***).

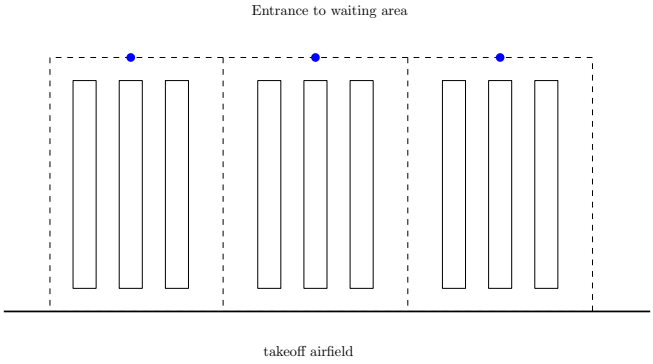


Figure 23: Node setting for waiting areas at gates

rID6 As the gate is a doorway and as such an opening within a physical boundary, the node will be set at the opening. Further, the gate is labelled with its relevant gate number.

Additional destinations: aIDs Each aID will be represented by a single node and classified according to its activity type (service, entertainment, information, sanitary) and labelled with its name. Further, for sanitary facilities are further labelled accoring to their accessibility for people using a wheelchair. The different type of structures are represented according to the cell approach of Yuan and Schneider (2010).

Connectors: horizontal and vertical Connector nodes are further divided into horizontal and vertical connectors.

Vertical connectors (VC) Vertical connectors are represented according to an alteration of the approach of (L. Liu & Zlatanova, 2012) by adding some information of the approaches of (Yuan & Schneider, 2010)) and (Goetz & Zipf, 2011)***Figure 24***. Consequently, VCs are represented as a node

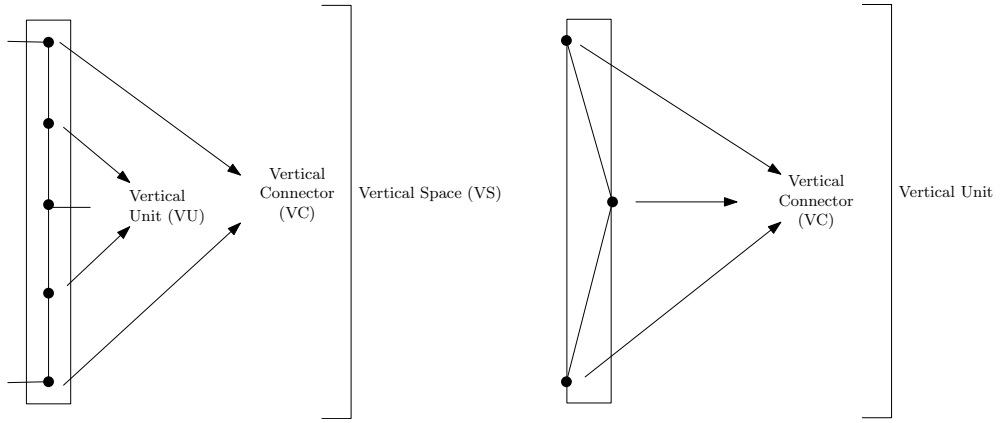


Figure 24: Depiction of the same vertical building structure, on the left interpretation of the concept of (L. Liu & Zlatanova, 2012), on the right the altered concept used within the current network

at the entrance of the connector,i.e. elevator, stairways and escalators are integrated in the network by including the entrance to the relevant vertical facility (e.g. a door to an elevator) by a node in the individual floor network. Further, a group of VCs which are linked to each other, forming a chain of nodes, are integrated into a so-called 'unit'(***Figure 24***). Accordingly, each chain of VCs form a vertical unit and each VC includes an identifier to which unit it belongs to. From the above follows further, that a vertical unit connecting n floor levels will be modelled with n nodes and (n-1) edges. The unit further includes information on the highest and lowest floorlevel which can be reached with the relevant connector unit. The inclusion of the concept of the connector unit is of later importance for the path finding algorithm.

Horizontal connectors Horizontal connectors divide into hierarchical and heterarchical nodes. A hierarchical connector represents a connection between two different levels of the hierarchy, for example from a terminal into an activity area. As the terminal contains the activity area, nodes determined as hierarchical connectors represent the concept of containment. Instead, heterarchical connectors represent accessibility, as the nodes represent a crossing between two adjacent structures accessible from each other and classified to the same hierarchical level. For example the crossing from a corridor into the terminal would be represented as a heterarchical connector, as terminal and corridor both are classified to the same hierarchical level. The distinction between heterarchical and hierarchical is made as a node can represent a crossing within one hierarchical level and a connection between two hierarchical levels. See ***Figure 25***, here the entering into the security control is a connection between public and security zone of which both are in the same hierarchical level (L2), but it is also between the public zone (L2) into an activity area (L3). Accordingly, a node can be hierarchical and heterarchical and as such needs appropriate labels to transfer relevant information to the user within the path description.

4.3.5 Pathway Generation

For the pathway generation the airport is decomposed into terminals and corridors, and handled separately, this as different approaches are applied within the different building structures. While for narrow corridors a centerline is created, for terminal halls a hierarchical direct-path graph is constructed.

Corridors: Centerline For corridor-like structures centerlines are created as pathways. For this, from the DXF files, only polylines which represent wall structures need to be selected and converted to polygons. As such, when linefeatures form a closed area, this area should be classified as a polygon. Then, polygons of corridors need to be converted to Linestrings to represent inner and outer polygon edges. Then these edges need to be converted to points to feed a Voronoi algorithm. However, a suggested simple method would be to use some tools to approximate the centerline. One open source solution is to create centerlines from polygons with the PostGIS function St_ApproximateMedialAxis. The function auto-generates a centerline from polygons without the need of converting polygons to points.

Terminal halls: Direct-path graph Within terminals, a a direct-path graph is generated. For it, within the terminal halls, each pair of connector nodes are linked with each other, except for connector nodes of activity areas which serve the same function within the boarding procedure, i.e. two activity areas which serve the check-in are not connected. This, as passengers will need to visit exclusively their relevant check in counter, without the need to visit other counters in another check-in area. Figure ***Figure 25*** shows the effect of including a hierarchy within a terminal, with two check-in areas and one activity area. In image (a) each object is connected with each other without the inclusion of a hierarchy. Image (b) shows the cell decomposition, in which the space is decomposed into corridor and terminal. The terminal space is further decomposed into activity areas. In the next image (c) connector nodes are linked with each other by edges and the target nodes connected to their relevant connector node. Here, the distinction is further clarified between hierarchical and heterarchical. While the entry in an activity area needs a virtual entrance from the terminal area into an activity area level, which resembles a change between hierarchical levels, the entrance to the terminal from the corridor is of heterarchical type, i.e. the passenger does not enter a new level in the hierarchy but only crosses between two structures within the same level of the hierarchy. This distinction is made due to the differing topology a relevant connector node represents. While a hierarchical connector represents accessibility and containment, a heterarchical connector solely represents accessibility.

4.3.6 Edge classification

Vertical and horizontal edges explain intuitively: vertical edges represent connections between vertical connectors and horizontal edges are the horizontal counterpart. Further the direction of edges need to be specified, as between some locations there are only one-way connections.

Horizontal edges Within the security control, horizontal edges need to be specified according to their direction as one-way. This as within the security area people move from the ticket control to the body scanner without the option to return. As such, this pathway needs to be one-way and needs to be specified accordingly. The same circumstance occurs for passengers of non-Schengen flights at the border control, as passengers once passing the border control will not be permitted to go back once in the waiting area.

Vertical edges Only VCs of the same type are connected by vertical edges. Further, edges of stairways and elevators will be bidirectional. However, in the case of elevators, edges are directed

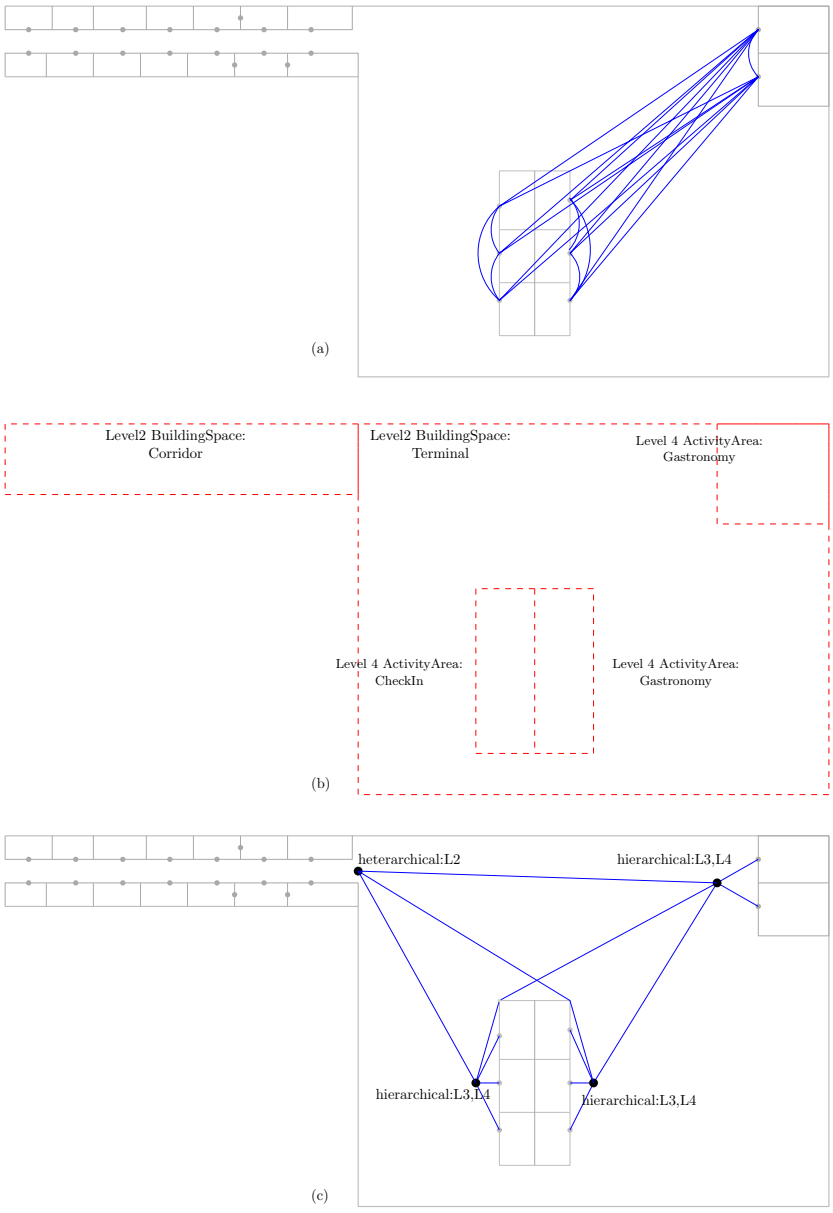


Figure 25: Comparing depiction of a direct-path(blue lines), the top image shows an ordinary direct path while the bottom shows the graph after the inclusion of a hierarchy.

according to their move direction (***Figure 26***). This, as escalators move into one vertical direction only, whether up or down.

