Università degli Studi dell'Aquila
Facoltà di Ingegneria

Tesi di Laurea Magistrale in Ingegneria Informatica

Route Directions using Visible Landmarks for an Indoor Navigation System based on Android device: "IndoorNav"

Relatore:
Prof. Clementini Eliseo

Laureando:
Davide Russo
Mat. 209653

Correlatore:
Prof. Zlatanova Sisi, TUDelft NL

Anno accademico 2012-2013

To my family.

ABSTRACT

For a successful wayfinding we must have information about "what is in" the environment and "where it is", a guide that is familiar with the environment and knows this information can lead your way. This guide may be another person or some electronic device, for example a software running on PDA or on smartphone device, and shall be capable of applying human principles of good wayfinding instructions. The aim of this research is to investigate how to automatically generate low-level route directions for wayfinding assistance, including visible Landmarks along route using visibility data-model approach. This process is then implemented in a software prototype(only for testing purpose), an Indoor Navigation System (IndoorNav), based on Android device, that translate routing directions into natural language textual instruction, use QRcodes for user positioning and a "Fat client" architecture with a small amount of spatial-data provided by a Web-Server using an XML file format.

KEYWORDS

Navigation, Indoor, Wayfinding, Landmarks, Location Based Systems, Android, Route directions,

ABSTRACT

Quando ci si trova in un edificio e si deve raggiungere una destinazione all'interno dello stesso bisogna essere familiari con l'ambiente per trovare autonomamente la strada corretta altrimenti é necessaria una guida che conosca l'edificio e ci indichi il percorso. La guida puó essere una persona fisica, o un sistema informatico autonomo capace di dare indicazioni applicando gli stessi principi di guida utilizzati comunemente dagli umani. Lo scopo di questa tesi é stato analizzare principi base per guidare persone all'interno di edifici a loro non noti, proporre un metodo per generare automaticamente queste indicazioni a basso livello tali da poter essere disaccoppiate dalla successiva traduzione testuale delle stesse mediante un formato intermedio di scambio dati basato su file XML, includere punti di riferimento interni all'edificio nelle istruzioni date, e generare il grafo che modella i possibili percorsi nello spazio navigabile mediante un approccio basato sulla visibilitá fra i nodi all'interno di ogni spazio navigabile (stanza, corridoio, scale, etc..). Il processo é stato implementato in un prototipo software "IndoorNav" basato su piattaforma client Android che utilizza per un sistema statico posizionamento mediante scansione di codici QR posizionati nell'edificio utilizzando la fotocamera interna al supporto stesso, ed un web service per la generazione delle direzioni e la gestione del dataset. Ció ha permesso di testare la funzionalitatica di quanto prodotto e trarre conclusioni e futuri sviluppi della tesi stessa.

KEYWORDS

Navigation, Indoor, Wayfinding, Landmarks, Location Based Systems, Android, Route directions,

Contents

Chapter 1

Introduction

In order to reach places that satisfy our needs and wants in indoor environment, we need to know where to go and how to get there. Assuming that we are unfamiliar with an environment, we can analyze many common cases of study. Assume that you are in the University campus of TUdelft (NL), at the Main entrance(North) of Architecture building(Figure 1.1) and you have a meeting in the "Room V" but you are unfamiliar with that building and you don't know how to get there.

Figure 1.1: *Architecture Building of TUDelft NL*

In many other examples we can came across the same scenario:

- **Hospitals**: you have to move across corridors and floors to reach the new doctor's room, or your daughter in the neonatal unit, but in this days there are temporary work in progress and the route to follow isn't the common one that you usually follow (Figure 1.2);

Figure 1.2: *A corridor of a Hospital*

- **Airports**: you're just arrived at the departure hall of "Fiumicino Airport" in Rome(IT), your assigned Gate N134 is somewhere in the Terminal C (International flights) but you are a bit confused because you can't see any useful sign about the right way to follow (Figure 1.3);

Figure 1.3: *Fiumicino Airport hall - Terminal C*

- **Train/subway Station**: you're in the London's Waterloo Station, and you

need to reach from the subway platform of "Jubilee" line, the closest exit to the "London Eye"; you know that following a wrong direction, the exit will be in the opposite side of your destination(Figure 1.4);

Figure 1.4: *London's Waterloo train station*

- **Shopping Mall**: arrived in the shopping Mall of America (Minnesota USA) you want to buy some presents for Christmas in the "Lego store", but inside the Mall there are more than 530 stores are arranged in three levels, and you get lost in a while (Figure 1.5);

Figure 1.5: *Inside the Mall of America - Minnesota USA*

- **Car Park**: in the same Mall of America, introduced before, the building provide 12'000 parking spaces divided into different floor levels, and after a day spent in the Mall you want to go home, but your car is one of the thousands cars in the park, how to find it? (Figure 1.6);

Figure 1.6: *Multi level car park*

- **Museums**: just entered in the "National Army Museum" located in the Chelsea district of London, the easiest way to visit it within an hour, seeing only the most famous paintings and objects exposed, is to follow a list of selected items, provided by the museum itself or someone else; but how to follow this guide without knowledge about the environment? You can easily get lost and lose a lot of time(Figure 1.7).

Figure 1.7: *National army museum - London*

All these examples highlight the need to understand at first what **navigation** means; according to Montello [Mon05], it can be defined as the "*coordinated and goal-directed movement of one's self (one's body) through the environment*". As Montello proposed, it may be conceptualized as consisting of two components: *wayfinding*, that involves localization of ourselves and the destination, and choosing route to take; *locomotion*, that refers to the movement of the body in the direction we intend, avoiding obstacles and barriers. Some researchers use the term of "navigation" more or less synonymously with "*wayfinding*", but according to Montello's definition [Mon05]: *wayfinding* is only "the goal-directed and planned travel of one's self around an environment in an efficient way". It's one of the primitive everyday problems humans encounter and it has become a major research direction in many areas (from this point we use the term "Navigation" synonymously with wayfinding, as common use in most of the literature).

Wayfinding research can be organized in two broad areas [RK04]: how humans and other agents actually find their ways [All99]; and how support humans in the activity of *wayfinding*. Regarding the first area of research, Redish [Red99] resumed five different strategies that a person can take to find a desired goal:

- *Random Navigation*: the individual has no information about the location and he is forced to search randomly;
- *Taxon Navigation*: the individual can find a visible cue through which he can reach the arrival point;
- *Praxic Navigation*: The individual can execute a fixed motor program (e.g. walk for 30 m than turn right);
- *Route Navigation*: Taxon and praxic navigation are special cases of Route navigation, it can be understood as the sequencing of taxon and praxis strategies. The individual, in complex maze, plans a sequence of sub-goals, as many early navigation tasks, and learn to associate a direction to each one, without knowledge of the rest of the environment.
- *Locale Navigation*: the individual has a mental representation of the surroundings, by using cognitive maps (mental representation of the surroundings) that are the model of the world as perceived from the individual, formed while walking through the environment. Using them he can plan a path between any location within the area.

Richter proposed a framework to evaluate strategies of "*how humans actually find their ways*" by using these dimensions [RK04]: **planning a route** and **following actually a route**.

Analyzing Redish strategies with Richter's framework, "Locale Navigation" is the only one that carries out both the tasks, but in our first assumption of this Thesis, we consider the case in which we are unfamiliar with the environment, so considering Locale Navigation, individuals can't plan a route, but only follow a route. We focus on **Route Navigation** and how to learn the route, we have two ways: repeat the same route many times and memorize all the directions associated to each sequence, as part of the main task, or use an external spatial knowledge that supports individuals on giving the directions needed. Following this second option, we enter in the second research area introduced before: "how to support humans in the activity of wayfinding". The act of support someone by giving directions is defined Guidance, the instructions given are defined **Route Direction** [Ric08]. In this thesis we deal with the *wayfinding* component of navigation task and we will not consider *locomotion* issues; we'll investigate about provide direction to follow for Route Navigation task.

1.1 Guidance systems: Definition & Classification

The previously introduced scenarios and the needs of guidance can be solved automatically by a navigation service, which should provide a real-time route guide tailored for the user's needs, without asking directions to someone inside the building. Systems that automatically and autonomously help user on wayfinding are called **Location-Based Guidance Systems** , they are one of the most important applications of LBS [1], and with the gradual maturating of ubiquitous computing and the rapid advances in mobile devices and wireless communication they have gained increasing interest also as an important application of ubiquitous comput-

[1]LBS: Location Based System

ing [HG10]. Focusing on interior environment, we define indoor-LBGS[2] as LBGS[3] with the aim of human guidance and the capabilities of Positioning in Indoor spaces. There are plenty of benefits on using iLBGS, we can **choose different pedestrian type**: normal person, disabled person, wheelchair, robot, and adapt the route to the user needs; **use an indoor and outdoor seamless navigation system** for drivers inside car park, person who travel between mixed indoor and outdoor paths (e.g. in a University campus, between two different buildings); use iLBGS for **emergency situation**, showing the closest exit, o to lead the firefighters to person suffered from accident.

In order to give an answer to all the challenges about indoor navigation, many indoor navigation systems have been proposed and some recent surveys, focusing on them, have tried to make a classification using different dimensions. Huang & Gartner [HG10] surveyed iLBGS using an evaluation framework (Figure 1.8) using the dimensions of Indoor Positioning, route communication, context-aware adaption, other features; Fallah et Al. [FABF13] provided a comprehensive overview of existing indoor navigation systems and analyzed the different techniques used for: locating the user, planning the path, representing the environment, interacting with the user. Merging the two, we can suggest an Improved Evaluation-Classification Framework for iLBGS with these dimensions (Figure 1.9):

- Localization;
- Environment Representation;
- Planning the route;
- Interaction.

Using this framework we can analyze related works involving Guidance Systems and introduce the aim of this thesis.

[2]iLBGS: indoor-Location Based Guidance System
[3]LBGS: Location Based Guidance System

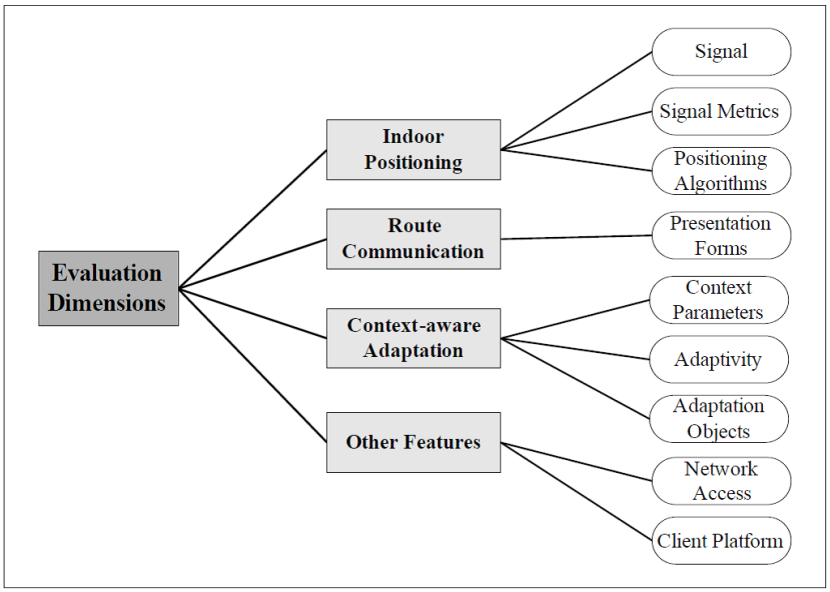


Figure 1.8: *Evaluation framework for Indoor Navigation Systems [HG10]*

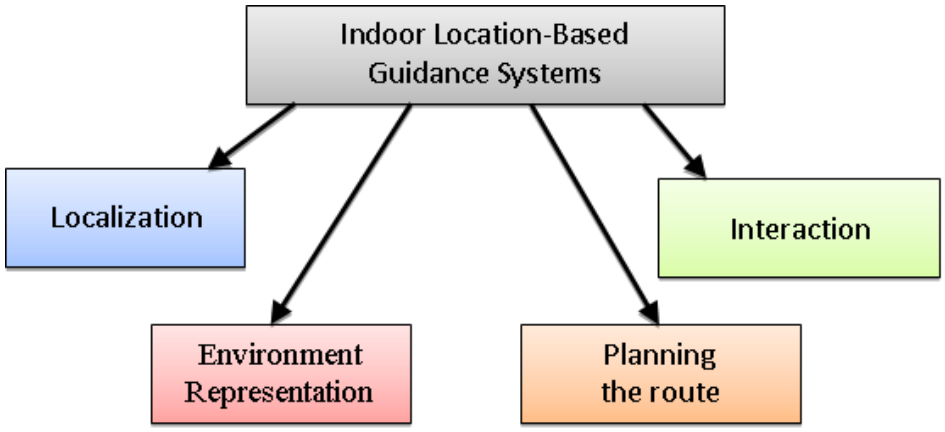


Figure 1.9: *Improved Evaluation & Classification Framework for Indoor Navigation Systems*

1.2 Guidance challenges

Indoor navigation, instead of outdoor (often referred to car Navigation), has more and new challenges to deal with; in this section we analyze the differences between them. Outdoor navigation guidance is a field of research well covered by plenty of papers and solution to all type of problems, and car navigation systems have become a mass product. Stoffel has resumed these key differences [Sto09]:

- *Shape diversity*: the network structure of roads is regular and clearly defined, road segments are linear and every junction of 3 or more segments there is a decision point; also large metropolis show a grid pattern, especially modern districts. In contrast, a systematic treatment of indoor environments is difficult, architects have more freedom in designing a building, rooms can vary in size and shape depending on their function. Particularly large rooms can be unique for their shape and multitude of connections.
- *Degrees of freedom in movement*: Vehicles are mostly bounded to lanes/rails, and drivers must respect driving rules: it's not allowed to turn just anywhere, reversing direction everywhere, stopping along the road everywhere, etc... Pedestrian motion is less restricted than vehicles, in large halls they can move freely. In Figure 1.10 is shown the difference between outdoor vs indoor freedom in movement.
- *Granularity*: the speed between vehicles and humans is different while moving, it involves a different level of details on the perspective of the surrounding space: it means that features of a building must be modelled at a higher granularity.
- *Network type*: Road network can be modelled by one-dimensional data structure, like a graph; on the other hand, it is much more difficult to extract path structure from building spaces, especially in large rooms in which we can only define a region of free moving. The transition in large spaces seems rather fuzzy!

Despite all these difficulties, everyday we find route to our destination point, also in unfamiliar buildings. How do we give/understand/follow directions, sometimes in very complex shapes? In human indoor route directions the most common environmental features used are *pathways*, *landmarks* and *choice points* [Lyn60]. In other words, we almost never find only numerical references to distances or turning angles

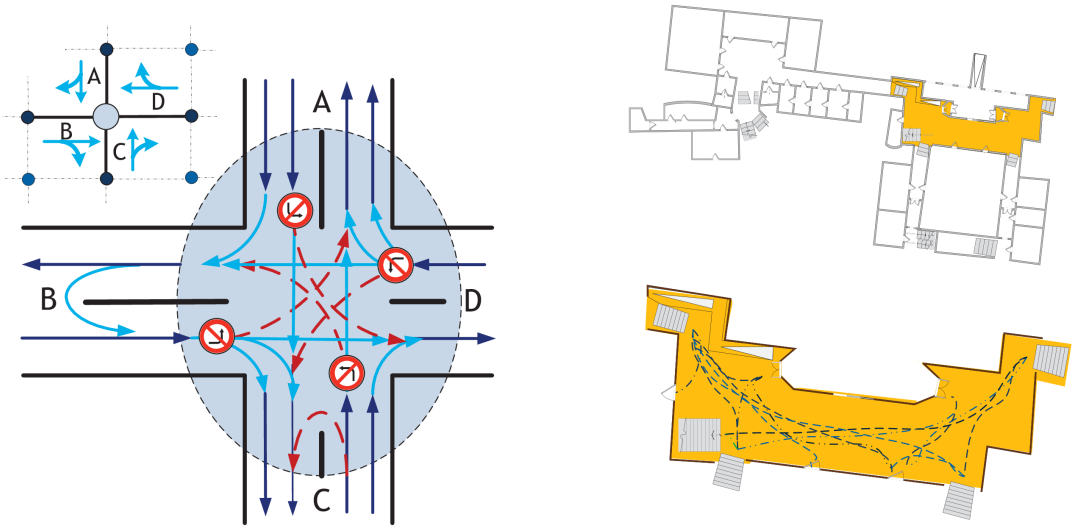


Figure 1.10: *Motion in Road Network vs Pedestrian in Indoor Environment*

(*pathways* description), instead people use actions anchored to each multiple-choice ambiguous location (decision points) or use **landmarks** to provide confirmation that the right track is still being followed(landmarks along route segments) [Ric13].

Landmarks are defined as "prominent features in the environment that are unique or contrast with their neighbourhood" [SW75]; and as "natural, built, or culturally shaped features that stand out from their environment" [Gol99] (Figure 1.11). A landmarks may be described also as an "environmental feature that can function as a point of reference that serve as sub-goals that keep the traveler connected to both the point of origin and the destination along a specified path of movement" [Lyn60]. Landmarks lead to shorter learning times, better recall in route description tasks, and better response in wayfinding tasks [RMT04]. Daniel and Denis [DD98] demonstrated that only about 15% of human direction elements are not related to landmarks. Integrating references to landmarks in wayfinding assistance is essential for generating cognitively ergonomic route directions. Although they are widely used in human wayfinding and communication about route, today's spatial information systems rarely make reference to them. The main reason is the lack of available data about landmarks or even agreed characteristics defining a landmark. Sorrows and Hirtle [SH99] proposed some basic properties for landmarks identification and classification (these categories are not mutually exclusive): Visual

Figure 1.11: *Examples of remarkable objects (landmarks) along the route*

landmarks are distinguished by their visual peculiarities; Semantic landmarks are distinguished by their use or meaning; structural landmarks are distinguished by their location in the structure of the environment.

Person traveling through an environment must be oriented in order to reach successfully the next decision point or the destination itself and maintain **orientation** using a combination of two processes [MS06]:

- **Landmark based orientation**: involves recognizing specific features in the environment, it requires internal or external memory;
- **Dead-reckoning orientation**: involves keeping track of components of locomotion, such as velocity, acceleration, travel duration.

An important limitation of dead reckoning is that it accumulates error if it is not periodically corrected via landmark-based process, because it provides any new position relative to previous one sequentially, a minimum error can be increased while using this process for long routes. It's clear that landmarks play a decisive role in human navigation, their importance is proved by a great amount of studies [RW02] [LHM99] [Ric07] [DWR10].

We can now conclude this section with a resume of the key aspects of route directions discussed before, resumed by Klippel et Al. [KRH09] in the Figure 1.12 below.

Cognitively ergonomic route directions

- Are qualitative, not quantitative,
- allow for different levels of granularity and organize spatial knowledge hierarchically,
- reflect cognitive conceptualizations of directions at decision points,
- chunk route direction elements into larger units to reduce cognitive load,
- use landmarks to:
  - disambiguate spatial situations,
  - anchor turning actions,
  - and to confirm that the right actions have been taken,
- present information in multimodal communication systems allowing for an interplay of language and graphics, but respecting for the underlying conceptual structure,
- allow for an adaptation to the users familiarity with an environment, as well as personal styles and different languages.

Figure 1.12: *Cognitive ergonomics Route directions [KRH09]*

1.3 Motivation and Goals

Indoor navigation isn't a new concept, but the common need to visit unknown buildings to find one's way increased the request of autonomous systems that can support human wayfinding, both in public and private business sector. It increased the scientific community's interest to study and make publications and prototypes for guidance support. Since no commercial and comprehensive system for pedestrian indoor navigation yet exists, the challenged of indoor navigation system for human guidance will be taking into account in this thesis with these requirements:

1. **Open spaces modeling**: closed spaces such as narrow and long corridors, stairs, elevators and other linear features represent a network with an implicit direction to move in, having less difficulty to deal with; instead open spaces

have a vast number of different ways for crossing it. We want an intelligent solution that can split and manage big open space through a network model that reflects the real human free movement with the minimum deviation to the real routes.

2. **Route planner**: we want a route planner that computes paths between any starting and ending point inside the environment and estimates distances as close as possible to the real paths.
3. **Positioning system**: we want a navigation system that hasn't additional infrastructural costs to calculate position and that can also work in emergency cases without power inside the building, only with an external network data connection (e.g. GPRS, UMTS, EDGE, etc).
4. **Humans orientation**: we want to have an egocentric orientation system (not allocentric) that can compute users direction while moving and capable to give instruction based on individual current orientation.
5. **Landmarks usage**: as explained in the previous sections, usage of landmark is widely performed in common human route directions, we want to integrate guidance with landmarks.
6. **Real-world directions**: we want to use directions as close as possible to the real-word spoken guidance often used by people.
7. **Hierarchical instruction**: we want to have a multiple level of details in route instructions, that exploit the advantage of splitting long path into smallest sequence giving higher level instruction and, if requested, other levels of details.
8. **Directions generation adaptability**: we want a dynamic route directions generation that reflects the changes of an editable data model. This requirement can answer to the request of emergency situations and work in progress cases, where the data datamodel must be changed and adapted to the current needs and availability and route directions must take into account this real time changes (e.g. a corridor is unavailable due to fire, a room is closed due to work in progress for the next 3 days, an emergency exit is unavailable for common use but in emergency case could be take into account). We don't want a static route direction system, using recorded videos, or stored directions that aren't flexible to daily scenarios.
9. **Internationalization**: we want to have language independent route genera-

tion system that can translate instructions in more than one language without changing the route generation process.

10. **Personal digital assistance**: we want a common platform for guidance with low costs for users and system management, widely spread and easy to use.

1.4 Thesis organization

In order to satisfy all these requirements, this thesis is structured as follows: in the Chapter 2 we will introduce a literature review that clarifies state-of-art of indoor navigation systems, and survey the most useful prototype proposed. After that in Chapter 3 we will explain our approach method explaining step by step our solution to all the challenges illustrated in this chapter. Then in chapter 4 we analyze salient parts of the algorithms proposed and prototype implementation. Finally in Chapter 5 we will resume test and evaluate them and in Chapter 6 we draw the conclusions and future work.

1.5 Notations and conventions

Formatting conventions:

- **Bold** and *italic* are used for emphasis and to signify the first use of a term.
- `Code` is used in code snippets and generally for anything that would be typed literally when programming, including keywords, constants, method names and variables, class names and interface names.
- The present report is divided in chapters. Chapters are broken down into sections. Where necessary, sections are further broken down into subsections, and subsections may contain some paragraphs.
- Figures, tables and listings are numbered inside a chapter. For example, a reference to figure j of chapter i will be noted Figure i.j .
- As far as gender is concerned, we systematically select the masculine when possible.
- As far as the author of this document is concerned, the first-person plural pronoun is used (royal we).

Chapter 2

Related works

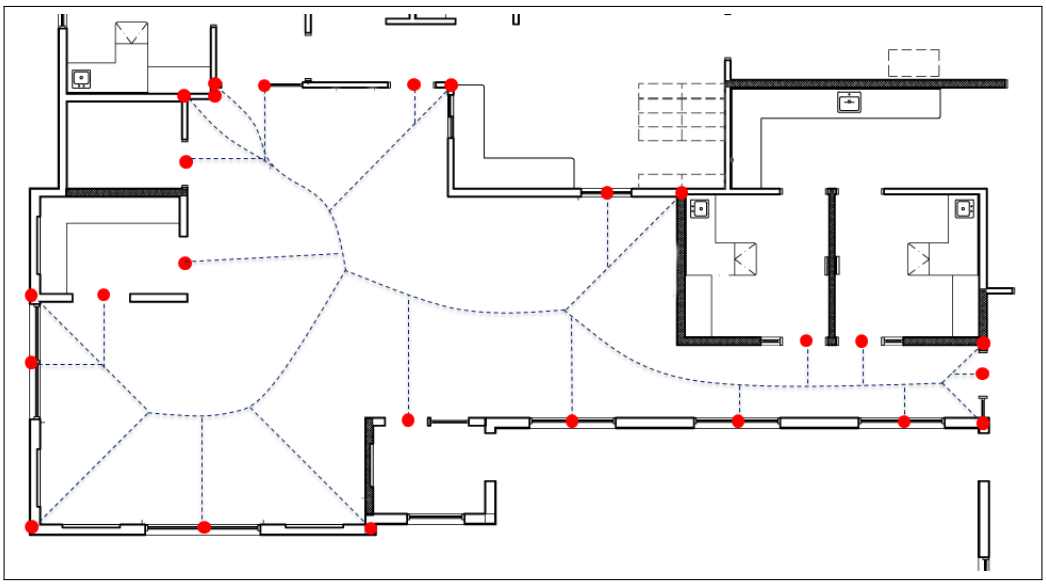
In this chapter we introduce a literature review: we analyze in deep each dimension proposed in the evaluation framework introduced in the previous chapter. At first we start from the Environment Representation with comparison between MAT and visibility approach, then we move to Localization Techniques focusing on QRcodes, then we explain planning techniques. Finally we introduce the Interaction field divided into how to generate route directions and how to communicate them to the users. Then we make a survey of the most influential and useful Navigation Systems for the aim of this thesis.

2.1 Environment Representation

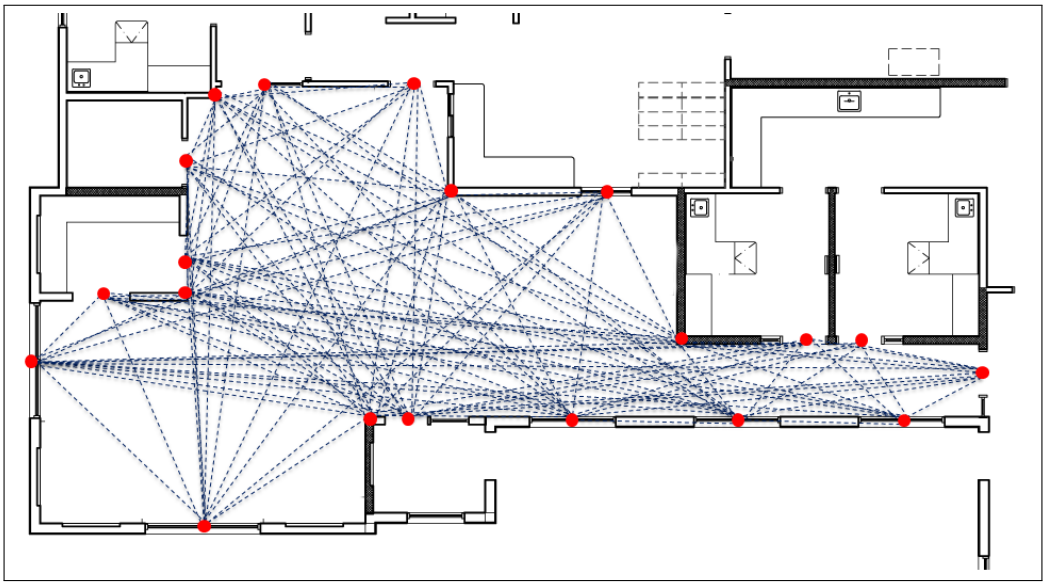
Human navigation systems require storing and retrieving different types of information. The stored information can be used for localization, path planning, generating directions, and providing location information. Depending on the approach employed by the system, this information may include floor plans, the location and description of objects in the indoor environment, locations of identifier tags or data collected using sensors [FABF13]. Historically, routing is based on graphs since road networks can easily be described as sets of nodes and edges. One popular approach to solve this problem is to focus on the topology of rooms and build a graph to represent this topology [Ste13]. Therefore, a very important phase in the environment representation is the simplification of the building structure, extract-

ing the geometrical data model to support the routing algorithm. There are two different approaches used for generate a Geometric Network Model of a building (Figure 2.1):

- Medial Axis Transform based [B+67];
- Visibility based [LPW79].



(a)



(b)

Figure 2.1: *Comparison of MAT based approach(top) and a visibility approach(bottom).*[1]

The first one is the most used in outdoor guidance systems, and assumes a primary role also in indoor guidance systems. Unlike car outdoor navigation, pedestrian navigation need more accuracy on network modeling: one has to observe that pedestrians usually are not constrained to straight paths like cars but can move freely in huge places or big halls. A static graph design, which represents every possible pedestrian movement, would induce the number of nodes and edges to grow to infinity, and is therefore infeasible. In big spaces, such as big halls or complex shape corridors, a medial axis model may not follow the natural routing directions that users follow moving from a door to another door. One of the major goals of doors visibility approach is to provide routing that adapts better to the walking behavior of pedestrians [LZ11].

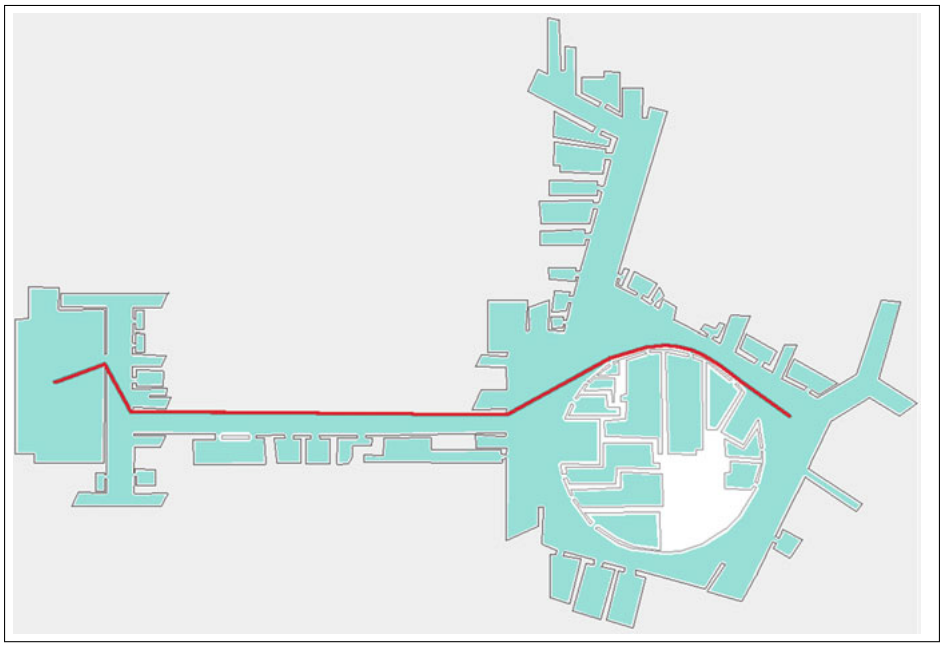


Figure 2.2: *Floor plan of a mall: the navigable space is described by the cyan polygon, it's a unique large space without doors and openings. The red line is an example shortest path computed between two arbitrary points.*

Another common problem affecting both the models is how to manage large spaces with strange shapes, consider the image in the Figure 2.2. It's a large hall of a mall, without doors or openings that could split the space into semantic sub-spaces. A subdivision is needed, for example, to give different semantics to the narrow

corridor that connects two different halls on the left and right side of the figure. Other examples are all the rooms around the hall in the right side, that haven't a physical opening but only a virtual entrance. Chown et al. introduced the concept of *Gateways* described as the place where ''*a visual narrowing is followed by a visual opening. A gateway occurs where there is at least a partial visual separation between two neighboring areas and the gateway itself is a visual opening to a previously obscured area. At such a place, one has the option of entering the new area or staying in the previous area. A gateway is often an entrance/exit to/from a larger space*" [CKK95]. Gateways, or also called "virtual-doors" are also used to compute the direction of the individual while passing through them, his direction is usually perpendicular to the virtual-door. Kuipers et al. [KMB$^+$04] produced the following image-sequence (Figure 2.3) that explain the idea of identifying gateways starting from MAT based graph (a) ending with gateways and outgoing-ingoing directions of each one (it isn't an automated algorithm, and nowadays no-one has proposed a solution yet).

In order to introduce the third dimension in buildings representation, Stahl introduced the concept of 2 1/2 dimensional model [Sta08]. Rooms are represented as polygons, with a counterclockwise ordered sequence of vertex, each vertex is represented through Cartesian coordinates (x,y,z). The z value represents the room's floor height above the ground level. Most of the room haven't slope, so we consider rooms of the same floor with the same fixed height. Spaces such as stairs or ramps are the only ones that have polygons defined by vertex with different height associated to different floors. Resuming, the z value isn't a continuous set, is a discretized set with heights of floors as elements: it's why Stahl called this model 2 dimensions (x,y) plus 1/2 dimension (z). An example is shown in the next Figure 1.4 .
The introduction of the third dimension representation is the starting point of Stoffel et Al. "Hierarchical model for pedestrian indoor navigation" [SSO08]. The key point of this research is that some of the spaces are regions with an implicit decision point, they are pivot spaces (such as corridor, halls, stairs, elevators) with a lot of connections to other spaces surrounding them. They have proposed an algorithm that produces a multilevel hierarchical graph from a standard connectivity graph of a building as shown in the Figure 2.5.