4.3.7 Introduction of a Z-coordinate

After vertical egdes are added in order to connect the VCs, a Z-coordinate per floor is introduced for edges and nodes. This, to represent the elevation difference between floors and to provide the correct metrical distance between the VCs.

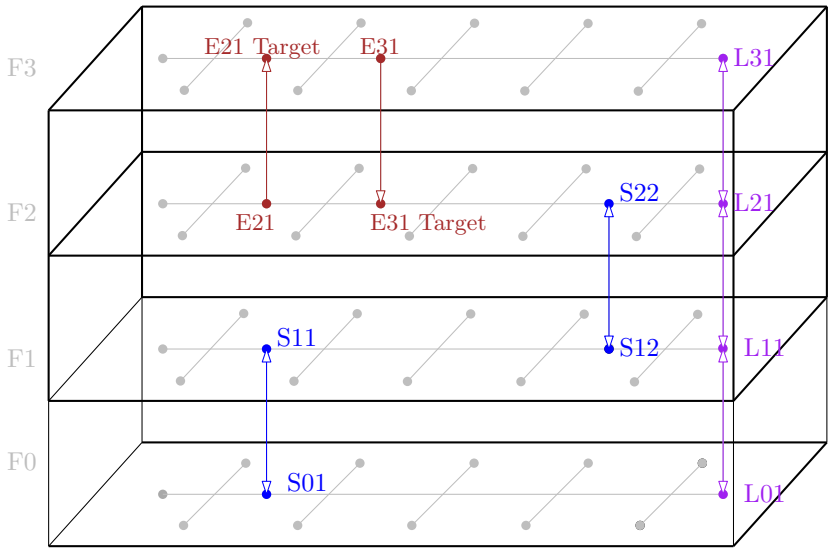


Figure 26: Different types of vertical connectors and responding edge directions. Abbreviations: S = Stairs, L = Lift, E = Escalator

4.3.8 Topology checking and data validation

To ensure the topological completeness of the model, topology is checked with relevant rules available in the relevant GIS.

4.3.9 Generation of network for each floor

For corridors, the nodes are snapped to the nearest point on the centreline to create a complete network for the corridors. Further, corridor networks and terminal halls are merged and connected in the network by a horizontal connector node.

4.3.10 Generation of one 2.5D Network Dataset

All floor network datasets are merged into one single routable 2.5D network dataset. It should include nodes of rIDs, aIDs, and VCs, and edges from to the centreline and the visibility graph.

4.4 Routing Algorithm

In essence, both suggested following path finding algorithms first try to find a path into the destination floor level to then calculate a path from the VC in the destination floor level to the the target node. However, the approach in determining the intermediate destinations differs between the algorithms. While the boarding path receives the intermediate destinations in the correct sequence in a list, Activity path first need to find appropriate intermediate destinations.

4.4.1 Boarding path

The calculation of the boarding path aims to find a short path between the actual start location rID0 and the last boarding destination which would be rID6, by further considering all rIDs. Therefore shortest paths are calculated for each pair of rID and finally merged into one path. For the shortest path calculations A* will be applied. This because of its informed search it is expected to find a short

path within a shorter time frame than Dijkstra. However, to implement A* on a 2.5D network requires an adaption of the pathfinding algorithm to the concept of A* as described in a previous section.

Network profiles To consider user preferences on how to take the vertical passages, four different network profiles are considered within the algorithms by initializing the network according to it (Table 4). Accordingly,the initialized network will exclude some VCs, in dependence of the chosen network profile.

Abbrev.	Description
f	fastest (VC type: all)
wd	walking disability (VC type: elevator)
co	comfortable (stairs where direction = down, elevators, escalators)
sp	sporty (stairs)

Table 4: Description of Network profiles

Boarding Path Here it follows a description of the pseudo code for the boarding path algorithm 1, in which the `verticalPassages` is called as helper function. Within the function `verticalPassages` the described cross storey problem is handled.

The `verticalPassages` function is primarily called by `buildPath`, and applies the concept of recursion to calculate a short path (not per se the shortest path) between the two given locations `S` and `D` by applying A* with the heuristic of euclidean distance. Furthermore, it considers vertical passages if `S` and `D` are not in the same floor. Refer to ***Figure 27***a where `S` is in floor 1 and the relevant destination location `D` is in floor 3. In this context, `verticalPassages` calculates a path from `S` to `D` by taking relevant vertical connectors as intermediate destinations.

`verticalPassages` first checks whether `S` and `D` are in the same floor, if not it selects all vertical connectors (VC) in the floor of `S` which go into the appropriate direction and stores them in a list. Then it calculates the euclidean distance from `S` to each VC. The list will be ordered in-place according to the Euclidean Distance, starting with the VC with the shortest distance to S. In the next step, it selects the VC from the list at index 0 (which is the first element) and sets the selected VC as new (intermediate) destination ('dvc')(***Figure 27***b). Then is applies A* to calculate the shortest path between the start location S and the selected VC (***Figure 27***c) and stores the path between them. In the next step it selects all VCs in the unit of dvc and selects the vc in the unit of which the floorlevel is closest or equal to the floorlevel containing the relevant destination node D. It sets this closest vc as new starting location, and, further, adds the edge between dvc and newdvc, to the already calculated path(***Figure 27***d). In case the newdvc is not already in the same floorlevel as the target level, it calculates again for all VCs within the relevant floor the EDs and selects the closest VC to newdvc. Then it calculates a shortest path with A* but this time it does not take the start location as starting point, but instead takes newdvc as start location and the closest vc in the same floor as target location and adds the edge between the relevant VCs (dvc and newdvc) (***Figure 27***e). This procedure repeats (***Figure 27***e-f) until newDvc and D are in the same floor. Accordingly, the recursion terminates when the floor of newdvc is equal to the floor of destination D. If newDvc and D are in the same floor A* will calculate a shortest path between the two nodes(***Figure 27***g). No extra edge will be added in this case as no vertical connectors need to be taken.

Algorithm 1: Boarding path

Data: api, actualPosition, flightNumber, network, profile
Result: Path from actualPosition through all required intermediate destinations relevant for boarding with the gate as final destination

Function *main()*
    network.init(profile) `// initialize network according to chosen network profile`
    destinationList = [] `// [] means empty list`
    destinationList.append(actualPosition) `// append adds a single item to the list`
    rIDList = api.getrIDList(flightNumber)
    destinationList.extend(rIDList) `// extend adds multiple items to the list`
    path = buildPath(destinationList)
    return path, convertToMinutes(path)

Function *buildPath(destinationList)*
    boardingPath = initPath() `// create a new empty path to work with`
    for $i \leftarrow 0$ **to** $len(destinationList) - 1$ **do**
        intermediateS = destinationList[i]
        intermediateD = destinationList[i+1]
        `/* merging a path to another path means to combine both paths to a single path as one would expect */`
        boardingPath.merge(verticalPassages(intermediateS, intermediateD))
    end
    return boardingPath

Function *convertMetersToMinutes(path)*
    time = path.cost/50 `// average walking speed 3km/h (3000m/60min = 50)`
    return time

Function *verticalPassages(intermediateS, intermediateD, pathSegments = initPath())*
    `/* pathSegements = initPath() means that an empty list is used for pathSegments if the caller didn't used this parameter. */`
    s ← intermediateS
    d ← intermediateD
    if s.floor = d.floor **then**
        pathSegments.merge(aStar(s, f))
        return pathSegments
    else
        vcs = selectVcs(s.floor)
        sortByEuclideanDistance(s, vcs)
        dvc = vcs[0]
        pathSegments.merge(aStar(s, dvc))
    end
    if d.floor in dvc.unit.selectAvailableFloors() **then**
        newDvc = dvc.unit.getVC(d.floor)
    else
        newDvc = select vc in dvc.unit where floor closest to d.floor
    end
    pathSegments.merge(newDvc)
    return verticalPassages(newDvc, d, pathSegments)
,