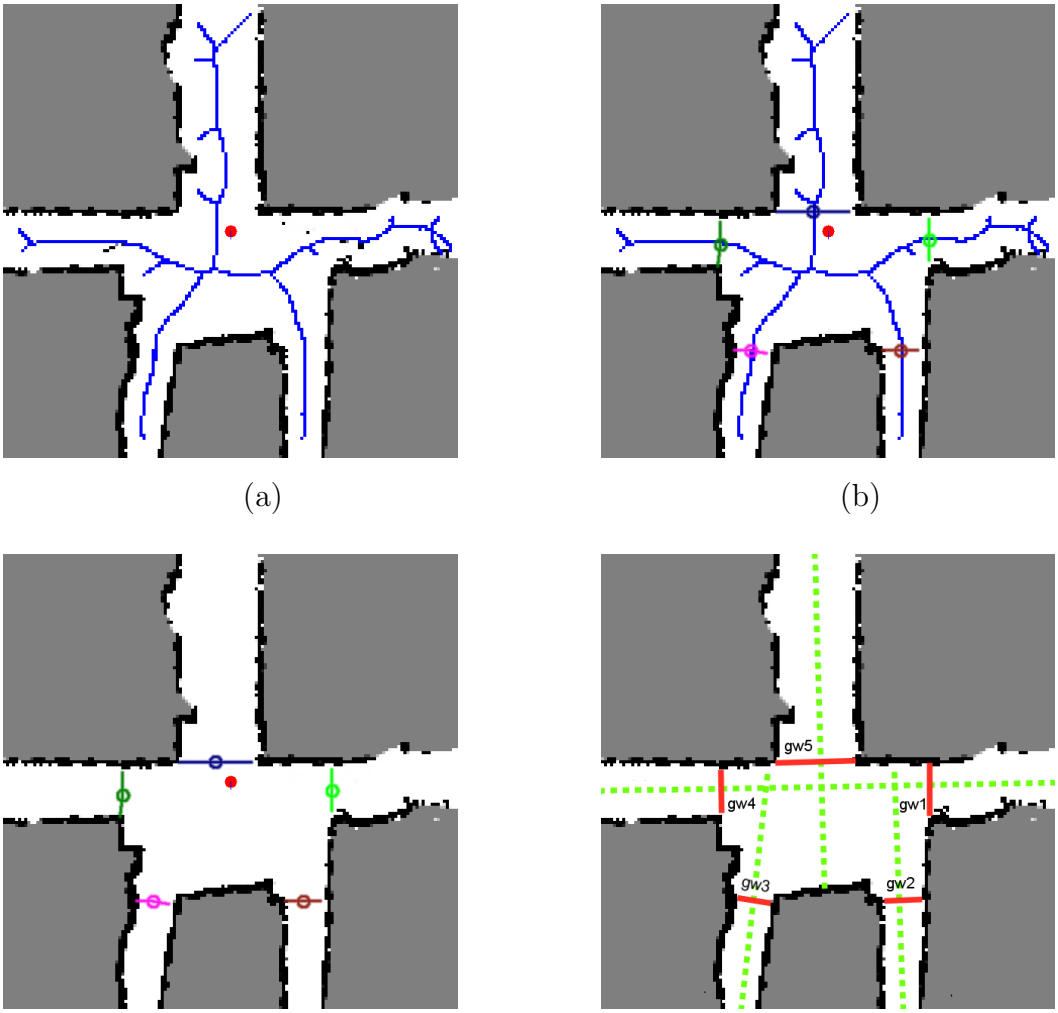


Figure 2.3: *Identifying Gateways and Local Topology. (a) To find gateways in corridor environments, the Kuipers algorithm computes the medial axis of the occupancy grid free space. (b) The maximum of the medial axis graph is found (where the distance of obstacles from the graph is maximal) and each edge is traversed, looking for constrictions (where the distance between the graph edge and obstacles is a local minimum). (c) The final gateways are drawn as lines connecting the graph edge minima (circle) with the closest obstacles. (d) Given the gateways, the directions are identified. [KMB+04]*

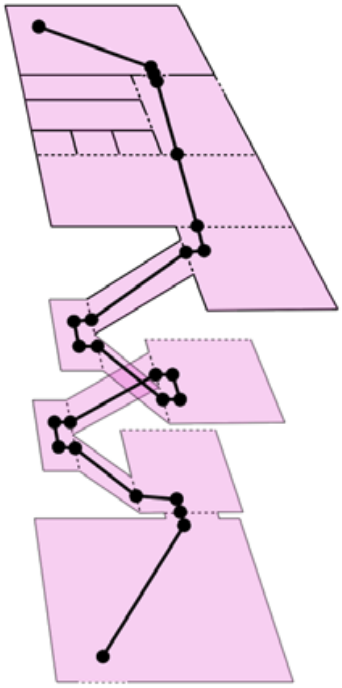


Figure 2.4: *The 2.5 dimensional data model and a route between two points. [Sta08]*

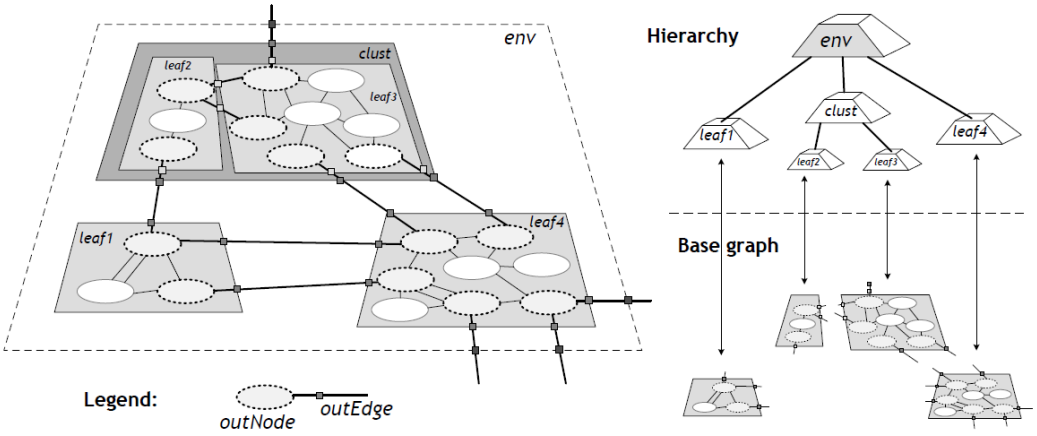


Figure 2.5: *Hierarchical Graphs as Conceptual Model [SSO08]*

Y.A.MA.MO.TO. Framework

Stahl et Al. [Sta08] developed a framework, Yet Another MAp MOdelling TOolkit (Y.A.MA.MO.TO.) that is a CAD modeling toolkit that offers an easy modeling interface for editing multi-level buildings in 3D space including the furnishing and landmark objects. The visibility approach used reflects natural human's movement inside a building, they added a buffer (Figure 2.6) along walls in order to avoid paths with zero distance from the wall that is unnatural. It provides a route planning component, based on visibility data model approach as shown in the next Figure 2.7.

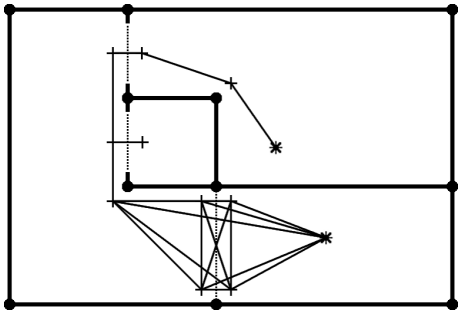


Figure 2.6: *Y.A.MA.MO.TO.: example of buffer along walls and computed paths*

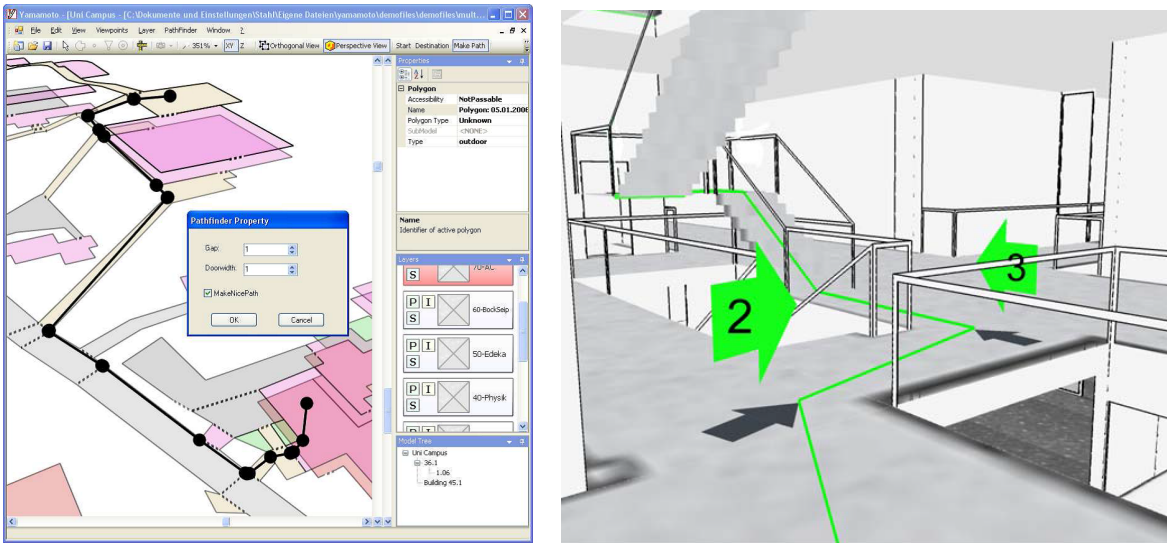


Figure 2.7: *Y.A.MA.MO.TO.: path generated example(on top left), buffers added to wall to avoid unnatural paths closest to walls; Video screenshoot during route direction(bottom).*

2.2 Localization

Most of the outdoor navigation systems employ GPS for positioning. Unfortunately, GPS can only be used outside of buildings because the employed radio signals cannot penetrate solid walls. To obtain an exact and dynamic positioning in indoor environment, additional installations (e.g., WLAN, sensor networks) are required [HG10]. There exist numerous different positioning approaches that vary greatly in terms of accuracy, cost and technology. We can subdivide location methods into four different major techniques [FABF13]:

- Direct sensing: determine the location of the user through the sensing of identifiers or tags, which have been installed in the environment. Sensing one tag is sufficient for determining the location of the user and tag reader can be easily embedded in hand held devices. The user's orientation can be determined from relative changes in location from subsequent reads of tags [WH05]. Different technologies have been identified that are being used for the tags:
  - RFID: passive [WH05] or active [DYZJ07];
  - Infrared short-range transmitters [BKW02];
  - Bar Codes such as QRCodes[2] [CTW08].
- Triangulation: use the location of at least three known points to determine the users' location, this techniques have a lower precision than GPS due to multipath reflection problems of walls and signal attenuation between floors. About technologies involved, several key aspects have to be considered [KH06]:
  - Signal type: Infrared, ultrasonic, WLAN [TAKH06], Bluetooth, ZigBee, UWB;
  - Signal metrics: Cell of Origin, Received Signal Strength, Angle of Arrival, Time of Arrival, and Time Difference of Arrival.
- Pattern recognition: use data from one or more sensors carried or worn by the user and compare this perceived data with set of prior collected raw sensor data that has been coupled with an environment map. This map of sensor data can be created by sampling at different locations or by creating it manually. Different sensing techniques used:
  - Computer vision: using embedded camera [RHM04];

[2]QRCode: Quick Response Code

  - Signal distribution or fingerprinting [APBC08] [CBM09];
- Dead-reckoning: estimate a user's location based on a previously estimated or known position. While the user is moving, the dead-reckoning system estimates the user's location through the aggregation of odometry readings: a combination of sensors such as accelerometers, magnetometers, compasses, and gyroscopes [Ret04].

Currently a lot of indoor navigation systems employ radio signal (WiFi, Bluetooth, RFID, etc.) for positioning using triangulation or pattern recognition technique, but they may suffer from the problem of signal impairments, such as Radio Frequency interference and multipath propagation. Also dead-Reckoning technique may suffers from positioning errors, because even if the positioning error is less than a feet, in two near room, divided only by thin walls or glasses, positioning couldn't be so easy and accurate. Mixing technologies in a Hybrid iLBGS could be the solution to improve accuracy of localization and decrease costs using preinstalled infrastructures. Direct sensing has different problems from the others technique, because the positioning is computed only near tags, but between them there aren't information about dynamic positioning of the user when he is moving: in this case the challenge is to provide complete wayfinding instructions, in proximity of each identifier/tag, on how to reach the next one, and how to detect wrong directions during navigation between tags.

2.2.1 Localization using QR-Codes

Recently, the 2D barcodes have been extensively developed to encode large volume of information in many applications. Among the 2D barcode systems, the QR Code is widely used because of its high reading speed, high accuracy, and superior functionalities. As a result, QR Codes are widely used not only in its original country, Japan, but also in many other countries now. QR Code, initial of Quick Response Code, is a 2D matrix barcodes developed by Denso Wave Corporation in 1994. As Denso Wave pointed out in its website [3] QR Code is capable of handling many types of data, such as numeric and alphabetic characters. It can encode up to 7,089 numeric characters or 4,296 alphanumeric characters in one pattern. In addition, unlike many traditional 2D bar codes that needs to be decoded by a

[3] http://www.denso-wave.com/en/

specific scanner, QR Code can be decoded by a small program in a cell phone or a personal computer with built-in camera. [KTC09]

In a research project by Frederic Aebi [Aeb12], has been developed INav (autonomous Indoor NAVigation system on Android): a software prototype that includes a library for scanning QR codes on Android. There exist various open source software suites for reading QR codes. They all basically use the same decoding principle, namely to use the built-in camera of a mobile device to scan and decode a QR code. One of these software suites, used in the INav software, is called ZBar [4]: it's a free library, easy to import in an Android application. In a typical scenario, a user enters the building and points with the mobile's camera to any of the QR-Codes available in every door. The following Figure 2.8 shows the idea of Code scanning.

Figure 2.8: *A user scanning a QR-Code as starting position*

[4]http://zbar.sourceforge.net/index.html

2.3 Planning the route

Path planning is an important part of the navigation, which can affect the overall performance of the system. Path planning can by defined as: starting from the Geometric Network Model's graph of the building, with oriented and weighted edges, compute the path from starting to end node in such a way to maximize the usability and success rate while minimizing the chance of the user getting lost. A smarter path planning technique needs to consider users' requirements and customize the path accordingly. Shortest path or shortest travel time is desirable for majority of users and most of the current navigation systems use the shortest path algorithms [FABF13], Dijkstra's algorithm is the most used but not the most efficient. Steur developed a custom A* algorithm, introducing a new heuristic based on different weight given to the vertical movement (Δ z) instead of horizontal (Δ x,Δ y), which reduce 77% run-time against standard Dijkstra's one, and 22% against A* standard [Ste13] (Figure 2.9).

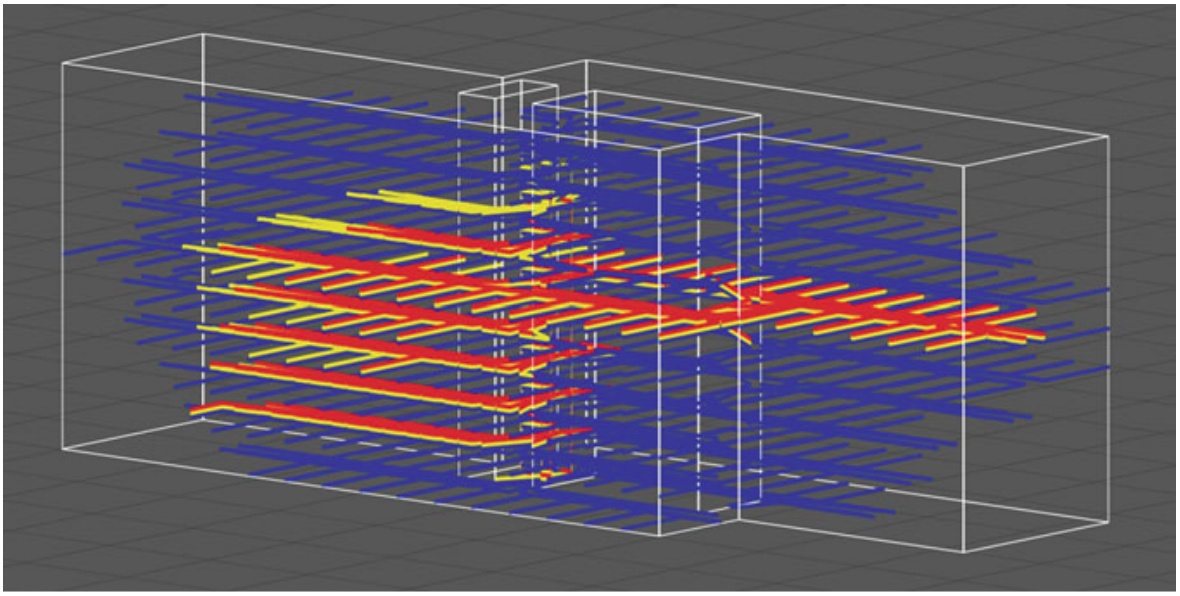


Figure 2.9: *Three sets of visited nodes: Dijkstras algorithm (blue), A* using the standard heuristic he (yellow) and A* using the proposed heuristic h3D building specialized for routing in 3D buildings. We are visualizing the edges leading to the visited nodes instead of the nodes itself because of clearer visibility [Ste13]*

A planning technique might minimize the cognitive load [BKW02] considering the complexity of a path and the directions provided, to help elderly or individuals with cognitive problems. For individuals with visual impairments a path that goes along walls reduces the chance of the user getting lost and a path which avoids low ceilings is much safer [FABF13]. Accessibility of the path might be considered when planning the path for wheelchair users or elderly such that stairs are avoided and the slope of each path is considered [PJS+96]: a flexible system might let the user to set preferences based on their needs [TAKH06]. Most automated navigation systems rely on computing the solution to the shortest path problem, and not the problem of finding the "simplest" path. When need to compare more than one path, the key points are:

- compute different shortest paths comparable using different metrics;
- identify a weighting function that reflect the amount or complexity of information required giving direction along the path, used as a decisional metric for the "simplest" path.

About weighting function, there are several studies that contain classifications of route instructions that might be used as a basis for a weighting function. Mark [Mar86] proposed a modification of the A* shortest path algorithm, which took into account both the total length and the "ease of description" of the route. He classifies different intersections according to the complexity of the instructions needed to successfully negotiate that intersection. Using this classification, Mark's algorithm adjusted the weights used in the shortest path computation to preferentially select routes through intersections that could be described using less complex instructions. Duckham [DK03] proposed weights in terms in its operation and its results indicate that simplest paths are still comparable in length to shortest paths.