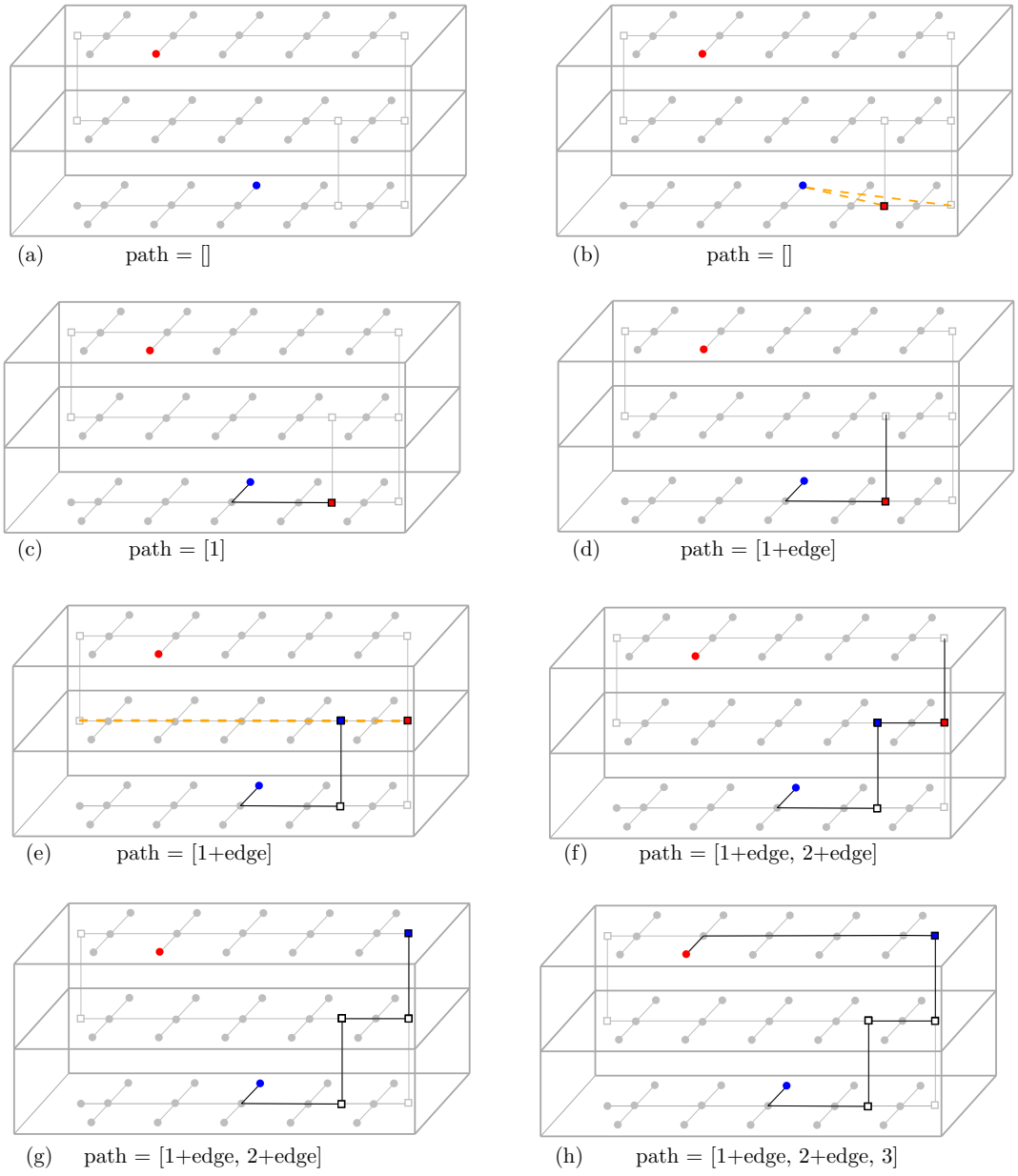


Figure 27: Calculation of a short path between two nodes by considering vertical passages

4.4.2 Activity path: Integration of additional destinations

In order to explore the airport, travellers may integrate additional activities (aIDs, as e.g. a cafe, cash machine or WC) into the calculated boarding path. As such, after the boarding path including all rIDs is calculated, additional destinations, chosen by the traveller, can be integrated. Thereby, three circumstances need to be taken care of within the calculation of the activity path(see (***Figure 28***).

1. Theoretically there are two time moments in which aID's can be accomplished .
   (a) time moment A between rID_1 and rID_2
   (b) time moment B between rID_4 and G (Non-Schengen: rID_4 and rID_5)
2. The time moments determine the area in which the tasks can be accomplished:
   (a) time moment A : public area
   (b) time moment B: security area
3. Some tasks have multilocal options.For example if the user wants to visit the WC, within the airport are multiple toilets available of which one needs to be chosen.

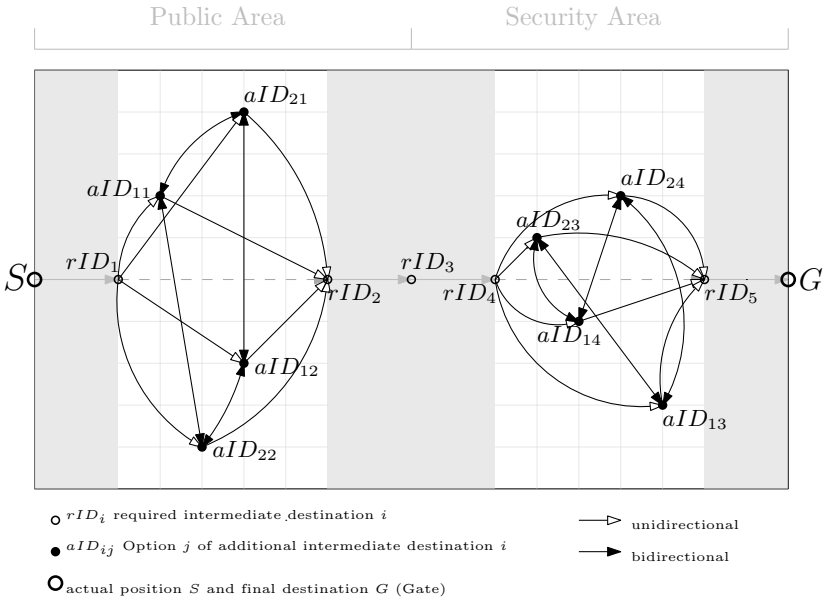


Figure 28: Calculation of a route considering in both areas (Public and Security) two aIDs (aID_1, aID_2) with each multilocal options(aID_1j, aID_2j).

The main function gets four parameters:

- the area chosen by the user (public or security area),
- the user's activity wishlist, which includes the preferred activities selected by the user,
- the list of relevant rIDs (already used within the boarding path calculation)
- the calculated boarding path

Algorithm 2: Activity path

Data: network, profile, chosenArea, userWishlist, rIDList, boardingPath
Result: Path from actual position through all required and additional intermediate destinations with the gate as final destination

Function *main(userWishlist, chosenArea, rIDList, boardingPath)*
    network.init(profile) **if** chosenArea == "public" **then**
        activitySegment = buildPath(findaIDs([rID1, rID2], userWishlist))
        activityPath = boardingPath.replaceSegment([boardingPath.rID1_rID2], activitySegment)
    end
    if chosenArea == "security" **then**
        if flight == "Non-Schengen" **then**
            activitySegment = *buildPath*(*findaIDs*([rID3,rID4], userWishlist))
            activityPath = boardingPath.replaceSegment([boardingPath.rID3_rID4],
             activitySegment)
        end
        activitySegment = buildPath(findaIDs([rID3, rID4] userWishlist))
        activityPath = boardingPath.replaceSegment([boardingPath.rID3_rID4], activitySegment)
    end
    return activityPath

Function *sortByEuclideanDistance(referenceNodeList, aIDList)*
    Sorts the nodes in aIDList according to the sum from the Euclidean distances from the relevant aID the reference node(s)

Function *findaIDs([referenceNode1, referenceNode2], userWishlist)*
    aIDList = userWishlist.select_aID WHERE aID.type in userWishlist
    referenceNode1 $\leftarrow rID_i$
    referenceNode2 $\leftarrow rID_{i+1}$
    foundaIDs = []
    while aIDList $\neq$ [] **do**
        sortByEuclideanDistance([referenceNode1, referenceNode2], aIDList)
        selectedaID = aIDList[0]
        foundaIDs.append(selectedaID)
        aIDList.remove_all WHERE "type" = selectedaID.type)
        referenceNode1 = selectedaID
    end
    return foundaIDs

Activity path The main function selects the rIDs from the rID list appropriate to the area in which the aIDs should be searched for (whether public or security area). The relevant rIDs are given as parameters to the `findaID()` which selects the aIDs relevant for the aID path. The process of `findaID()` is described in detail in the next paragraph. The `buildPath()` is called and takes the results from `findaID()` as parameters and returns the activitySegment, which is a path from rID_i to rID_{i+1} inclusive relevant aIDs. replace_segment() then replaces the relevant segment in the boarding path between the relevant rIDs by the activity segment and returns the activity path, which is the complete boarding path including aIDs.

Figure 29 describes the procedure of selecting appropriate aIDs for three preferred activities which have multi-local options. For example, a user wishes to visit a café, a toilet and a cash machine before going to the security control. First, the algorithm takes two reference nodes, which are the nodes to which the to-selected aID should be close to. In the given example the two first reference nodes would be rID1 and rID2, so check-in counter and ticket control, as those are the determined boarding locations within the public area in-between additional destinations can be carried out. Secondly, all aIDs are selected which type correspond to the user's preferred activities and are stored in a list. For example, in ***Figure 29***a, let's suggest the white squares are all aIDs with gastronomy type = cafe, the crosses are all aIDs with sanitary type = toilet, and the black squares are aIDs with service type = cash machine. After setting the reference nodes and selecting relevant aIDs the algorithm calculates the Euclidean distance from both reference nodes, S and D, to each aID in the list and selects the aID_{ij} which has the shortest Distance sum calculated from the distance S to aID_{ij} and the distance from D to aID_{ij} (image b). The selected node will be stored as as aID_1 in the foundaID list. Furthermore,aID_1 replaces the check-in counter as new reference point. All nodes which type = type of aID_1 are removed from the the aIDList (***Figure 29***c). In other words, if aID_1 is a café, then all aID with gastronomy.type = café will be removed from the list and as such excluded from further processing. In the next step aID_1 will serve as the new reference point (image d) while the procedure will repeat until the List with aIDs is empty , i.e. until for each activity the user wishes to visit one appropriate aID is found(***Figure 29***g). ***Figure 29***e shows when the path is calculated from rID1 to rID2 inclusive the found aIDs. As it can be seen, the resulting path is conceptualized in the way that the user slowly recedes from rID1 (check in counter) while concurrently approaching rID2 (ticket control within security control). In this way, the calculated path represents a small detour within the boarding path, using available time for additional tasks.

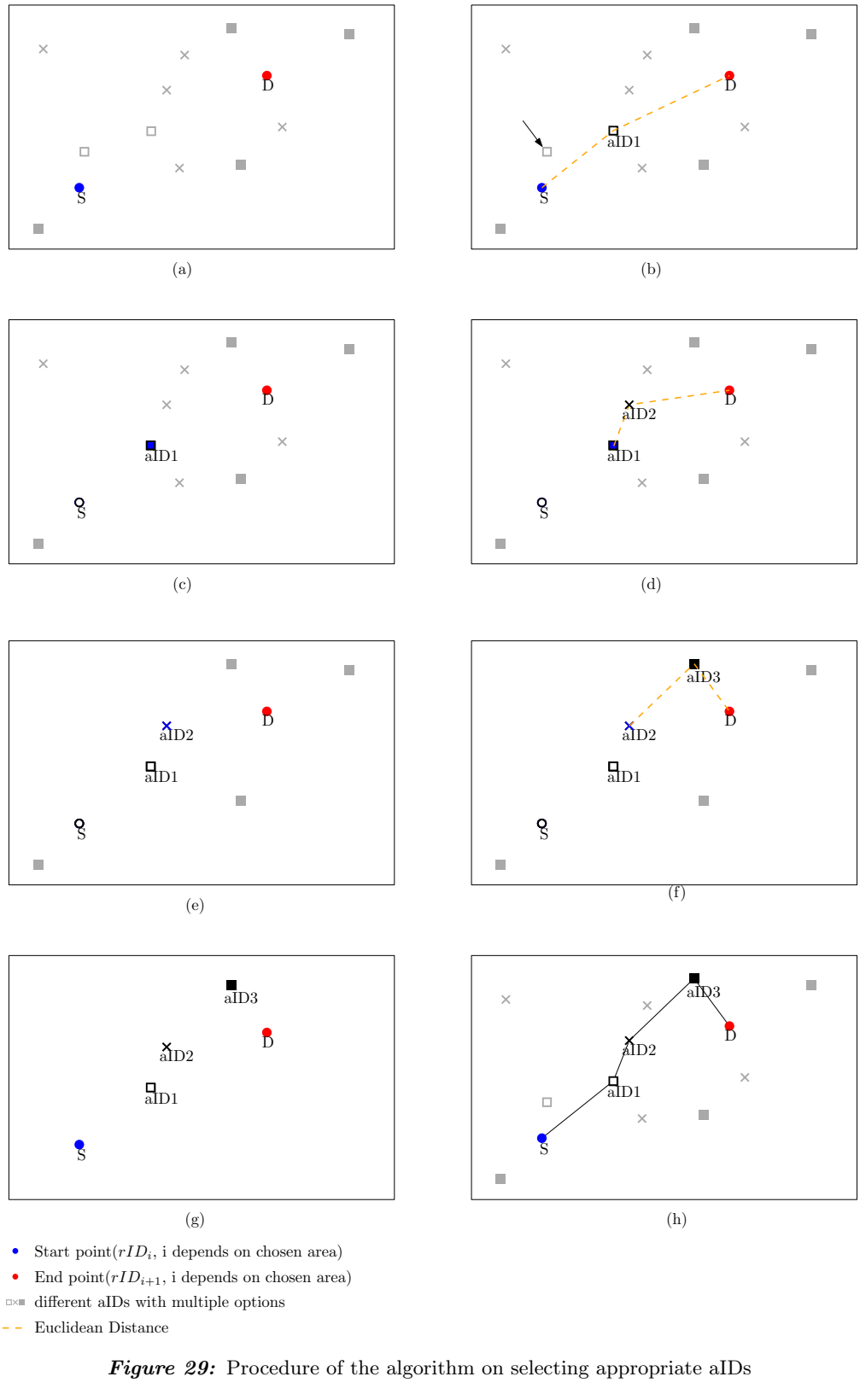


Figure 29: Procedure of the algorithm on selecting appropriate aIDs

4.5 System requirements and architecture

Figure 30 shows the suggested information flow of the route planner.

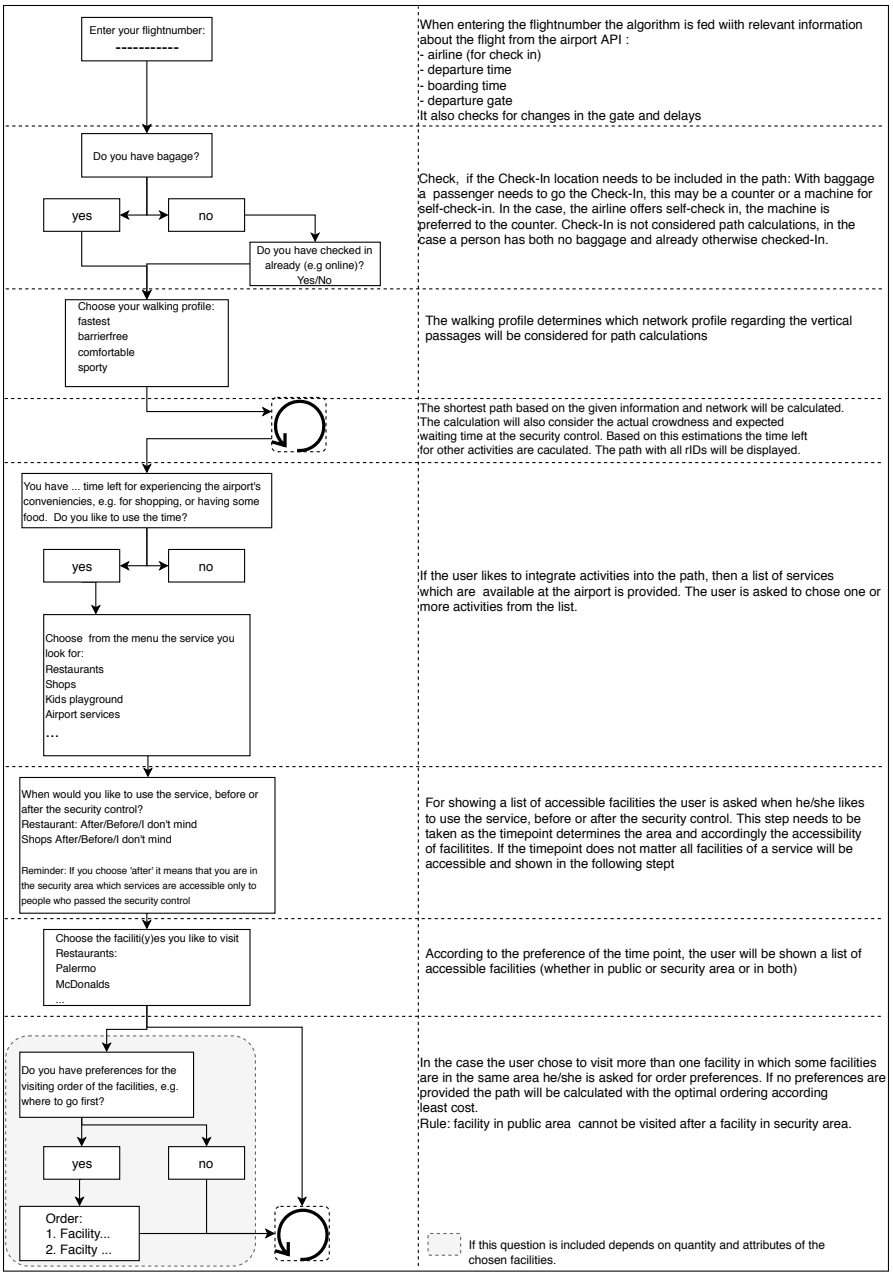


Figure 30: Information flow of the suggested route planner

The user needs to provide relevant information via the interface, as e.g. the flight number and additional destinations , in order to feed the pathfinding algorithm. However, information on the relevant destinations for boarding are stored within the airport database and as such, the application needs to be able to communicate with the airport database to receive relevant flight information. Further, the

application needs to be able to communicate with a spatial database to initialize a network for routing.

To set up the route planner as a web application the system architecture is suggested as in ***Figure 31***.

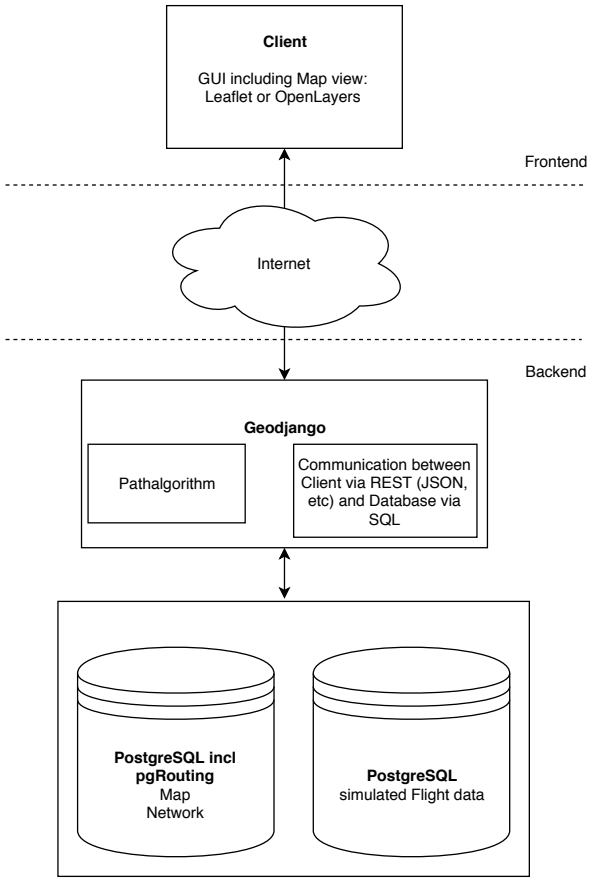


Figure 31: Suggested system architecture of the route planner

The architecture is split in frontend and backend. The frontend is a typical web application running in a browser. It uses standard web technologies like HTML5, CSS and JavaScript to present the airport map to the user (Leaflet or OpenLayers) and to communicate with the RESTful backend over the internet. The backend is based on Django and the GeoDjango package for geospatial web application development. The backend provides a RESTful API (a standardized method to communicate via HTTP) and deploys the path algorithm. REST describes an architectural paradigm (or an "API style") for HTTP-based webservices. It is not necessary to describe REST's concepts, but it became a solid style implemented by various web frameworks (typically used in the backend) and client libraries (typically used in the frontend). Thus, it can be used to establish a well-defined communication channel between the frontend and the backend.The backend communicates with a PostgreSQL database extended with PostGIS and pgRouting . The data for the airport API is provided by a mock database due to the lack of a real database.