Another interesting research was made by Grum [Gru05], using the factor of lowest probability of a user getting lost. The proposed weighting function for evaluating the risk value of each decision node is:

$$\frac{2\times\sum(length\ of\ wrong\ choices)}{(possible\ choices)}$$

Each compared path has a total weight that is the sum of the risk value of each path's node, than the lowest one is the "simplest" according to Grum. The explanation of the risk value formula is easy:

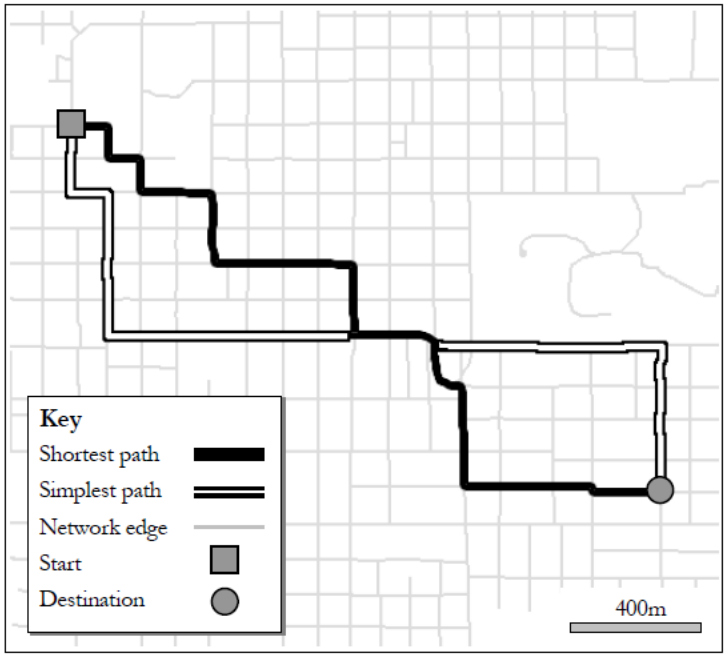


Figure 2.10: *Example:comparison of a shortest and simplest path*

- in each decision node you have more than one choice (number of possible choices),
- each choice is an edge with a metric distance to the next decision node associated (length of choice),
- excluding the right edge to follow, the others are wrong choices each one with its length of choice (length of wrong choice);
- summing the lengths of wrong choice you obtain the numerator;
- than the formula is completed dividing by: possible choices.

About computing multiple shortest paths, the reference implementation is the Yen's k-shortest path algorithm [Yen71]. There are few indoor navigation research's projects that use this way to compute path comparison, one is C-NGINE project (Contextual Navigation Guide for Indoor Environments) made by Nikoloudakis but they not implement a weighting function for a "simplest" path choice [NKB+09]. Another prototype software called OntoNav use k-shortest path but leave to the user the choice of which is the metric to use: path length, travel time, etc... [TAK+05]

2.4 Interaction

Route directions are a primary means to guide someone in finding one's way, they are task oriented specifications of the actions to be carried out to reach the destination. The process of communicating route directions is fertile ground for scientific inquiry, it describes what can be referred to as cognitive transaction: an individual who needs a route direction seeks it from a knowledgeable source, the transaction is successful if he states his inquiry unambiguously. The source of information understands the request, has the specific information needed and conveys the route directions in comprehensible form. Obviously, there are a number of ways in which transactions of this type can be achieved. About externalize route directions there are different techniques divided by the human senses:

- **Hearing**: iLBGS based on Audio speech, for instance, use recorded directions or speech synthesis to provide directions to the users. Speech may be safer than using a display as it requires less attention and can be easily facilitated using an headset without impairing the user's normal navigation capabilities and surrounding awareness, but they are language dependent and are not suitable when the environment is noisy or when a user is hearing impaired [RHM04].
- **Touch**: Haptic technique are also another way to communicate route directions, they provide output using the sense of touch and do not interfere with user's ability to sense their immediate environment using sight or hearing [WH05], but they're not very used due to costs of the wearable user guidance system.
- **Sight**: Visual technique is the most used way to provide directions using a display of a public screen or a PDA[5], it needs a good surrounding view to be performed.

In outdoor navigation systems, we usually find a combination of visual and audio techniques, both usable because modern vehicle are well soundproofed and with a large view of the surrounding space. Instead in indoor navigation, we have to deal with different conditions: usually we travel through noisy environments, with reduced visibility due to presence of others people in the same space whom partially/full cover the view, and usually with something to hold in your hands (bags, hand

[5]Personal Digital Assistant

of a children, etc...). The positive side is take humans walk slower than vehicle, and can stop/turn/think/act whenever they want without time restrictions.

Common outdoor techniques can't be applied for indoor navigation as they are, we need new approaches adapting the existing ones. Due to all the described constraints, visual technique is the only one partially usable for indoor navigation purpose. We can resume different useful visual presentation forms as follows:

- **2D maps**: especially floor plans, are still the most popular presentation form. The main reason is certainly their pervasive use in physical guides (such as paper maps, You-Are-Here maps installed in the environment). [BCK05]
- **Textual or verbal instruction**: is the most simple presentation form for navigation. Concerning route communication, textual guidance is similar to verbal guidance. The only difference is that when using textual guidance, users have to read the text on the screen [Rad07].
- **Augmented Reality**: it can give a good overview of our environment and can contain a lot more detail than traditional maps. Unfortunately it is very difficult and sometimes even impossible to use them as navigation aids. One of the main disadvantages lies in the large data files and their high requirements on memory space, a more precarious problem is the demand on display size that is necessary to present the complexity of a 3Dfile. [Rad07]
- **Image with arrows**: can be a helpful presentation form in a navigation system, even though it is not really valuable as a navigation aid. Users need a lot of time to compare reality with the photograph, could be an optional choice that people can choose to gain additional information. The same holds for panorama views. [Rad07] [Rad03]
- **Video**: have similar properties as photographs and their potential as route information aids can be rated as rather low. Objects can be directly compared and identified with reality, but the quick movement of the film often provokes the loss of orientation because it does not give a lot of time to watch everything in detail. [Rad03]

For indoor-LBGS, maps (especially floor plans) are still the most popular presentation form. One reason is certainly their pervasive use in physical guides [BCK05]. An important issue for maps is their orientation in relation with User one and with the surrounding space [AW92]. The common used technique is fixed orientation (e.g. north-up) but someone with more orientation problems requires a "forward-

up" or "track-up" alignment: the top is aligned to the human front direction and the bottom with the backward one, and left and right on the map are left and right in the surrounds. Map alignment has implications for the design, the common solution is to allow both orientations.

Despite the widespread use of maps, however, travelers still frequently make use of verbal directions [FMG+90]. Textual instructions are still the most simple presentation form for navigation [Rad07] and they are easy to create and can be used in almost every PDA[6]. Allen's cognitive psychology studies analyzed route directions' production and comprehensions, he describes all the challenges about the communication of spatial information with textual form. The production and comprehension of route directions is based on a **structural organization** of the route communication process, specifically has four phases [All97] [All00]:

1. **Initiation**: Ask for destination point and optionally constraints to be observed.
2. **Route description**: the respondent provides a set of communicative statements that provides information to reach the destination. It involves two types of statement: directive and descriptive. Route descriptions involve specific components, most importantly, environmental features, delimiters, verbs of movement, and state-of-being verbs.
3. **Securing**: includes questioner's reaction to the route description, clarification queries and confirmation statements.
4. **Closure**: it's a social convention, typically verbal indications that the episode is reaching a conclusion.

Route description's generation is the major challenge in Allan's framework, how to generate them is the main objective of the next section.

[6]Personal digital assistant

2.4.1 Generate Low-Level Route Directions

Once the path has been calculated, the first thing to think about is directional information that allows locating entities in space and define the user orientation. Directional relations are used in several respects in route directions: they state the location of entities encountered along the route (like landmarks) with respect to the way finder or other entities; they announce a change of heading at decision points, e.g. represent turning actions; and they may relate these actions to an entity's location to better anchor them in space. People represent spatial knowledge, such as distances and directions, qualitatively. Since distances and directions are represented as qualitative categorical knowledge, people apply these categories also in route directions. In research on qualitative spatial reasoning, several qualitative direction models have been proposed. These models divide the two-dimensional space into (labeled) regions. These sectors map all possible angular bearings to a usually small, discrete set of categories. Against homogeneous four-sector and eight-sector direction models, Klippel et al. empirically elicited such a heterogeneous direction model for turning actions in way finding (Figure 2.11) [KDK+04].

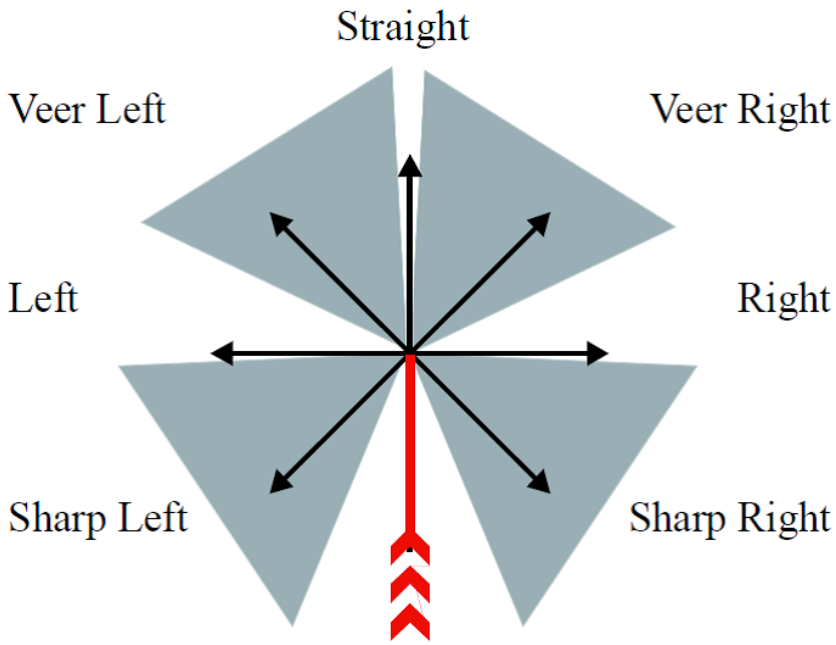


Figure 2.11: *qualitative direction's model [KDK+04]*

Clarified the direction concept and having a route planned, now we move to route directions communication. Since a path is divided into segments, we have a direction to follow for each segment. Associated to each direction we must provide instructions to communicate each time. Starting from a route direction MacMahon [Mac] proposed a framework, "Representing Route instructions", in which he introduced a general structure of an instruction, we can resume it as follows:

- **Simple Action**: route instructions low-level action types: *Turn*, change the user's orientation without changing position; *travel*, change location walking without changing direction along the path; *verify*, check an observation against a description of an expected view; *act*, do some action involving features in the environment such as "open the door", "overcome the coffee machine", "take the elevator"; *declare-goal*, declare destination reached.
- **View description**: describes known objects(landmarks) visible from the follower perspective, each annotated with appearance and positional information (see usefulness of Landmarks in the previous chapter). The positional information is the expected relative position of the object within the view, and based on the previous qualitative direction sentences, it's encoded in two attributes: *orientation* from the perspective of the follower and *distance*.
- **Compound Action specification**: match actions with view descriptions, using adverbs, verb objects, and adverbial clauses and prepositional phrases translated to *pre-conditions*, *while-conditions*, and *post-conditions* (e.g. describe direction to take, how far to travel, and/or the views that will be seen when the action is accomplished: a review is visible in the Figure 2.12.
- **Qualitative reasoning properties**: to preserve the properties for qualitative spatial reasoning [Fre91]: *uniqueness*, each object exists exactly once, preserved by combining multiple references to the same object of class of objects (e.g. "overcome three doors on the right"); *topology*, properties related to the physics of space, movement in space is possible only between neighboring locations, preserved by intrinsic data model construction; *conceptual structure*, properties related to the neighborhood of spatial relations, preserved by taking care to not over-specify spatial position or relations.

The importance of landmarks usage combined to the integration of "view description" in simple action of route instructions statements, introduce the need of an algorithm for landmarks selection, because not all the visible landmarks are useful

Route Instruction Turn Arguments

When	Phrase	Arg.	Action Model
Pre	COND	DESC	Face(faced=*Desc*)
Pre	LOC	VIEWD	Travel(until=*ViewD*)
While	DIR	Dir	Turn(direction=*Dir*)
Post	PURPOSE	DESC	Face(faced=*Desc*)
Post	TOWARD	VIEWD	Face(faced=*ViewD*)
Post	AWAY	VIEWD	Face(faced=*ViewD*)
Post	ONTO	PATH	Face(faced=*Path*)

Route Instruction Travel Arguments

When	Phrase	Arg.	Action Model
Pre	ALONG	PATH	Face(faced=*ViewD*)
Pre	COND	DESC	Face(faced=*Desc*)
Pre	LOC	VIEWD	Travel(until=*ViewD*)
Pre	TOWARD	VIEWD	Face(faced=*ViewD*)
Pre	AWAY	VIEWD	Face(faced=*ViewD*)
Pre	DIR	Dir	Turn(direction=*Dir*)
While	PAST	VIEWD	Travel(past=*ViewD*)
While	DIST	Dist	Travel(dist=*Dist*)
Post	UNTIL	VIEWD	Travel(until=*ViewD*)

Figure 2.12: *Verb complements, dependent clauses, and prepositional phrases are modeled as simple actions to be taken until a view description is matched against the observation model.*

for route description. An example in shown in the next figure, from Hampe publication [HE04] (Figure 2.13). There are plenty of studies involving this research question, but Landmark selection is still an open field of research, many people demonstrated that the characteristics and background of the observer have a significant impact on selecting and identifying landmarks; actually no one founded a general solution for indoor landmark integration. We can only state general rules: the majority of the used landmarks are located close to the observer [LHM99]; structural landmarks and objects have balanced preference, with or without a semantic component.

Putting aside at the moment the landmarks selection, we can explain what happens when there's more than one object of the same class along the path, e.g. doors, paintings, or other structural landmarks or objects considered as landmarks. Klippel et Al. [KHRW09] proposed a process called **chunking**, resumed as selecting and merging for a route direction more than one selectable landmark into one instruction. The proposed alternatives are:

- **Numerical chunking**: count items of the same class of objects and summarize them into a single instruction with count indication.
- **Chunking based on landmarks along the route**: select landmarks close to decision points and use them to give the next instruction based on their position against the next direction (e.g. turn right before the Landmark, take the corridor opposite to the landmark, etc. . .). Select landmarks along route

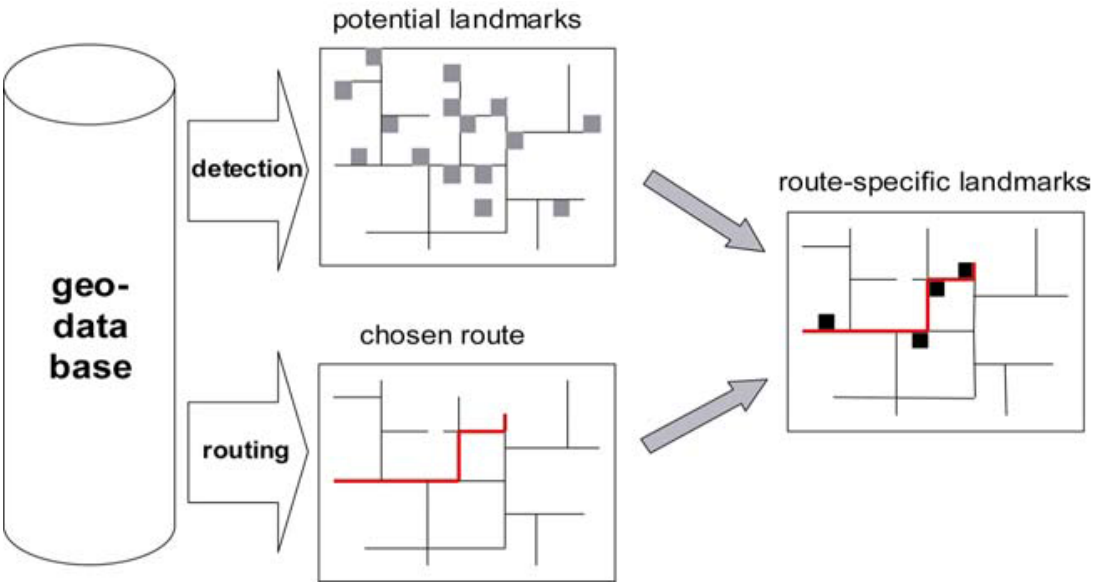


Figure 2.13: *Determine route-specific landmarks [HE04]*

to confirm the right direction to follow.

We can also apply chunking theory to subset of edges of the route path, this technique is called **segmentation** [KHRW09] [GD02] and consists of merge route directions of more than one path edge into unique instruction, using *Path segmentation* by splitting paths into all decision points, ignoring for instance only when the direction doesn't change along the route, or using *Landmark based segmentation* by splitting paths in order to create sub-goals, each one is a salience landmark along the route, easy to find and see. The output of this process is often a sequence of directions with a hierarchical structure, stored using a markup language to simplify the parsing of the instruction: typically XML is the language used to describe that route directions, that are still coded and not explicitly translated into natural comprehensive language, they are usually called *low-level route directions*.