4.6 Testing

4.6.1 General Performance

First, in order to give a descriptive overview on the performance of the route planner, more general tests are conducted. This include calculation times, the correctness and completeness of the paths regarding the inclusion of all rIDs and aIDs and the influence of the differing network profiles regarding the chosen vertical connector type on resulting calculated paths lengths.

4.6.2 Usability and Usefulness

Second, tests on usability and usefulness are conducted. Further, it is analysed if there is a specific target group for which the route planner may be especially valuable. For this tests, participants need to be recruited. Usability tests are conducted by applying a combination of questionnaires and structured interviews . The concept of usability refers to the degree to which the system is easy to use or 'user-friendly' (Wickens et al., 2004). Usability is tested by considering the variables of learnability, errors and satisfaction (Nielsen, 1993). Concretely, the route planner (1) should be easy to learn so that the participant gets to the desired route quickly, (2) should have a low error rate in the way that the participant makes few errors during the use and if an error occurs the user should easily find a solution for the error (e.g. the participant chooses accidentally the wrong network profile but knows how to correct this with selecting the relevant network profile) and (3) should be pleasant to use by the participant, in other words, they like it. For this, participants are asked to first use the route planner and afterwards fill in a questionnaire. The first three question treat the usability factors (learnability, error rate, satisfaction), for each factor one question. A Likert-scale will be used as response system. For example, the question for learnability may be 'How easy did you get to see your desired path?'. The relevant response would range from 1 till 5, of which 1 signifies 'very difficult' and 5 'very easy'. The other two questions would be similarly arranged. A fourth question is included to get to know about the usefulness as regarded by the participants, more concrete, if there would be a need for an indoor route planner for airports in general. The question may be formulated as e.g. "How useful do you find an indoor route planner for airports?" Control questions are included in order to understand the eventual primary target group for an airport route planner, in other words for whom the airport route planner may be especially useful. It is considered that people with low flight experience and people with high use of smartphones may value the route planner more and as such establish the target group. Consequently, control questions will cover the level of flight experience and the use frequency of smart phones. Further, as literature suggests that spatial anxiety (which is nervousness in unfamiliar environments) influences wayfinding negatively (Farr, Kleinschmidt, Johnson, Yarlagadda, & Mengersen, 2014), it is considered that the route planner may be valued higher by people scoring higher on spatial anxiety. As the route planner is considered to reduce received stress (see Introduction) it may be concluded that is also helps to reduce nervousness. For this, an exploratory study will be conducted on the relationship between spatial nervousness and received usefulness of the route planner for the relevant participant. Spatial anxiety will be measured by the revised Spatial Anxiety Scale (Lawton & Kallai, 2002). The questionnaire includes 8 items with responses ranging from 1='not at all nervous' to 5='very nervous' (see ***Figure 40***). Further, measured ratings of usability and usefulness of the routing service will be correlated with measured spatial anxiety as well as with smartphone use frequency and flight experience. The tendencies regarding correlations may give insight in the primary target group for the route planner. If measured usability correlates (or tends to correlate) positively with measured spatial anxiety it may suggest that potential user receiving more stress in unfamiliar environments consider the route planner as more beneficial than people without spatial anxiety.

5 Results

The main research question of the current work was :

How to build a route planner for the indoor environment of airports considering the needs of departing travellers?

To answer the question included a theoretical and a practical part. The main contribution of the current work refers to the theoretical part thereby answering the first three subquestions.

How to build a routable graph-based network of indoor space with regard to spatial, quantitative as well as qualitative information? It is suggested to create a hybrid spatial model based on a combination from visibility graph and centreline, enriched with partially airport-environment specific semantics and a hierarchical classification applying an appropriate cell decomposition of the internal structure of the airport. More specific, it is advised to create an accessibility graph which applies a hierarchy and as such suggests to decompose wide areas into cells which are represented in the network with their access points. Further, it is advised to use two different methods on edge creation determined by the type of space. Accordingly, for narrow spaces as corridors the creation of a centreline is suggested, while in wider spaces as the terminal halls, a door-to-door path is considered to be more adequate.

Which algorithm(s) is(are) adequate for indoor routing? Two algorithms are designed of which algorithm 1 aims to calculate a boarding path and algorithm 2 a path with additional destinations, named activity path. The path algorithms correspond to theory as well as user requirements specific for departing travellers. It was concluded that for shortest path calculations the heuristical approach of A* would be of first choice compared to Dijkstra, due to its performance. However, for implementing A* in a 2.5 D network a helper function VCpath was designed, which is integrated in both algorithms to handle A* in the 2.5D network (***Figure 32***). Further, the boarding algorithm was designed in the way that it considers the boarding destinations as intermediate destinations, and aims to give out a complete path from start point to the relevant gate by considering also vertical passages. In the end, the algorithm gives out the time needed to walk the path. Further, the network on which to base the path is initialized by considering user's needs on vertical connector type.

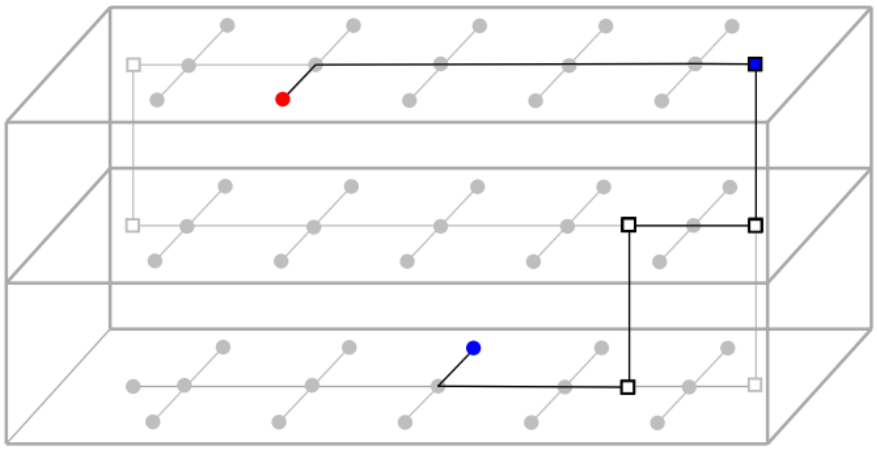


Figure 32: Algorithm 1 and 2 helper function VCpath: Short path calculation based on A* by considering also vertical passages

For the activity path a helper function findaID was designed which suggests a naïve approach on how to select a close aID from multilocal aIDs without the need to first calculate all shortest path as suggested in other literature. As such, the activity algorithms does not aim to give out per se the most shortest path but instead a short path which further navigates passengers implicitly into the direction oft the next required boarding destination(***Figure 33***).

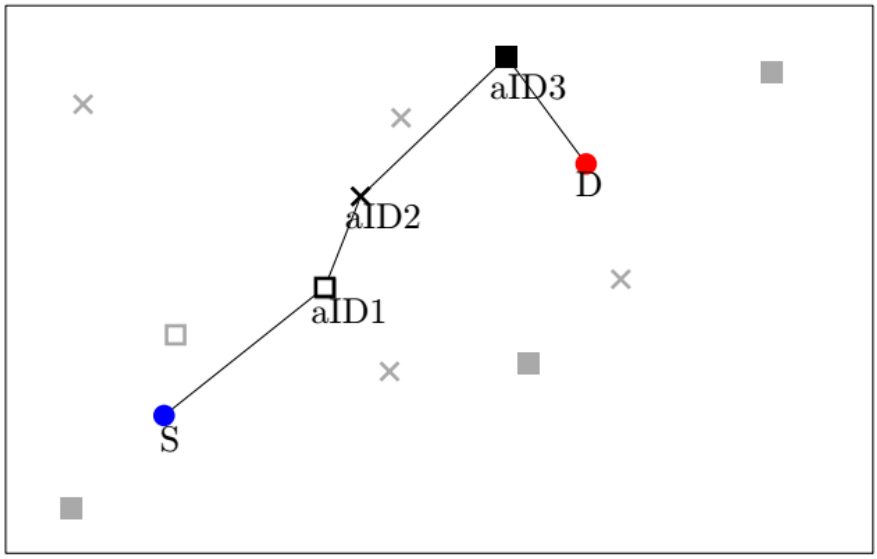


Figure 33: Algorithm 2 helper function findaIDs : Selecting close aIDs by implicitly navigating the user to the next boarding location D

.

What are the systems requirements to build the desired route planner?
Besides a user interface, the system of the suggested route planner needs to provide for the communication between passenger, airport API, the Postegre database (which provides the routing network) and the scripts of the path algorithms. The suggested architecture for the geospatial web application includes the REST framework, Geodjango and PostgreSQL with the extension PostGIS and pgRouting.

The last two subquestions relate to the pratical implementation of the route planner which was cannot be answered yet due to time limitations and unexpected challenges with the data (pre)processing. Some cautious recommendations and challenges working with CAD data in the context of setting it up for the suggested network can be given here.

5.1 Challenges with data preprocessing

The following paragraph will set out from the optimal data to work with, followed by a short insight in the experience with the real received data.