Following the previous remarks on how to **generate low-level route direction**, Richter [Ric08] developed **GUARD** (Generating Unambiguous, Adapted Route Directions), a computational process used for generating context-specific route directions and their interplay. It consists of four major steps depicted in Figure 2.14. First, for each decision point all applicable low-level turn instructions are generated. The individuals turn instructions are then combined applying simple chunking

rules; these chunks are adapted to general chunking principles in a post-processing step. Finally, in an optimization process those chunks from the generated set of chunks that result in the optimal context-specific route direction are chosen. The process was developed for Outdoor navigation, using road network (Medial Axis based) dataset; as it is isn't easily adaptable for indoor navigation, it needs a review for pedestrian free movement and locomotion constraints. In the next image we also found an example of Context-Specific Route Directions for a given route made by GUARD system (Figure 1). Other standard XML models for low level route instructions has been proposed by by Mani et al. "SpatialML Annotation Scheme" [MHC08] [7], he proposed a custom data model with XSD schema similar to the GUARD output.

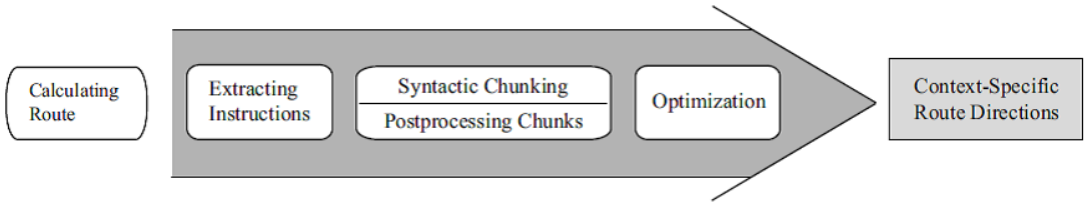


Figure 2.14: *Overview of GUARD, the generation process for context-specific route directions. [Ric08]*

Code 1: *Example of GUARD XML output for the given route [Ric08]*

```
1  <?xml version="1.0" encoding="UTF-8"?>
2    <ROUTE>
3       <Direction>
4          <instruction point="1" relation="straight">
5          </instruction>
6          <instruction point="2" relation="right">
7             <landmark id="1" type="Map"/>
8          </instruction>
9       </Direction>
10      [...]
11   </ROUTE>
```

[7] http://sourceforge.net/projects/spatialml/

2.4.2 Generate Natural Language instruction

The last step is translate the low level route instructions into **high level natural language** ones, this is a research field covered by CORAL[8] project of Robert Dale, Sabine Geldof, Jean-Philippe Prost [DGP02] [DGP03], a guidance prototype's image in shown in Figure 2.15. Also Cuayahuitl et Al. [CDR+10] improved algorithms for natural language generation starting from an XML file(Figure 2.16) and Geldof's research RPML:"Route Planning Markup Language" [GD02] covers this field of research with his proposal.

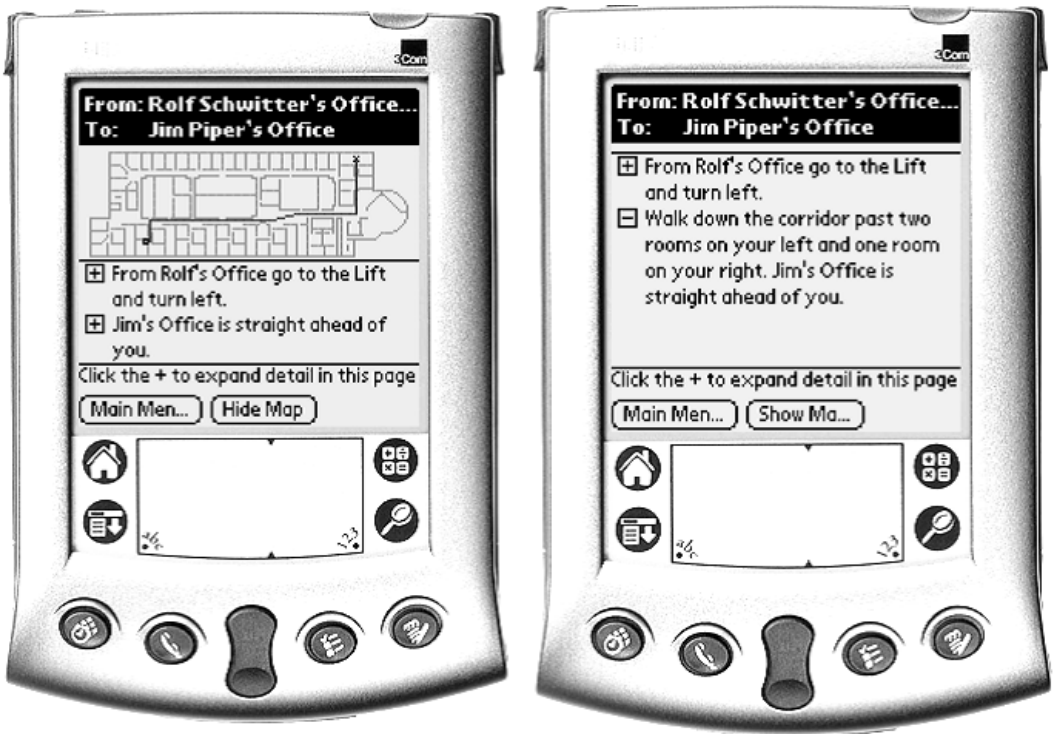


Figure 2.15: *CORAL natural language generation prototype [DGP02]*

[8]http://web.science.mq.edu.au/ coral/

Figure 2.16: *Sample route with high-level instructions derived from applying Cuayahuitl et Al. Algorithm [CDR+10]*

2.4.3 Google Indoor maps

We want to spend a paragraph for the only one commercial and widespread navigation system that is starting providing indoor navigation in addition to outdoor navigation: Google Maps[9]. It is a common web service for outdoor representation of spatial data and outdoor guidance. It provides a user interface for plan paths between points on the maps; then provides textual instruction mixed with map visualization of the route for guidance. Now (2013) it's getting even more useful by providing maps for indoor spaces such as airports, shopping malls, large retail stores, and transit stations. This new service integrate the indoor data modeling to the pre-existing outdoor one. It use visibility approach to model indoor spaces, provides input fields for starting and ending point, then shows guidance using maps and textual instructions referred only to the changes between floors, not a detailed guidance step by step. They ensure dynamic positioning by showing on the map the current position and direction of the user using Wifi Signal-Strenght. The following figure shows a route between two points of interest (Subway restaurant on the 2nd floor and Lego store on the first floor) inside the "Mall of America" (Minnesota - USA). The Figure 2.17 highlights the datamodel approach used, visibility graph, and the provided directions visible on the left. This textual direction are very simple, and focused only on distances to cover and high level directions between floors, it's not comparable with GUARD or CORAL system. It's the first version of this system, actually not frequently used as outdoor maps, but certainly they will implement more features for this application in the future, and for sure it will become a reference system for indoor navigation.

[9]http://www.google.com/maps/

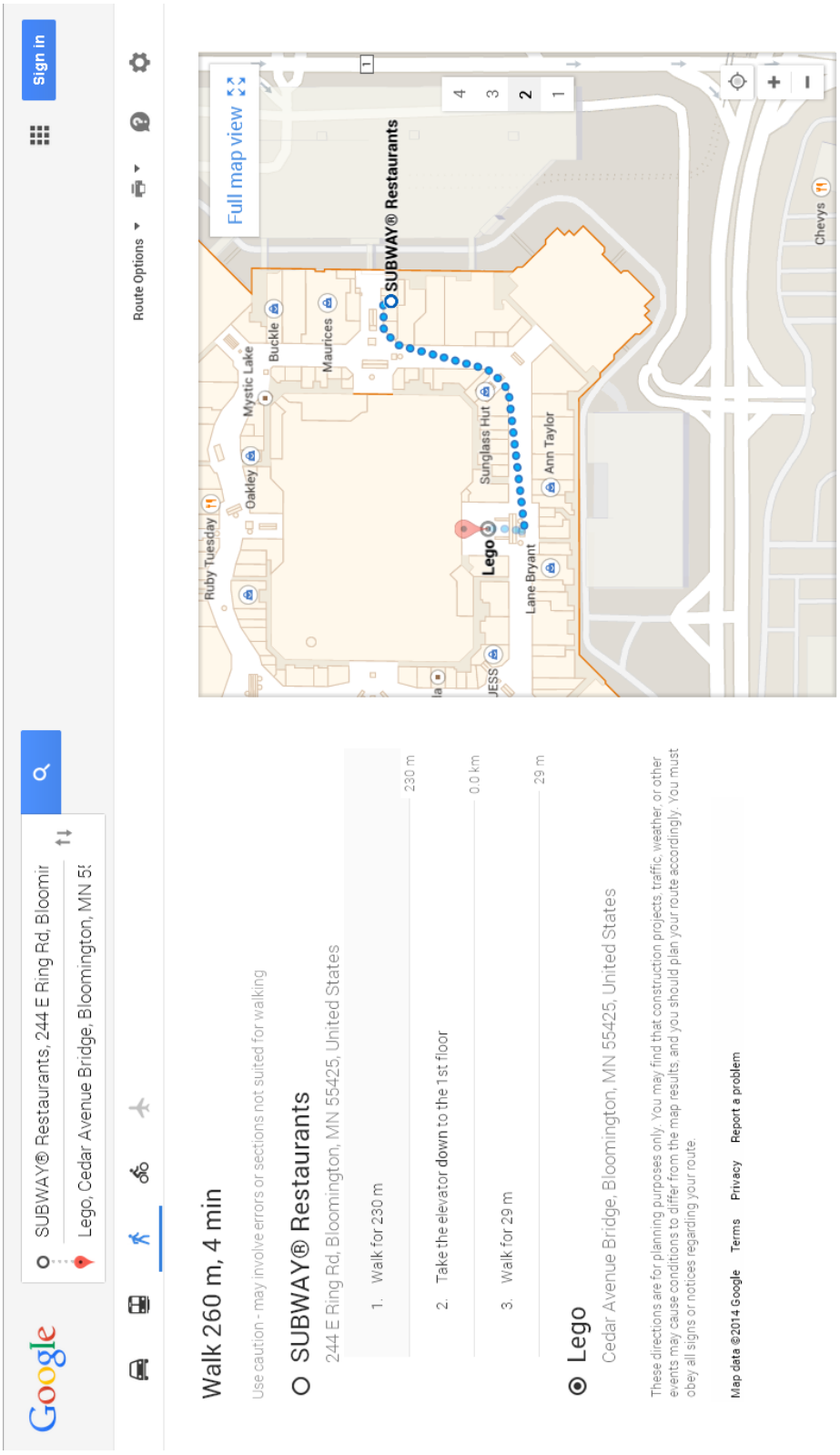


Figure 2.17: *Google indoor maps with an example of route direction*

Chapter 3

Methods

This chapter describes and explains the methodology employed in this study, starting from the "Approach overview" section with research question and sub-objective. Then we explain in the following sections the choices made to achieve the goals; according to framework's dimensions introduced in chapter one we follow this order: Data modeling, planning route, positioning and interaction.

3.1 Approach overview

After the literature review, it's clear that a guidance system that answers at the same time to all our requirements doesn't exist yet. Ones that use instructions closest to real world spoken guidance, with landmarks integration, are very few; also about representation of indoor environment that solves large space ambiguity there are only few proposals, but generally speaking no commercial and wide spread guidance systems have been released yet. We made a short review of some of the cited projects/prototypes proposed in the previous chapter, filtered by using the requirements specified in the first chapter 1.3 of this thesis. We can satisfy all the requirements having found a solution for each one with a reference project or research paper. The Google indoor maps service is the reference project for **open spaces modeling**: geometrical data model, based on visibility theory, is a solution of large spaces modeling; and **route planning**: shortest path algorithm implemented on it is the easiest path planning technique, also used

in most of the Guidance systems; it provides also route directions using textual instructions but they are not detailed and without hierarchical structure. Also The Y.A.MA.MO.TO. framework [Sta08] implements geometrical data model using visibility graph, and is secondary reference project for data modeling. The Richter's framework GUARD [Ric08] is the reference paper for most of the requirements, **humans orientation**: it allows egocentric orientation in route direction generation; **landmarks usage**: GUARD based systems include landmarks in route direction generation; **real-world directions**: it generates textual instructions as the closest solution to real world spoken guidance; **hierarchical instruction**: chunking and segmenting paths into smallest sequence by applying hierarchical graphs theory is the answer to hierarchical instructions; **directions generation adaptability**: it allow flexibility and adaptability of Route directions generation to the real world scenarios; and **internationalization**: the introduction of low level route instructions, that splits route directions generation from natural language translation module (CORAL project [DGP03] is the reference for natural language generation). The iNav project [Aeb12] is the reference project for **positioning system** requirement: localization using QRCodes is a solution with low infrastructural costs and adaptability to emergency cases; and for **personal digital assistance** requirement: smartphones technology is the easiest and wide spread way to communicate with users using PDA systems. We have resumed all in the Table 3.1.

Requirements	GUARD	Y.A.MA.MO.TO.	Google indoor maps	INav
Open spaces modeling	NO	YES	YES	NO
Route planner	NO	YES	YES	NO
Positioning system	NO	NO	NO	YES
Humans orientation	YES	YES	NO	YES
Landmarks usage	YES	NO	NO	NO
Real-world directions	YES	NO	YES	NO
Hierarchical instruction	YES	NO	NO	NO
Directions generation adaptability	YES	NO	YES	YES
Internationalization	YES	NO	YES	NO
Personal digital assistance	YES	YES	YES	YES

Table 3.1: *Requirements VS projects from literature review*

In this thesis we want to answer to all the requirements by merging the reference projects explained before, catching the best solution for each criterion. The aim is to go a bit in deep in this new way, than the research question of this thesis is:

"Can be a visibility data-model approach useful for generating low-level route directions using landmarks for an indoor Location Based Guidance System?"

Parallel to the main research question, focused on low-level route instruction generation, several objectives are derived and defined:

- **Main objective**
  1. Analyze human wayfinding instruction principles and define a route instructions domain model(using UML notation);
  2. develop an algorithm for automatic generation of low-level route instructions using landmarks;
  3. Implement algorithm using Java programming language and export re-

sults into XML file format.

- **Related objectives**
  4. Define the **domain software model** for environment modeling, using UML standard notation.
  5. Implement shortest path algorithm.
  6. Develop a desktop based interface (only for testing purpose) showing dataset, providing input form for path testing and showing planned path.
  7. Develop a sketch of a Natural Language generation algorithm (only for testing purpose).
  8. Implement a WebService providing route instructions generation by URL request/response;
  9. Implement QR-code positioning on "IndoorNav" using built-in camera as reader.
  10. Implement a software prototype called "IndoorNav", based on Android Operating system, with a user interface for Guidance.
- **Data set and testing**
  11. Generate testing dataset based on Geometrical model of the OTB building of TUDelft University (NL), using CAD modeling software.
  12. Test the "IndoorNav" prototype using real world scenarios.

In the following we introduce an overview to the methodology used to answer to these objectives, explaining in deep the **Related Objectives** solutions and introducing generically the main objective approach. This one is then detailed in the next chapter 4; and tested in the chapter 5. The programming language used in this thesis is Java, both for algorithms on Web Server and on Android client application. The framework used for development is Eclipse, Web services are hosted on web using Apache Catalina TomCat web server, and Android SDK refers to Platform version 4.2 (Jelly bean).

3.2 Environment Domain Model

The domain model represent the key concepts of the domain, identifies the relationships among all the entities and lists their attributes, provides a structural view of the domain, describes and constrains the scope of the problem domain. The domain model can be used to verify and validate the understanding of the problem domain among various stakeholders. It defines a vocabulary and is helpful as a communication tool. Since our chosen data model is based on Liu et Al. "Door-to-door visibility approach" [LZ11], spaces assume only additional semantic information to the main graph, so we can't take into account models such as CityGML or IFC.

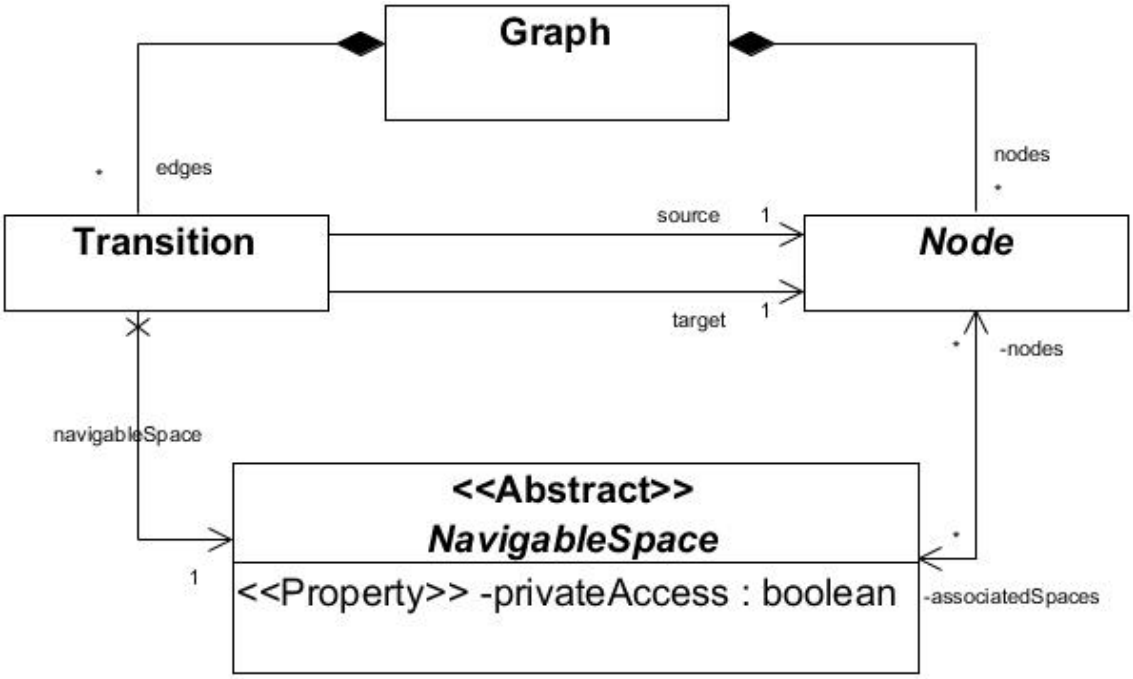


Figure 3.1: *UML class diagram of simplified Graph description*

Our model is based on **nodes** such as Openings (Windows, doors and virtual-doors), Concave corners and Furniture objects; and **transitions** that connect each node to the visible others. Virtual-Door concept is the same introduced in the previous chapter2.1 according to Chown et al. definition of gateway between spaces [CKK95]. An example is shown in the Figure 3.2, in which starting from the building geometrical model of a floor(a), we highlight doors and concave corners and connect

them using visibility approach (b), and then we remove the geometrical data of the surrounding spaces , leaving only nodes and transitions (c).