It is concluded, that in the optimal case the CAD data would include:

- a classification of structures according to the building structure they belong to, e.g. (poly)lines of walls, windows, furnishing, lift, stairs and elevators should be classified accordingly
- annotation which provide consistently exact and complete names for relevant structures rather

than abbreviations. This in order to facilitate the identification of accessible and use-relevant structures

- door polylines which have an identifier to which roomstructure it provides access to
- with regard to stairs and escalators, it would increase the workflow to have an annotation for the direction

If the data would be given as above suggested, after georeferencing the file, the preprocessing procedure until access point generation for building structures (rooms, stairs, escalators, elevators) would include six steps(***Figure 34***).

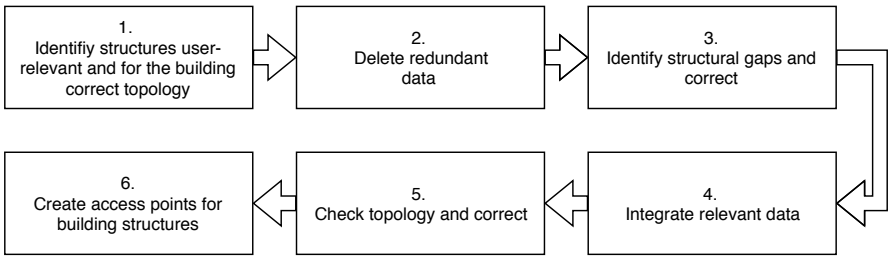


Figure 34: Suggested steps for data preparation

Step 1 Within the first step it needs to be clarified which data from the file would be relevant and accordingly selected. In the context of the indoor route planner, structures are defined as relevant when (poly)linefeatures represent internal building structures considered to add useful information for the user corresponding to the correct LOD and, furthermore, structures which are needed to create a correct topology of the building. For the route planner, walls and window structures, doors, escalator, lifts and elevators would need to be included. However, the received data differentiate data only into three classes, namely walls, doors and furnishing.

Step 2 The second step includes manual identification and deletion of structures which are considered to be redundant within the selected feature classes. The extent of time needed for this depends on the data classification specified in the DXF. For the work at hand this step needed a considerable amount of time due to the coarse classification of features which was specified by classifying linefeatures almost entirely into walls, including also redundant structures which were in the course of the work identified as for example radiators, steam outlets and parts of roof (not to interchange with room) structures. See ***Figure 35*** original data for one floorlevel before(top) and after cleaning(bottom).

Step 3 During the third step, structural gaps need to be identified. Structural gaps are here defined as gaps between structures which are not the result of editing without snapping but which occur due to the chosen kind to draw building structures. As CAD files as DXF are generally not created for routing, the approach to represent structures does not need to comply with the rules to build a topology in a GIS. Accordingly, representation needs to be corrected for routing. In the current work this step further needed extensive manual work as some structures were represented as not touching and as such gaps between representations needed to be detected and filled. ***Figure 36*** show examples of this sort of gaps.

The top image shows one of the main entrance to the airport, which is represented as a revolving door. The arrows point to the gap of the representation, which create an opening of boundary of the building. This opening needs to be filled in order to create a closed boundary of the internal of the

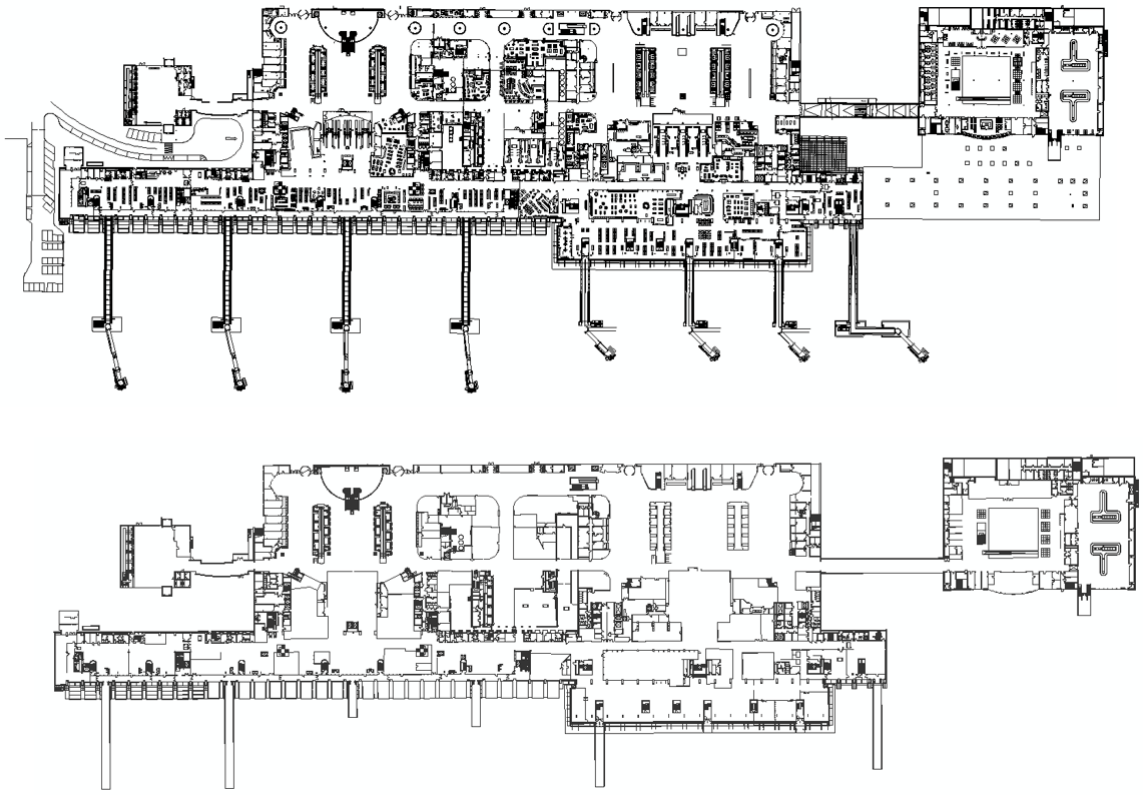


Figure 35: Top: Original data for one floorlevel of the airport. Bottom: After cleaning data of the Airport

building.
The middle image shows a type of wall, which were drawn as disconnected structures. It could be found on several places within the building. As they appeared to be small, it was difficult to identify. The bottom image shows the entrance to the security control and gives an example of the positive relationship between occurrance of structural gaps and complexity of corresponding the structural representation. Those type of gaps as presented in ***Figure 36*** were not detectable by GIS tools checking the topology, but instead needed to be manually identified and filled in order to create the boundary which the structures aim to represent. Not filling these structural gaps would cause topological conflicts.

Step 4 The fourth step includes the integration of the linefeatures into one single linefeature. This includes the generation of a linefeature from all accessible structures, as halls, corridors and the adjacent, accessible room structures. This step is needed for the later process of centerline creation.

Step 5 The fifth step refers to checking the topology on a variety of topological rules applying to linestrings as overlap, dangling nodes and gaps within lines. In the current work, also here, an extensive amount of errors was identified and needed to corrected often manually. Some examples of visible typical errors as gaps and dangling nodes are shown in ***Figure 37***.

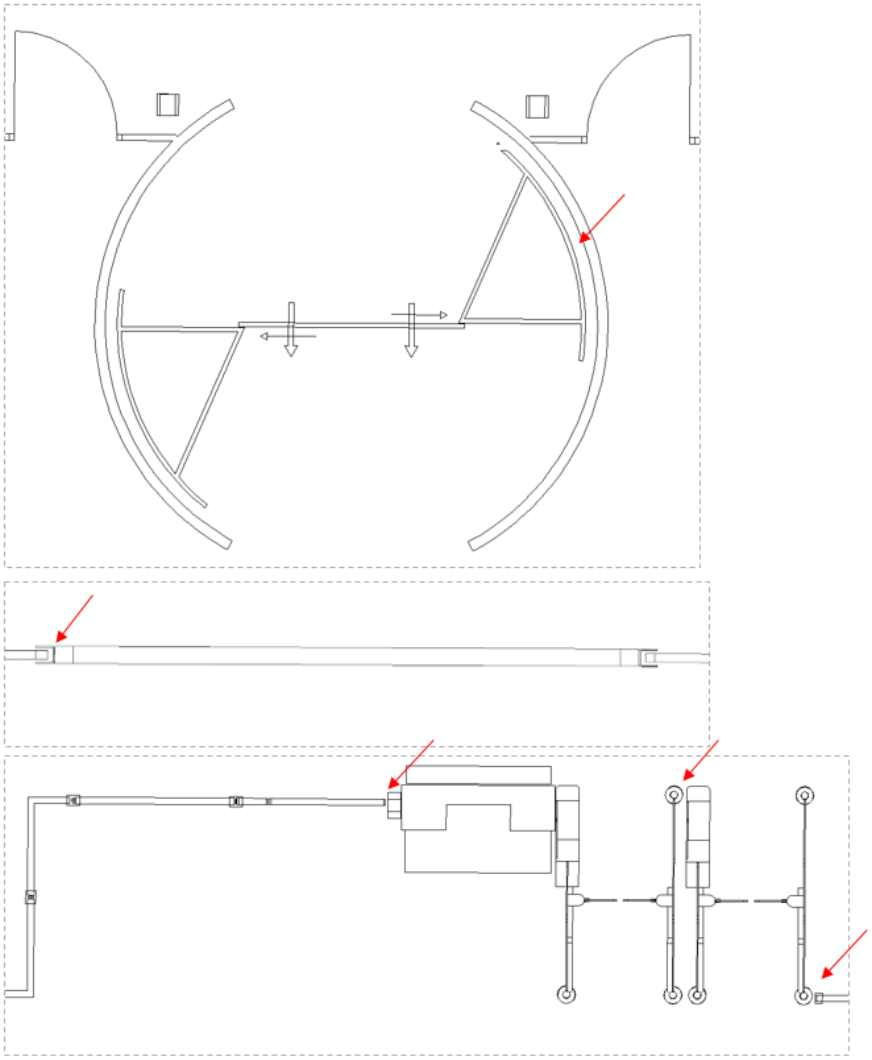


Figure 36: Examples of so-called structural gaps

However, within the current work the error rate on the unvisible topological errors of overlapping lines was as high as that the complete floorplan could not be processed in its entirety. The applied and suggested solution in such a situation was to partition the floorplan into smaller parts in order to check each part on its topology one by one and remerge the corrected parts into a complete floorplan.