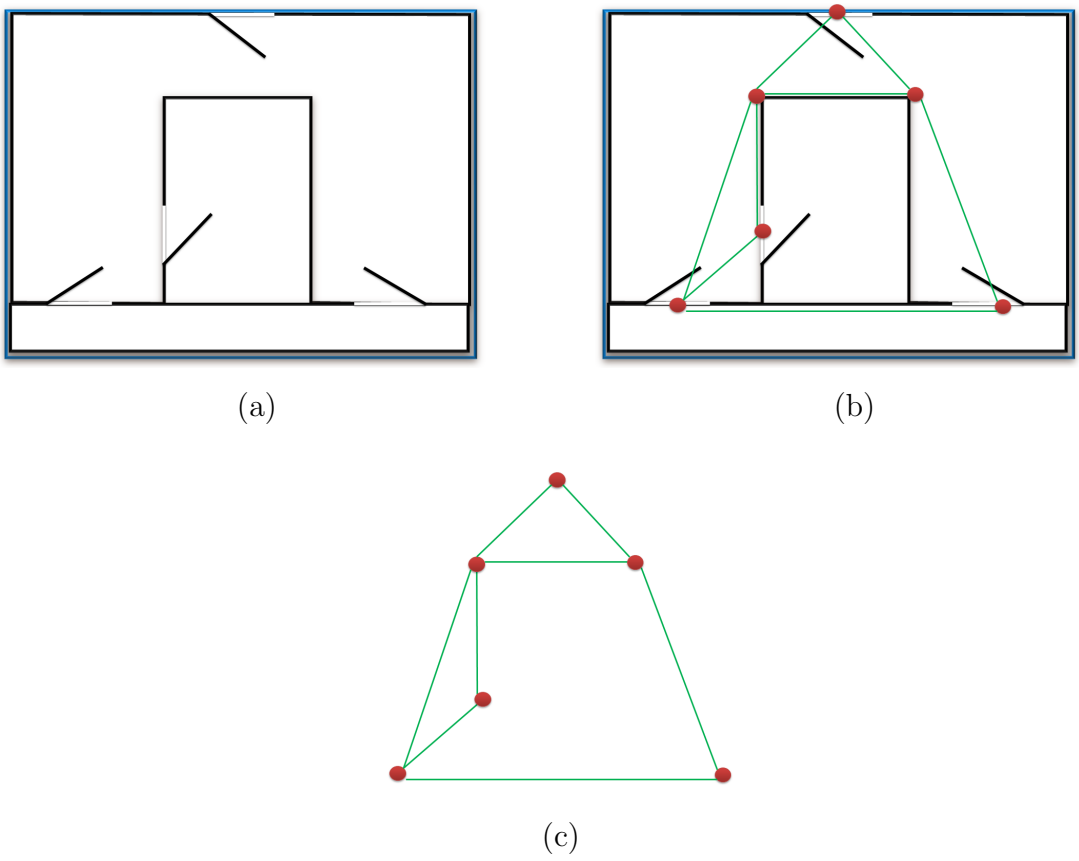


Figure 3.2: *Door-to-door visibility graph generation example, red dots are **nodes**, green segments are **transitions***

In order to maintain metric distances between nodes for distances calculation purpose we added a geometrical 3Dpoint to each node that represents the three-dimensional position into the environment. Each node has one or more "Semantic-Spaces" associated, for instance a door-node has two spaces associated. For positioning needs each door can have one or two QRcodes associated(one for each side of the door) that identify the position of the users and, if needed, in which side of the door is the person. Another important data stored in Openings objects is the "direction" concept. Assuming that a human, while passing through an opening, maintain the travelling direction perpendicular to the opening (virtual) surface, we store this direction (one of the two possible directions, for instance enter/exit from a door). We measure the counterclockwise angle in radiant between the direction

and a plain reference system with an X and Y axis decided before data set generation and then used (the X axis) as reference for angle calculus (Figure 3.3). Also Images can be attached to a node as additional data. An overview of the graph is shown previously in Figure 3.1, and the UML class diagram of nodes domain model is shown in Figure A.1 of the Appendix A.

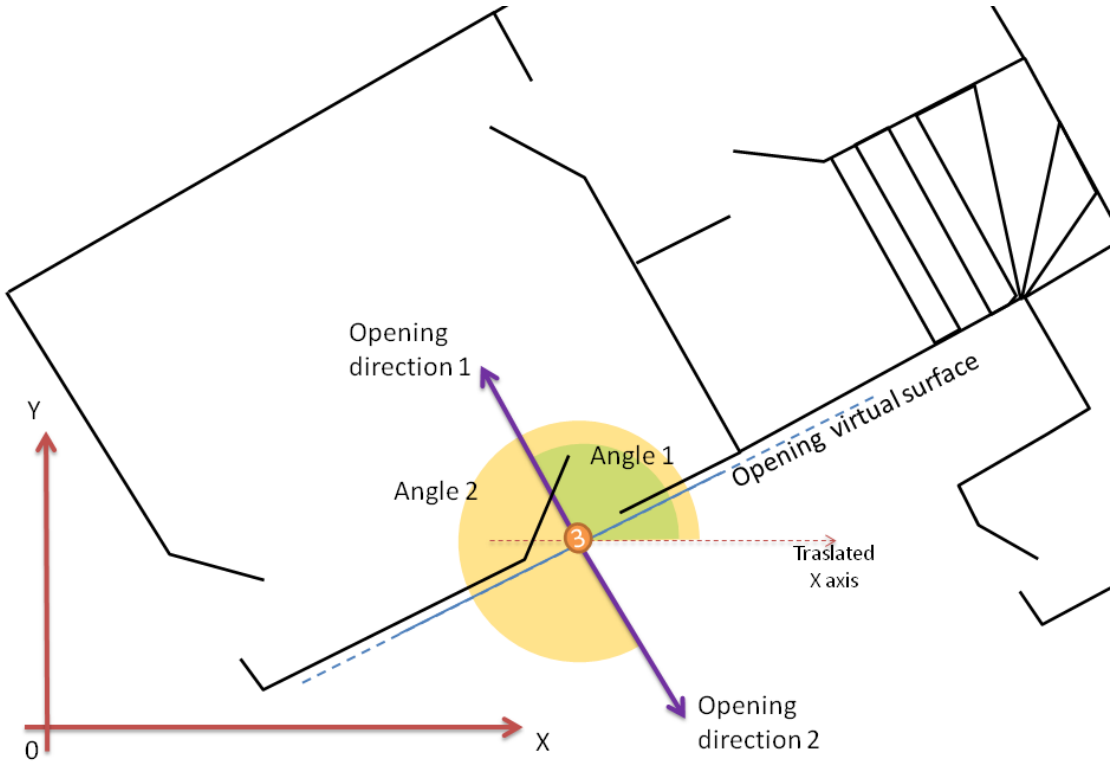


Figure 3.3: *Openings direction concept, for instance a door(as node n3) with virtual surface(blue segment), two perpendicular feasible directions(violet arrows), with relatives counterclockwise angles(yellow and green) referring to the X axis(red arrows)*

Semantic spaces are modeled as additional information to the Nodes and the Transitions, as shown in the Figure 3.1. At first we introduce the concept of **Building** as shown in Figure 3.4 (a), splitting it into floors (b), and then highlighting floors in yellow, as **HorizontalSpace**, and vertical connectors as **VerticalSpace**. Each VerticalSpace and HorizontalSpace is then divided respectively into **VerticalUnits** and **HorizontalUnits** that are **NavigableSpaces**(for instance rooms, halls, staircases, elevators, etc...). Then **OutdoorSpace** is added to model the surrounding space around the building, and at last we can resume a Building as collection of VerticalSpaces, HorizontalSpaces and OutdoorSpaces. Each NavigableSpace has a collection of nodes that belong to it. The Figure A.2 in Appendix A resumes the entire semantic spaces domain model. Merging the previous three partial domain models, we obtain the complete class-diagram of the data model.

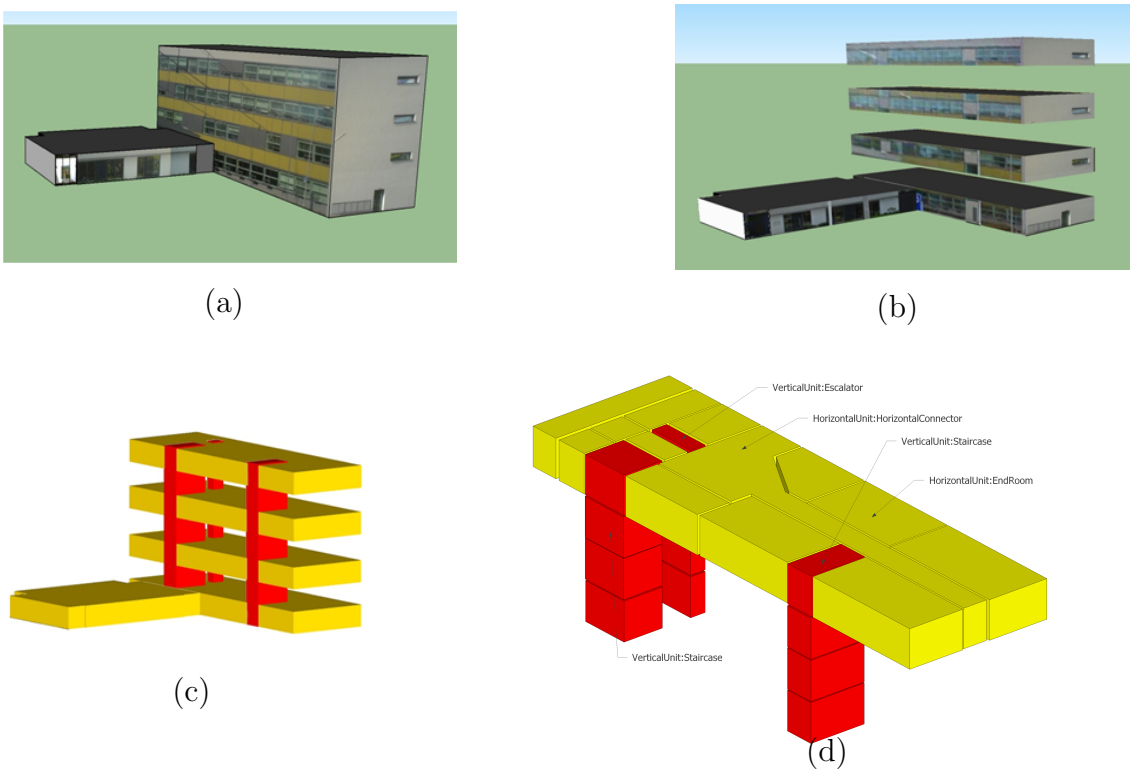


Figure 3.4: *OTB building(TUDelft University - NL) semantic space modeling of floors as four* ***HorizontalSpace*** *objects, and three vertical connectors as* ***VerticalSpace*** *objects. Each collector space, contains one or more navigable spaces, VerticalUnit or HorizontalUnits(d)*

3.3 Planning route and desktop interface

Given two nodes on the graph(for instance IDs 71 and IDs 488), using JGraphT[1] (a GPL free Java graph library that provides mathematical graph-theory objects and algorithms) we use Dijkstra's Shortest-Path algorithm to obtain the shortest path as Collection of Transition over the Graph. In order to understand graphically the datamodel and the computed path we develop a minimal desktop interface visible in Figure 3.5 with a dataset made from a floor of OTB building(see following Chapter 5 for complete dataset generation). The interface provides two text input fields on top, for start and end node ID input. We have provided an optionally checkbox for elevator usage (if user don't desire the use elevators). The buildings with more than one floor, are divided into floors using tabs named floor0, floor1, etc.... On each floor the interface shows nodes and transition of the dataset, and the computed path highlighted with blue arrows. The interface development has been necessary to check dataset and path calculus accuracy, before starting developing route instructions.

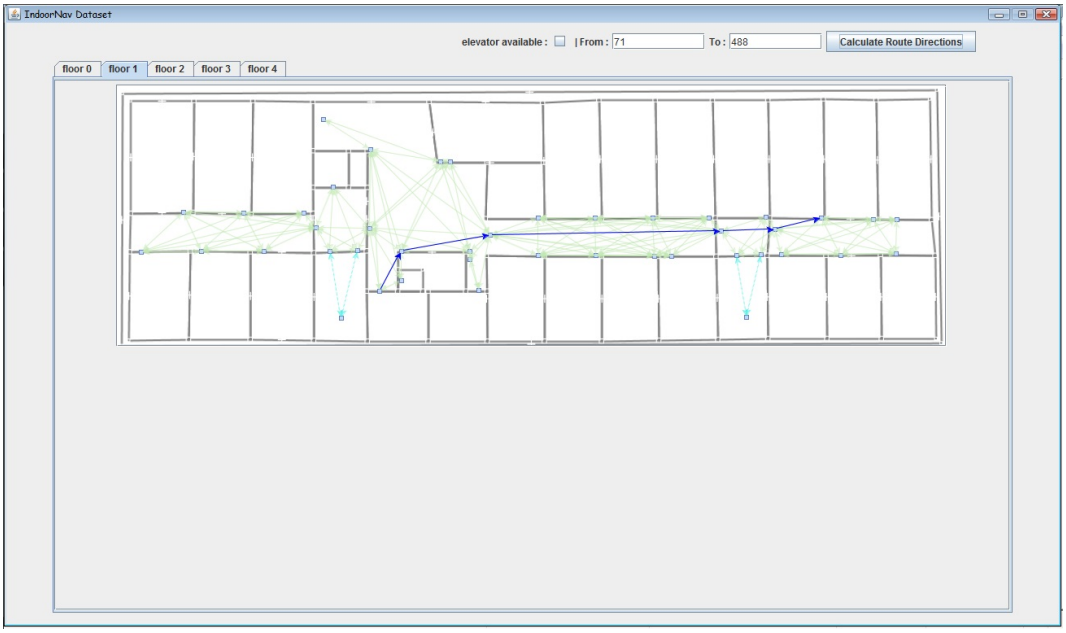


Figure 3.5: *Desktop interface with data set and paned path highlighted in Blue*

[1] http://jgrapht.org/

3.4 Low-level Route Direction generation overview

With the supporting interface and a working shortest path algorithm, we can introduce the core problem of this thesis: the interaction part of an iLBGS, and in particular how to generate low level route directions, that can be easily translate in various presentation forms(textual, arrows, images, etc. . .). We choose to follow GUARD system approach [Ric08], originally thought for outdoor navigation, adapted for indoor environments using visibility based data model. Steps followed are the same as Richter's ones: extract abstract instruction for each decision node, apply chunking rules on it, optimize them using segmentation. We assume that a path between two nodes of the dataset has been computed, having al least one transition. The output isn't a natural language series of instruction, but an abstract data representation, written in Xml format, that encapsulate all the information needed for a complete translation into next natural language. The algorithms implemented can be found in Appendix B, all the references to classes of domain model can be found as UML diagrams in Appendix A.

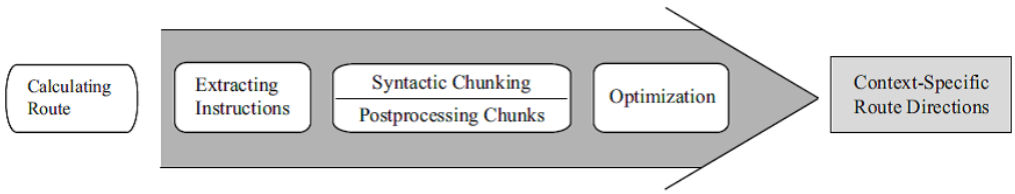


Figure 3.6: *Overview of GUARD, the generation process for context-specific route directions. [Ric08]*

3.4.1 Reference Dataset

Before starting, we want to introduce an example of dataset that will be used as reference for the route direction generation. The Figure 3.7 shows a floor of an imaginary test building, with nodes (orange dots with node number) and transitions (blue segments) and a path calculated between nodes 1 and 9. The Appendix C shows other figures, such as Figure C.2 nodes reference classes according to domain model proposed, the Figure C.3 represent the semantic space hierarchy between

units(HorizontalUnit and VerticalUnits) and collectors (HorizontalSpace and VerticalSpace) using red arrows, and nodes associated NavigableSpaces with blue arrows. Then we show in the Figure C.4 a computed path and associated NavigableSpace for each Transition of the path (also all the other Transitions of the graph not visible in the Figure, however, have their associated NavigableSpaces).