Step 6 Step six refers to point generation. This step is included in the data preparation as it differs from most practical works by the data needed for generating nodes for the network for the room structures. Most works for indoor routing suggest centerpoints to represent target structures within the network which are generally created from annotations within the CAD or the centerpoints of room polygons. However, in the current work, instead, the main idea was to represent access points and as such the approach to generate point data for the routing network and corresponding needed data

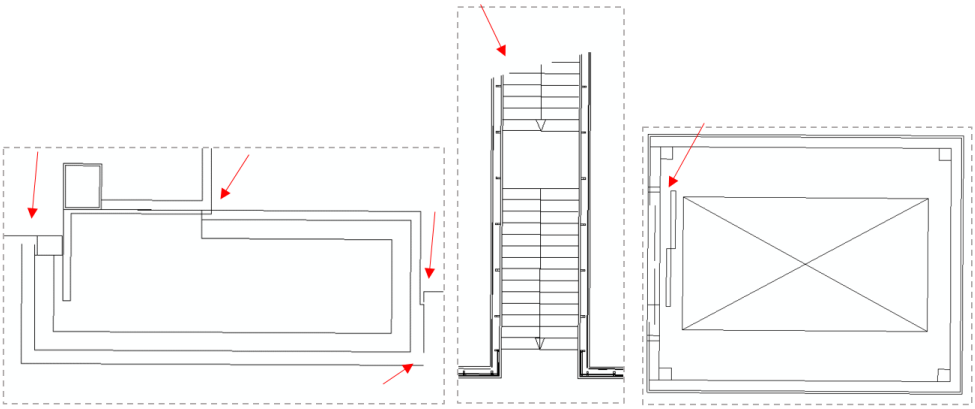


Figure 37: Examples of visible topological errors

differed from 'best practices'. To generate access points it is suggested to create point data from door structures. ***Figure 38*** shows generated access points for room structures, however, this was only received for some target structures due to time limitation as it was costly, including both, automation but also manual adjustment.

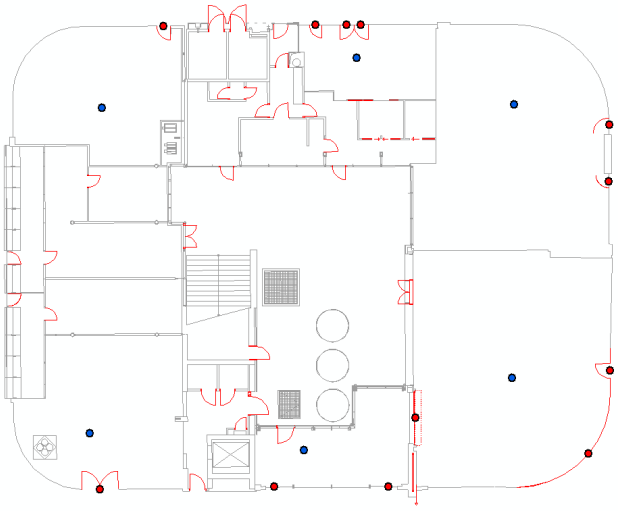


Figure 38: Generated access points for aID type 'sale'(red). Blue points are corresponding annotation point data.

It was costly as door structures were not further labelled to which structure they give access to. That doors were segmented without an identifier to which structure they belong to was comparatively minor challenge(see AppendixA for applied solution to get point data from doors). Accordingly, in order to increase the workflow of the creation of the network, doors would need to give information to which building structure they give access to. This would avoid the need to include additional steps as e.g. polygon creation of roomstructures and spatial joining the access point data. Further,for stairs and escalators, the inclusion of up and down direction as annotation placed at the relevant location of

the structure (as shown in ***Figure 39***) would further facilitate the access point generation. This, as from annotation it is an easy task to generate point data within a GIS. Annotations of directions are often included in CAD, however this was not provided in the current data and therefore the need to adjust the point data to the correct location manually.

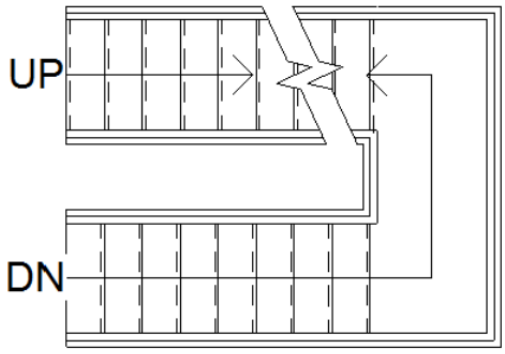


Figure 39: Annotation for directions in CAD representation of stairs. Image source: blogs.rand.com

6 Evaluation

Nowadays routing has become ubiquitous, and indoor routing even though still a small rubric becomes of increasing interest for further exploration. The current work's main contribution was a design approach for a process-focussed indoor route planner specific for the unique environment of airports. It included suggestions on how to set up a routable network as well as pathfinding algorithms and a system architecture considering the need to communicate with the airport API. Further, some challenges were addressed which may occur during the work with CAD data as well as cautious recommendations on the needed set up of the to-receive data to facilitate the preparation process for the suggested indoor routing approach. Extensions to the approach would be e.g. to consider real-time waiting times at e.g. check-in or security control which would be in accordance to the approach of (Gendreau, Ghiani, & Guerriero, 2015) on factors affecting travel times. (Gendreau et al., 2015) refers to exogenous factors which in this context, refer to external circumstances at the airport which affect the speed to take the boarding procedures, as e.g. crowded waiting lanes at check-in or at security control. Those exogenous factors need to be carefully considered in order to adequately estimate the time available for additional tasks. For this, the route application would need to get the estimated waiting times of the airports database. Waiting times may be easily integrated by just adding the expected waiting times to the calculated boarding path time. Further, literature suggests landmarks as useful addition to path descriptions ((Fellner et al., 2017)). As such it may be useful, to add landmarks into the network, which may further add information and as such facilitate the orientation within the airport building next to the access points of the space decomposition. A further addition may include the automatic generation of destinations which follow a topic. For example, a kids-route could include a visit to the airports kids playground and to the observation deck or,a 'on sale' route, which would include those shops which are currently offering special quotations. The first would be possible with static destinations, as kids playground and observation deck are static locations in the network, while the second would need dynamic information from the airport in order to select those locations currently 'on sale'. Even though the route planner was designed for airports, it may also be applicable for other indoor environments in which multiple destination need to be taken to accomplish multiple tasks in a certain sequence. In such a case the route planner would serve as a 'memory add'. In terms of cognitive

information processing, the departing process is a ‚script', which describes a scheme that has a typical sequence of activities (Wickens et al., 2004). However, schemas are knowledge structures that rely on cognitive information processes and as such can be incomplete or erroneous (Wickens et al., 2004). One concept developed to support memory (or knowledge) is the concept of ‚knowledge in the world' (Norman, 1988). It emphasizes the inclusion of memory aids in the environment in order to ensure a person does not need to rely on knowledge in the head. As a consequence, people can rely on ‚hints' from the environment instead of referring to their (eventual erroneous) knowledge. According to this, when referring to the departure situation at an airport or any other process, a path emphasizing the locations needed to visit for the process supports not only the the building of a correct cognitive map, which is the mental presentation of spatial information, which is essential for orientation (Wickens et al., 2004) but, if process relevant locations are represented in the correct order, it transports the knowledge of the sequential task of the process and therewith supports in understanding what needs to be done next. For example, buildings of authorities may be locations, in which visitor need to visit multiple offices. Often the sequence and the offices to be visited is determined by a procedure, e.g. to file for a certain application. In general, the process-focussed approach of the route planner may support the clarification of processes as it connects the tasks with concrete location needed to be visited which may seem otherwise complicated. As such, the route planner functions as a memory add as defined above. Thereby the task may not need to be within just one single building. The approach may be extended to networks which connect more than one building. Further, it may also be used as a tool to analyse and visualize e.g. administrative processes on their logical correctness regarding their sequence. For example, if some filing needs a person to visit a location A for getting a document A in order to get application B at location B, but Document A needs the provision of application B, this logical error may be visualisable with the current routing approach and more informative and expressive than an error notification in plain text.

References

Bian, W., Guo, Y., & Qiu, Q. (2014). Research on personalized indoor routing algorithm. In *Proceedings - 13th international symposium on distributed computing and applications to business, engineering and science, dcabes 2014* (pp. 275–277). doi: 10.1109/DCABES.2014.60

Churchill, A., Dada, E., de Barros, A. G., & Wirasinghe, S. C. (2008). Quantifying and validating measures of airport terminal wayfinding. *Journal of Air Transport Management*, *14*(3), 151–158. doi: 10.1016/j.jairtraman.2008.03.005

Correia, A. R., Wirasinghe, S. C., & de Barros, A. G. (2008). A global index for level of service evaluation at airport passenger terminals. *Transportation Research Part E: Logistics and Transportation Review*, *44*(4), 607–620. doi: 10.1016/j.tre.2007.05.009

de Barros, A. G., Somasundaraswaran, A. K., & Wirasinghe, S. C. (2007). Evaluation of level of service for transfer passengers at airports. *Journal of Air Transport Management*, *13*(5), 293–298. doi: 10.1016/j.jairtraman.2007.04.004

Dijkstra, E. W. (1959, December). A note on two problems in connexion with graphs. *Numer. Math.*, *1*(1), 269–271. Retrieved from `http://dx.doi.org/10.1007/BF01386390` doi: 10.1007/BF01386390

Downs, R., & Stea, D. (1973). *Image and environment: Cognitive mapping and spatial behavior.*

Dudas, P. M., Ghafourian, M., & Karimi, H. A. (2009). ONALIN: Ontology and algorithm for indoor routing. In *Proceedings - ieee international conference on mobile data management* (pp. 720–725). doi: 10.1109/MDM.2009.123

Einblicke.informationen über den flughafen stuttgart. (2017).