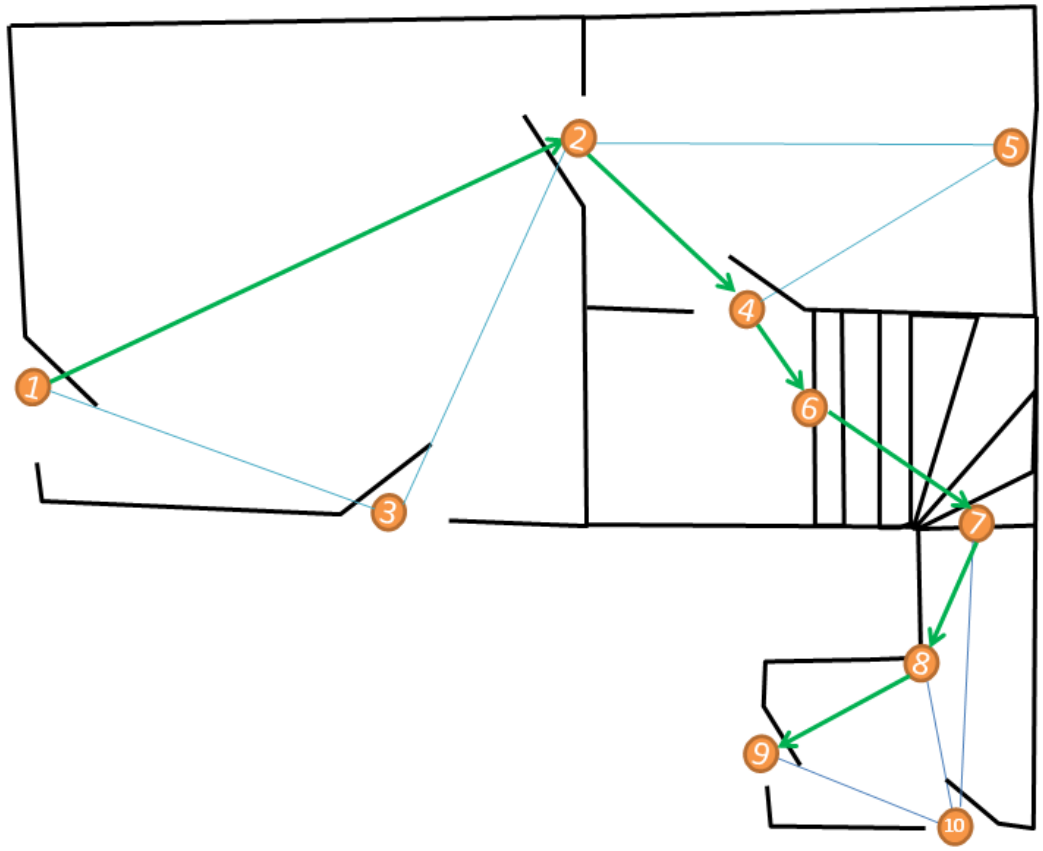


Figure 3.7: *Example of a Building floor, with visibility graph: nodes are the orange dots with node number and the transitions are the blue segments. A path between nodes 1 and 9 is highlighted with green arrows.*

3.4.2 Qualitative direction model

We explain in this paragraph how to apply egocentric qualitative direction model introduced by Klippel et al. [KDK+04], with the help of the Figure 3.8.
In each node of the graph, we have stored a direction as explained in the previous

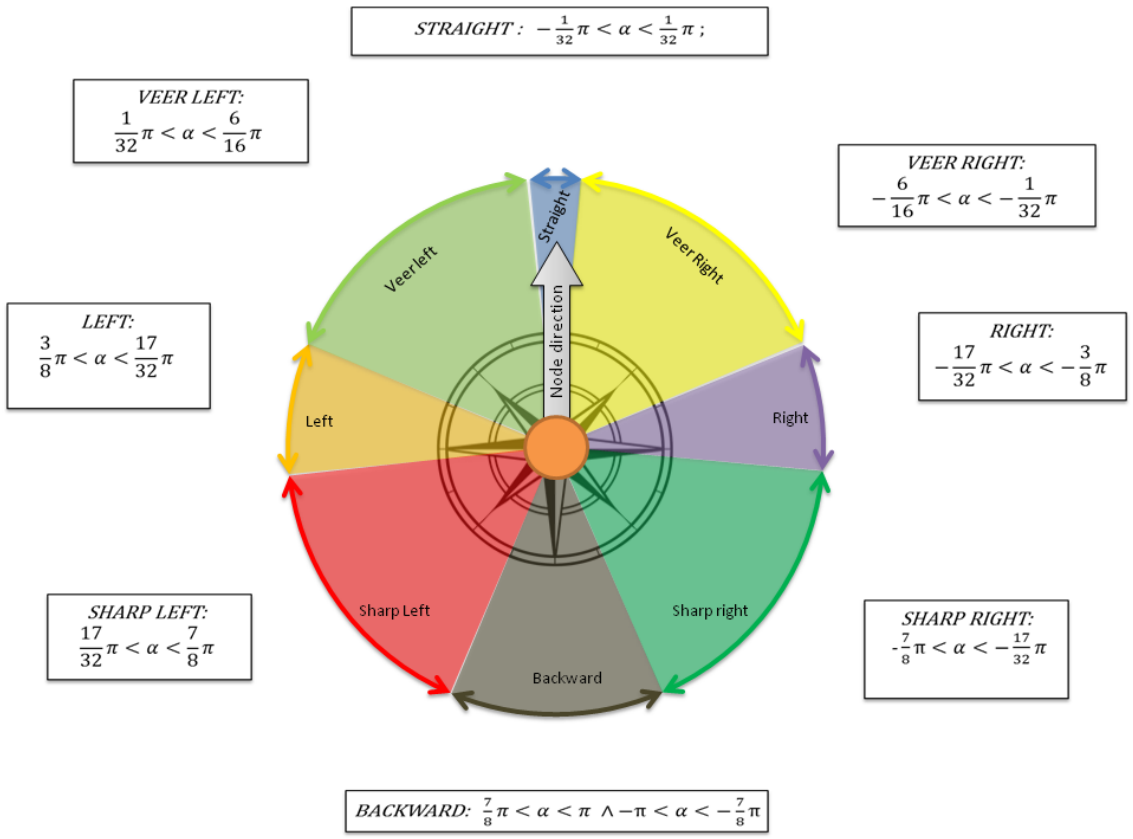


Figure 3.8: *Qualitative Direction model with reference angles of each sector*

chapter that represent the starting user orientation passing through this node. We put the direction model overlapping the node, centered on it, and match the node direction with the "straight" model direction (Figure 3.9). Than we can easily determine qualitative direction for all visible nodes matching all connecting edges with respective belonging sector. For instance when he user in Figure 3.9 pass through the node 1, he has two visible nodes, the node 2 that has the qualitative direction "Veer Left" according to green sector translation, and node 3 that has "Veer right" qualitative direction. So when we want to refer to the node 2 (in this case it's the next node of the path) we can use the qualitative direction model and suggest "Veer left and walk until…".

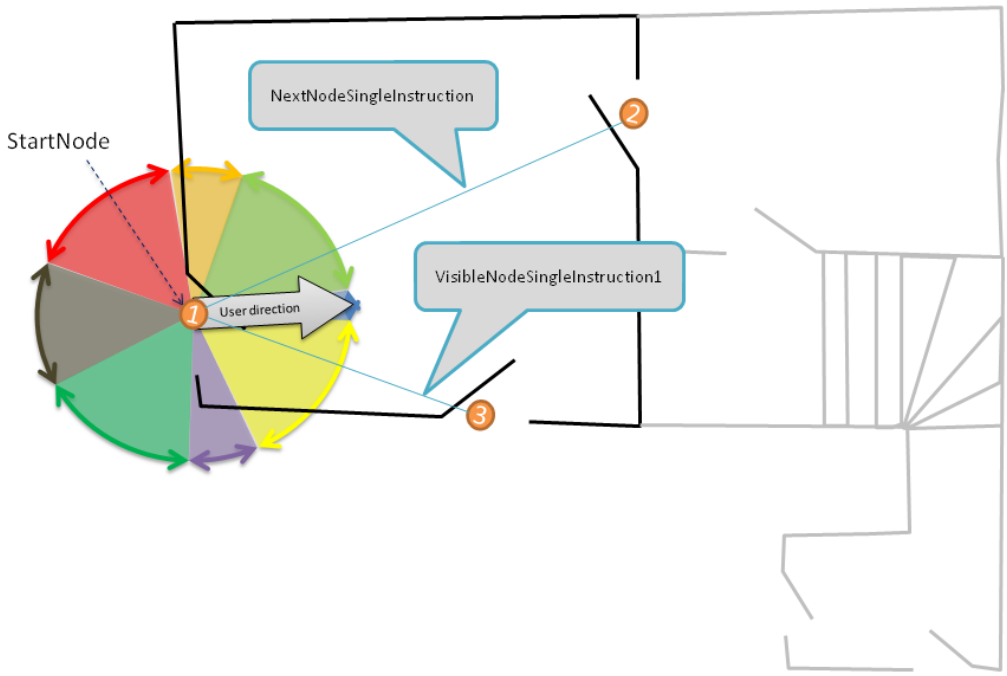


Figure 3.9: *Qualitative Direction model applied to the node 1 of this dataset example*

3.4.3 Route Directions Extraction

Starting from **extraction** step and using our visibility datamodel, for each edge of the planned path we create an object of the SingleEdgeDirection class. Then for each starting node of these list of path edges we need to collect all the outgoing segments that belong to the same NavigableSpace of the path's one. Then we generate an instruction to reach the next node by giving starting direction, path distance and next node name (for instance, it could be subsequently translated as "Turn right and walk for 10 meters reaching Door 1.240"), and then we add a direction for all the rest of visible nodes. We repeat this process for all the nodes in the path obtaining for each segment of the path a "next-node instruction" and a collection of visible nodes. For instance in the previous Figure 3.7, when user is located on the node 2 and need to reach the node 4, the transition that belongs to the same space of (2,4) is only (2,5), the others outgoing transition of node 2, that are (2,1) and (2,3) belong to different NavigableSpace and are not considered. So in this example, for node 2, we collect the edges (2,4) and (2,5) and define for

each one the qualitative direction, then we store the next node data about edge (2,4) separately from the others (in this case only one other visible node, so only edge(2,5)) using an object of the class NextNodeInstruction, and each of the others as new object of VisibleNodeInstruction class.

3.4.4 Route Directions Chunking

With **Chunking** step, we analyze all visible nodes directions of each node along the path, trying to highlighting nodes closest to the next goal node (for instance could be subsequently translated as"Door 1.240 is on the Right of the Escalator") and aggregate nodes of the same Class type for numerical counting instruction(for instance could be subsequently translated as "Door 1.240 is after three door on the right"). We collect, for each SingleEdgeDirection, visible nodes, if they exist, closest to the next goal node in both left and right side, within a fixed radius. We introduce ClosestChunkingInstructions that store also the belonging side of the closest node, as explained in the example in Figure 3.10. After that, we want to group(for each SingleEdgedirection) the visible nodes of the same class and then divide them into leftSide and rightSide referring to the next-node direction. We store each group of at least one element into another object of the class called NumericalChunkingInstruction, in which we also store the counting number of elements belonging to it. The idea is explained using the example in Figure 3.10.

3.4.5 Route Directions Optimization

Finally **Optimization** step is based on segmentation concept. At this time we have direction divided into path segments, but we want to merge some of them into higher level instructions using some criteria in order to achieve a hierarchical structure. We propose two criteria, first of all based on *room* optimization in which all transition on the same navigable space (for instance a room with non convex shape, without visibility from the starting and ending node and with corners to be overcome to reach the next opening node) are merged to create a room Instruction that resume all the direction in it. With this solution the user, only if needed, receive detailed direction step by step, and in most cases he is able to reach the next space without any other instruction. In details we encapsulate each SingleEdgeDirection

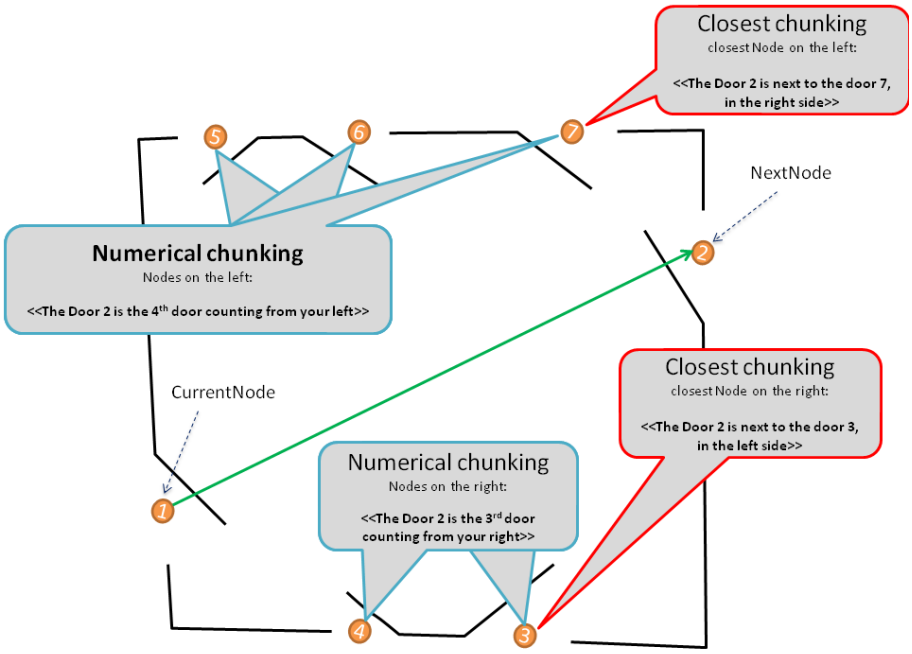


Figure 3.10: *Example of Numerical and Closest Chunking Instructions with a Natural Language Translation*

in a new object of RoomDirection class, and group them into the same RoomDirection object when the belonging NavigableSpace of consecutive edges remains the same. As shown in Figure 3.11, we want to remark what's happens between nodes 7 and 9 where edges (7,8) and (8,9) belong to the same navigable space.

The second criteria is based on *floor* optimization in which all VerticalUnits and HorizontalUnits directions belonging to the same collectorSpace (HorizontalSpace or VerticalSpace) are collapsed into one floor instruction as first (higher level) instruction to be performed, as shown in Figure 3.11. Usually user with VerticalSpace movement between floors needs only this instruction to reach the next HorizontalSpace (for instance a VerticalSpace instruction could be subsequently translated as "reach the third floor", and user doesn't need more instruction to reach the next step).

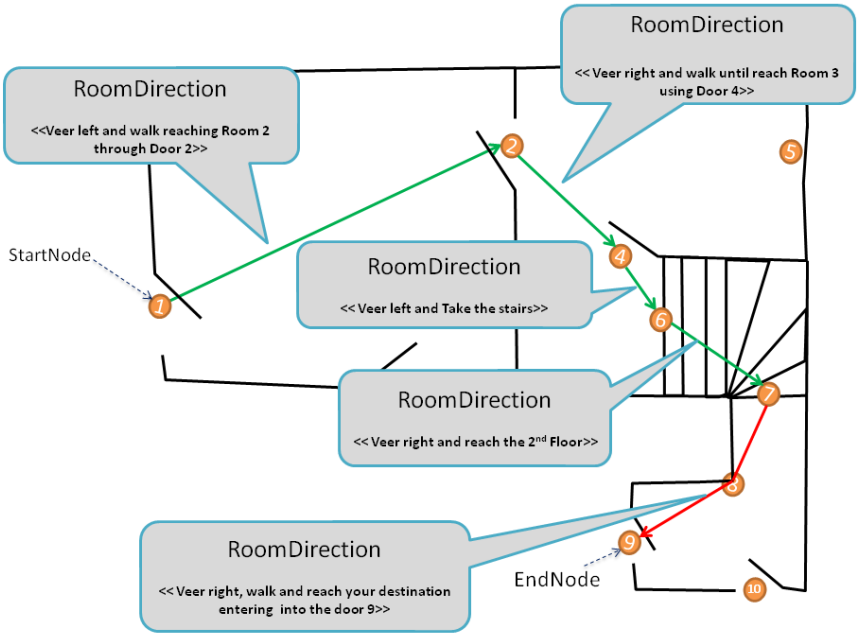


Figure 3.11: *Example of room segmentation with a Natural Language Translation*

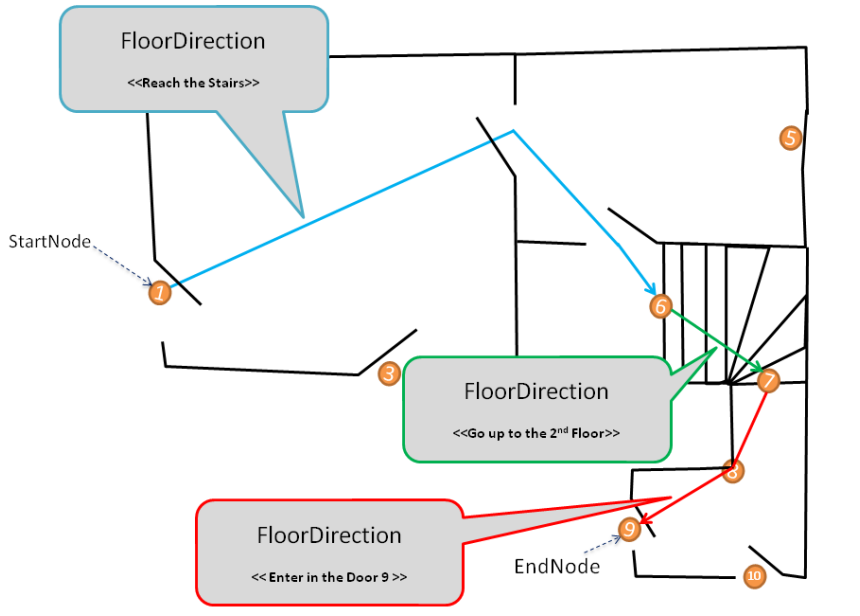


Figure 3.12

3.4.6 Route Directions process Output

As explained before, the output of this process is an XML file that contains all this instruction: the input of the next translation phase.