Farr, A. C., Kleinschmidt, T., Johnson, S., Yarlagadda, P. K. D. V., & Mengersen, K. (2014). Investigating effective wayfinding in airports: a Bayesian network approach. *Transport*, *29*(1), 90–99. Retrieved from `http://www.tandfonline.com/doi/abs/10.3846/16484142.2014.898695` doi: 10.3846/16484142.2014.898695

Farr, A. C., Kleinschmidt, T., Yarlagadda, P., & Mengersen, K. (2012). *Wayfinding: A simple concept, a complex process* (Vol. 32) (No. 6). doi: 10.1080/01441647.2012.712555

Fellner, I., Huang, H., & Gartner, G. (2017, Jun). "turn left after the wc, and use the lift to go to the 2nd floor"—generation of landmark-based route instructions for indoor navigation. *ISPRS International Journal of Geo-Information*, *6*(6), 183. Retrieved from `http://dx.doi.org/10.3390/ijgi6060183` doi: 10.3390/ijgi6060183

Fouskas, K., Giaglis, G. M., Kourouthanassis, P. E., Pateli, A. G., & Tsamakos, A. (2002). On the potential use of mobile positioning technologies in indoor environments. In *Bled econference.*

Gaudlitz, E. (2015). *Indoor navigation using wi-fi as a positioning technology.* `https://www.infsoft.com/blog-en/articleid/40/indoor-navigation-using-wifi-as-a-positioning-technology`. (Accessed: 2018-03)

Gendreau, M., Ghiani, G., & Guerriero, E. (2015). Time-dependent routing problems: A review. *Computers & Operations Research*, *64*, 189–197. Retrieved from `http://linkinghub.elsevier.com/retrieve/pii/S030505481500146X` doi: 10.1016/j.cor.2015.06.001

Gilliéron, P.-Y., Daniela Büchel, & Ivan Spassov. (2004). Indoor navigation performance analysis. *Proceedings of the 8th European Navigation Conference GNSS*, 17–19.

Goetz, M., & Zipf, A. (2011). Formal definition of a user-adaptive and length-optimal routing graph for complex indoor environments. *Geo-Spatial Information Science*, *14*(2), 119–128. doi: 10.1007/s11806-011-0474-3

Hagedorn, B., Trapp, M., Glander, T., & Döllner, J. (2009). Towards an indoor level-of-detail model for route visualization. In *Proceedings - ieee international conference on mobile data management* (pp. 692–697). doi: 10.1109/MDM.2009.118

Hart, P. E., Nilsson, N. J., & Raphael, B. (1972, December). Correction to "a formal basis for the

heuristic determination of minimum cost paths". *SIGART Bull.*(37), 28–29. Retrieved from http://doi.acm.org/10.1145/1056777.1056779 doi: 10.1145/1056777.1056779

Hölscher, C., Vrachliotis, G., & Meilinger, T. (2006). The floor strategy: wayfinding cognition in a multi-level building. In *Proceedings of 5th international space syntax symposium* (Vol. 2, pp. 823–824).

Lawton, C. A., & Kallai, J. (2002). Gender differences in wayfinding strategies and anxiety about wayfinding: A cross-cultural comparison. *Sex Roles*, *47*(9-10), 389–401. doi: 10.1023/A:1021668724970

Liu. (2017). Indoor semantic modelling for routing: The two-level routing approach for indoor navigation. *A+BE | Architecture and the Built Environment*(17), 1–254. Retrieved from https://journals.open.tudelft.nl/index.php/abe/article/view/1879 doi: 10.7480/abe.2017.17

Liu, L., & Zlatanova, S. (2011). A "Door-To-Door" Path-Finding Approach For Indoor Navigation. *Proceedings Gi4DM 2011: GeoInformation for Disaster Management, Antalya, Turkey, 3-8 May 2011*, 3–8. Retrieved from http://repository.tudelft.nl/assets/uuid:de84a1c2-6596-4888-a0dd-898886e1f48f/270048.pdf

Liu, L., & Zlatanova, S. (2012). A semantic data model for indoor navigation. In *4th acm sigspatial international workshop on indoor spatial awareness, isa 2012* (pp. 1–8). doi: 10.1145/2442616.2442618

Lorenz, B., Ohlbach, H. J., & Stoffel, E.-P. (2006). A hybrid spatial model for representing indoor environments. In J. D. Carswell & T. Tezuka (Eds.), *Web and wireless geographical information systems* (pp. 102–112). Berlin, Heidelberg: Springer Berlin Heidelberg.

Luebecke, D. (2003). *Level of detail for 3d graphics.*

Nielsen, J. (1993). *Usability Engineering* (Vol. 44) (No. 3). Retrieved from http://www.useit.com/jakob/useengbook.html doi: 10.1145/1508044.1508050

Norman, D. A. (1988). The Psychology of Everyday Things. *The Psychology of Everyday Things*, 1–104. doi: 10.2307/1423268

Raubal, M., & Egenhofer, M. J. (1998). Comparing the complexity of wayfinding tasks in built environments. *Environment and Planning B: Planning and Design*, *25*(6), 895–913. doi: 10.1068/b250895

Smolders, M., & Görtz, H. (2016). Indoor wayfinding at Amsterdam airport. *GIM International*, *30*(3), 16–19.

Stoffel, E.-P. (2009). *Hierarchical graphs as organisational principle and spatial model applied to pedestrian indoor navigation* (Unpublished doctoral dissertation). lmu.

Wickens, C. D., Gordon, S. E., & Liu, Y. (2004). *An introduction to human factors engineering.*

Worboys, M. (2011). Modeling indoor space. In *Proceedings of the 3rd acm sigspatial international workshop on indoor spatial awareness - isa '11* (pp. 1–6). Retrieved from http://dl.acm.org/citation.cfm?doid=2077357.2077358 doi: 10.1145/2077357.2077358

Yuan, W., & Schneider, M. (2010). iNav: An indoor navigation model supporting length-dependent optimal routing. In *Lecture notes in geoinformation and cartography* (pp. 299–313). doi: 10.1007/978-3-642-12326-9-16

Appendices

A ArcPy helper functions

```
# Title: Make points for doors


# Import arcpy library
import arcpy

# Set workspace. Needs to be a GDP to be able to use the function in_memory
arcpy.env.workspace = r'[...]\ArcPy.gdb'

# Define variables
lineFeatures = r'services_DoorsNormal'
mini_geometry = r'in_memory\mini_geometry'
polygonToLine = r'in_memory\polygon_to_line'
openingLine = r'in_memory\openingLine'
openingPoint = r'services_DoorsNormalPoints'

arcpy.MinimumBoundingGeometry_management(in_features=lineFeatures,
                                         out_feature_class=mini_geometry,
                                         geometry_type='CONVEX_HULL',
                                         group_option='None',
                                         group_field=None,
                                         mbg_fields_option=True)

arcpy.PolygonToLine_management(in_features=mini_geometry,
                               out_feature_class=polygonToLine,
                               neighbor_option=False)

arcpy.Erase_analysis(in_features=polygonToLine,
                     erase_features=lineFeatures,
                     out_feature_class=openingLine,
                     cluster_tolerance=None)

arcpy.FeatureToPoint_management(in_features=openingLine,
                                out_feature_class=openingPoint,
                                point_location=True)
```

```
# Title:  Defragment disassembled single features

import arcpy
arcpy.env.workspace = r'[...]\ArcPy.gdb'

lineFeatures = r'Gates'
buffer = r'in_memory\buffer'
dissolvedBuffer = r'in_memory\dissolvedBuffer'
spatialJoin = r'in_memory\spatialJoin'
dissolvedLineFeatures = r'gates_dissolvedLineFeatures'

arcpy.Buffer_analysis(in_features=lineFeatures,
                      out_feature_class=buffer,
                      buffer_distance_or_field='0.1 Meter',
                      line_side='FULL', line_end_type='ROUND',
                      dissolve_option=None,
                      dissolve_field=None,
                      method='PLANAR')

arcpy.Dissolve_management(in_features=buffer,
                          out_feature_class=dissolvedBuffer,
                          dissolve_field=None,
                          statistics_fields=None,
                          multi_part=False,
                          unsplit_lines=True)

arcpy.SpatialJoin_analysis(target_features=lineFeatures,
                           join_features=dissolvedBuffer,
                           out_feature_class=spatialJoin,
                           join_operation='JOIN_ONE_TO_MANY',
                           join_type=True,
                           field_mapping='INTERSECT',
                           match_option=None,
                           search_radius=None,
                           distance_field_name=None)

arcpy.Dissolve_management(in_features=spatialJoin,
                          out_feature_class=dissolvedLineFeatures,
                          dissolve_field=['JOIN_FID'],
                          statistics_fields=None,
                          multi_part=True,
                          unsplit_lines=False)
```

B Questionnaire Spatial Anxiety

Revised Spatial Anxiety Scale (Lawton & Kallai 2002)

How nervous are you when…
1 = not at all nervous 5 = very nervous

1) deciding which direction to walk in an unfamiliar city or town after coming out of a train/bus/metro station or parking garage?
1 2 3 4 5

2) finding my way to an appointment in an unfamiliar area of a city or town?
1 2 3 4 5

3) leaving a store that I have been to for the first time and deciding which way to turn to get to a destination?
1 2 3 4 5

4) finding my way back to a familiar area after realizing I have made a wrong turn and become lost while traveling?
1 2 3 4 5

5) finding my way in an unfamiliar shopping mall, medical center, or large building complex?
1 2 3 4 5

6) finding my way out of a complex arrangement of offices that I have visited for the first time?
1 2 3 4 5

7) trying a new route that I think will be a shortcut, without a map?
1 2 3 4 5

8) pointing in the direction of a place outside that someone wants to get to and has asked for directions, when I am in a windowless room?
1 2 3 4 5

Figure 40: Questionnaire by Lawton and Kallai (2002).