3.5 Natural Language Generation

About generating textual instructions, we decided to not go in deep in this field and use an easy English translation(only one language for testing purpose) of abstract route directions because this area of research is well covered by other scientific fields such as psychology of language. the reference projects are CORAL [DGP03] and [CDR+10]. The approach used is to translate floor instructions and room instruction using this pattern: qualitative direction + action verb + next space name. For next nodes instructions we provide direction + distance +next node name; for chunked visible nodes we provide numerical count of objects and leaving side, for closest nodes to the goal, we provide node type and leaving side. the output of the process is a completion of the previous input file, so always an XML format for the output, that can be sent to the interaction system to be presented to the user via visual interface. An example that resumes an instance of hierarchical directions with natural language translation is shown in the Figure 3.13.

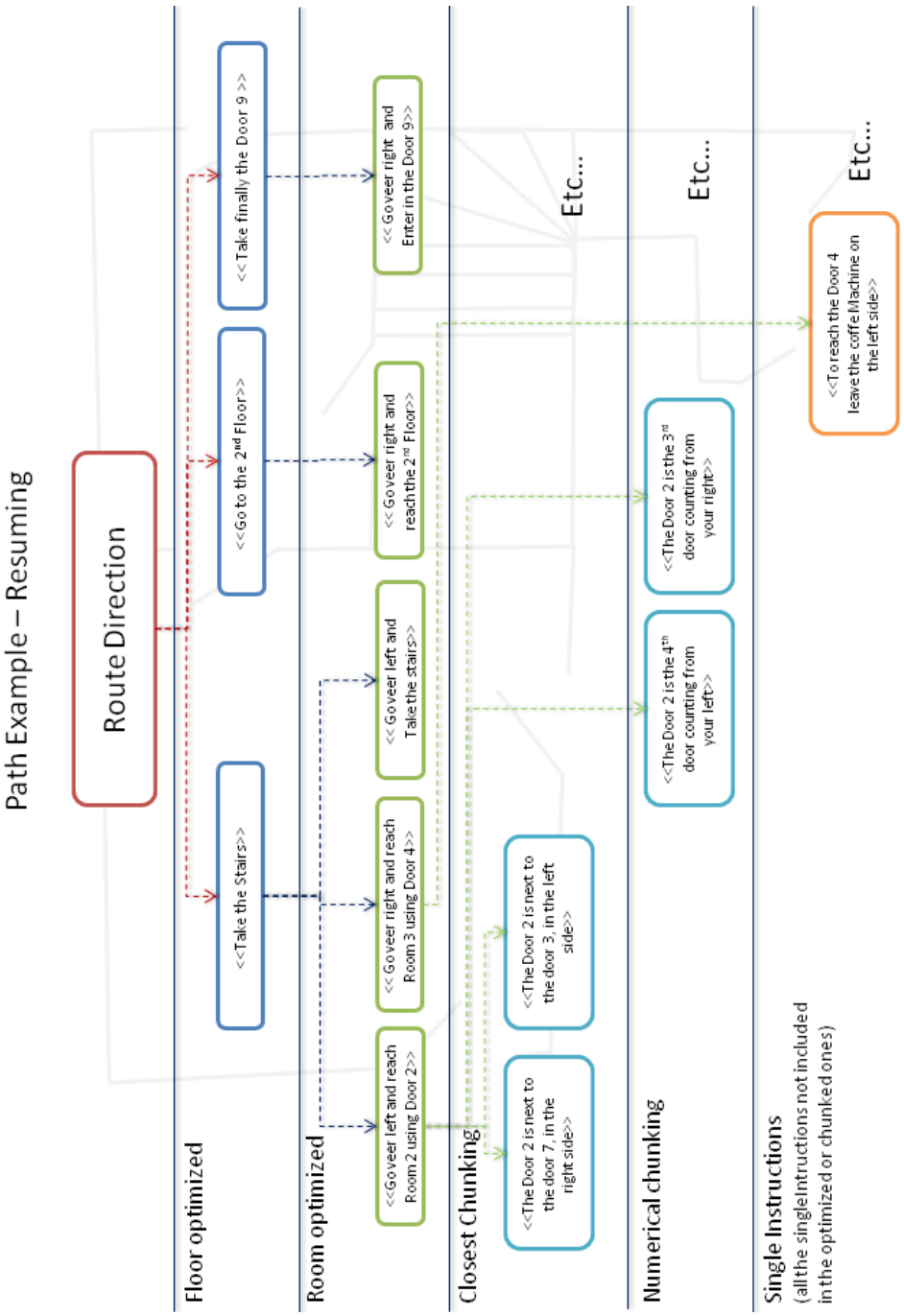


Figure 3.13: *Route directions output example, with hierarchical instruction and translation into natural language*

3.6 Web service

In order to provide route direction service to the users a web service has been developed. It's based on Apache Catalina TomCat Java server, a servlet called "RouteDirectionsServlet" answer to GET/POST web requests providing the XML file containing route directions with natural language translation. the parameters needed are: *from* and *to* node IDs, and elevator boolean usage value. The next Figure 3.14 shows an example of web service deployed [2].

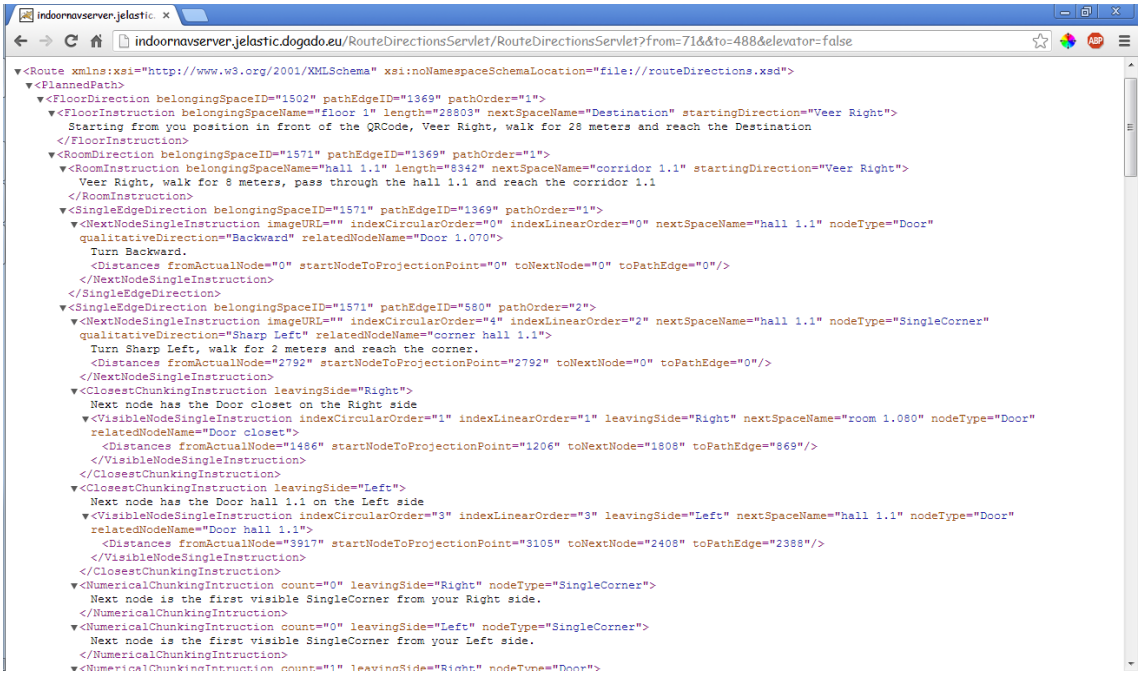


Figure 3.14: *IndoorNav Web service: route directions request and response on browser view.*

[2] http://indoornavserver.jelastic.dogado.eu/RouteDirectionsServlet/RouteDirectionsServlet?from=71to=488elevator=false

3.7 Android prototype: IndoorNav

Following our starting requirements of Guidance system using textual instruction with addiction of arrows and images (when needed) we have developed an Android based Application called "**IndoorNav**".

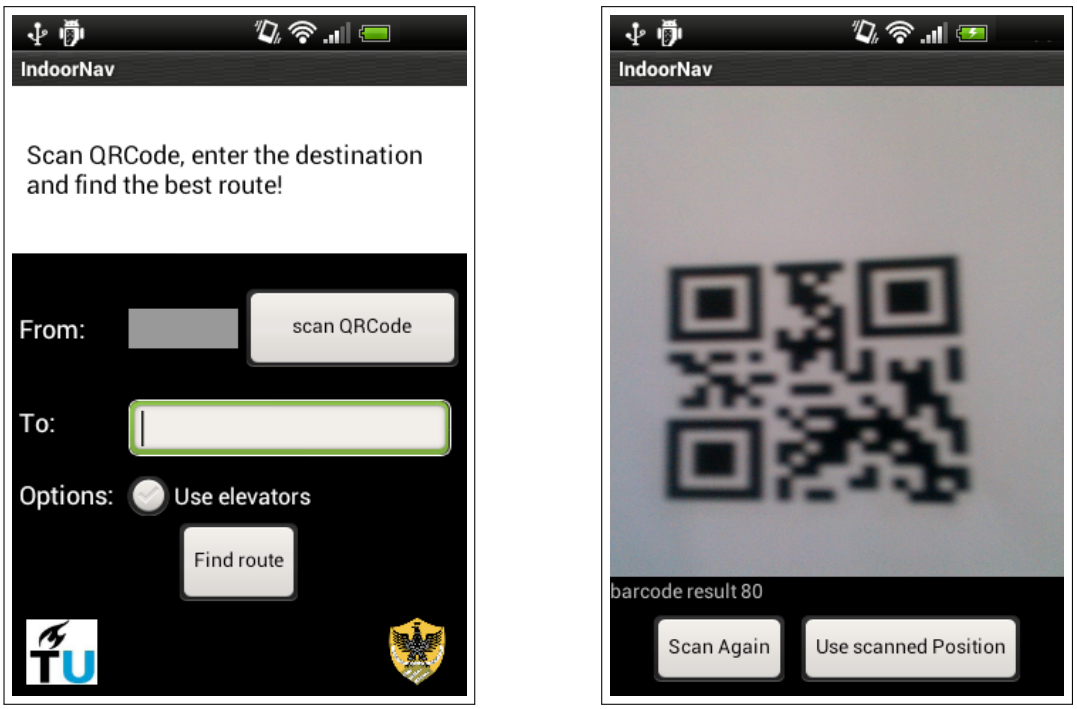
In order to answer to positioning research question, we used QRCodes as direct sensing tags (due to low infrastructural costs and high accuracy of positioning User positioning) developed using ZBar[3] Android library for QRcodes Image Scanning, that easily fits to our needs and doesn't requires too much time for integration in an android application. while creating a dataset, the management of the guidance system needs to create and assign to each door at least one qrcode per door that translates the nodeID into a 2DImage. The scanning library, when activated, automatically detects QRCode orientation and its embedded reference code and sends the code back to the caller application.

Figure 3.15: *Easy diagram of interaction between: User - Android Application - Web Service*

[3]http://zbar.sourceforge.net/

Before showing the interface and its basic functions we introduce the interaction between User-IndoorNav-WebService. The Figure 3.15 doesn't want to be an UML diagram, only a visual help to explain the system functioning. At first user scan a QRCode, it is used for understanding his starting position, then he enters ending desired point using interface interaction (and optionally disables elevators usage). Thirdly clicking on "find route" button on interface, the IndoorNav application send a request to the web service and (fourthly) receive an XML response that is (fifthly) parsed and graphically (sixthly) rendered in the next view so that user can read and follow textual instruction. Whenever user wants, he can go back and scan again a QRCode near to him in order to update his position and update route directions. In the Figure 3.16 we show the first steps of Interaction with IndoorNav application: scanning, filling and sending the request.

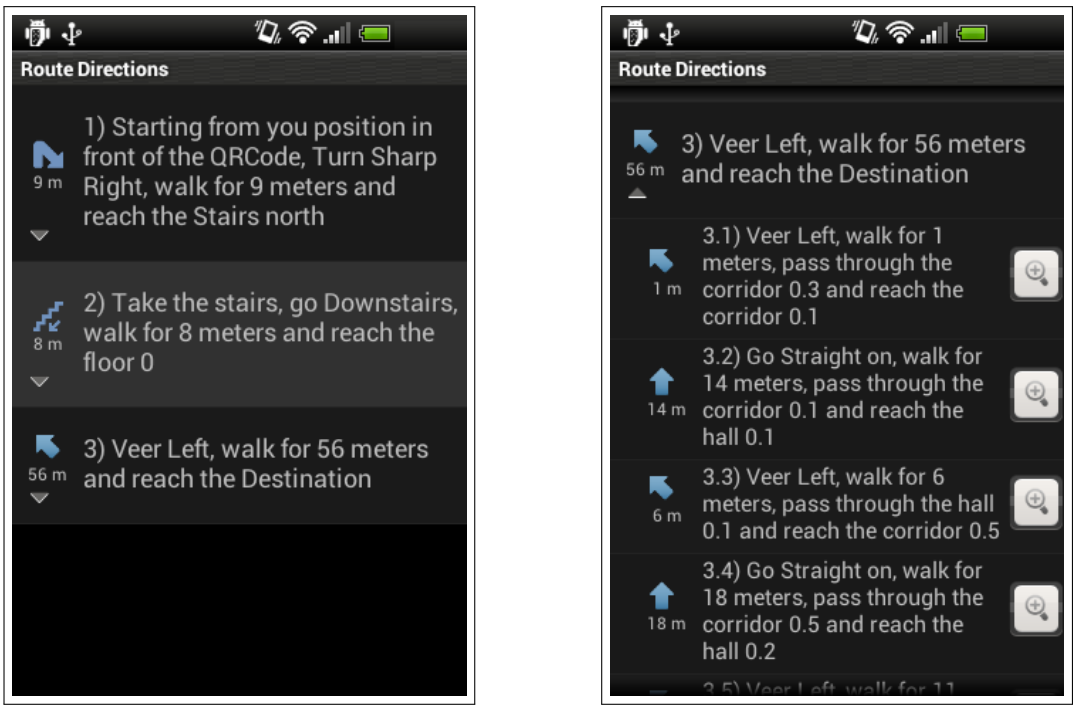
The presentation form of route instruction, following hierarchical structure of them, is shown in Figure 3.17. At first level interface shows Higher level instruction with an arrow indicating starting direction and the distance to be covered. Clicking on one of them, the interface shows the second level list, always with arrows and distance. When we need a third level of details interface provide a button on the right that shows a different interface with an arrow on top, then the distance, the textual instruction and optionally an Image of the next target node. Scrolling down this view the interface gives the possibility to receive other hints, such as chunking instructions and closest landmarks to the goal. All the instructions are numbered so that in the second level, users have always the reference higher level, for instance instruction number 3) and lower level instruction 3.1).



(a) (b)

(c) (d)

Figure 3.16: *IndoorNav screenshoots: starting interface(a), scanning QRCode(b),filling the requested fields(c),loading route instructions(d)*



(a) (b)

(c) (d)

Figure 3.17: *IndoorNav screenshoots of textual Instruction presentation: (a)the higher level instruction, (b) the room segmentation, (c)the single edge instruction with an image,(d)single path edge instruction with additional information)*

Chapter 4

Test and results

In this chapter we want resume our test of the prototype and show the results to analyze pros and critical points of the proposed navigation system. To do this we needed a building in which use IndoorNav app, and we needed its dataset according to our model. After that we given some different tasks to partecipants and collected results noting them on paper forms during users traveling along the route.

4.1 Dataset generation

Before starting our test, we need to choose a location for our test. The building in which we made this research, OTB research institute of TUDelft university, is the easiest choice. We used an OSM[1] file containing all geometrical data of the building including floors, rooms (with room number) and doors[2]. We used JOSM java openStreetMap editor for editing task[3] (Figure 4.1). Staring from this file, we decided to build the dataset according to our domain model manually, computing manually the visibility graph on JOSM editor and then implementing a Java parser to import these OSM file (XML based) in our system. We generated QRCodes images for each door node using Java qrcode generator library included in Zxing

[1] http://www.openstreetmap.org/

[2] Data set provided by PhD candidate Liu Liu of GIStechnologies dept. of OTB - TUDelft

[3] https://josm.openstreetmap.de/

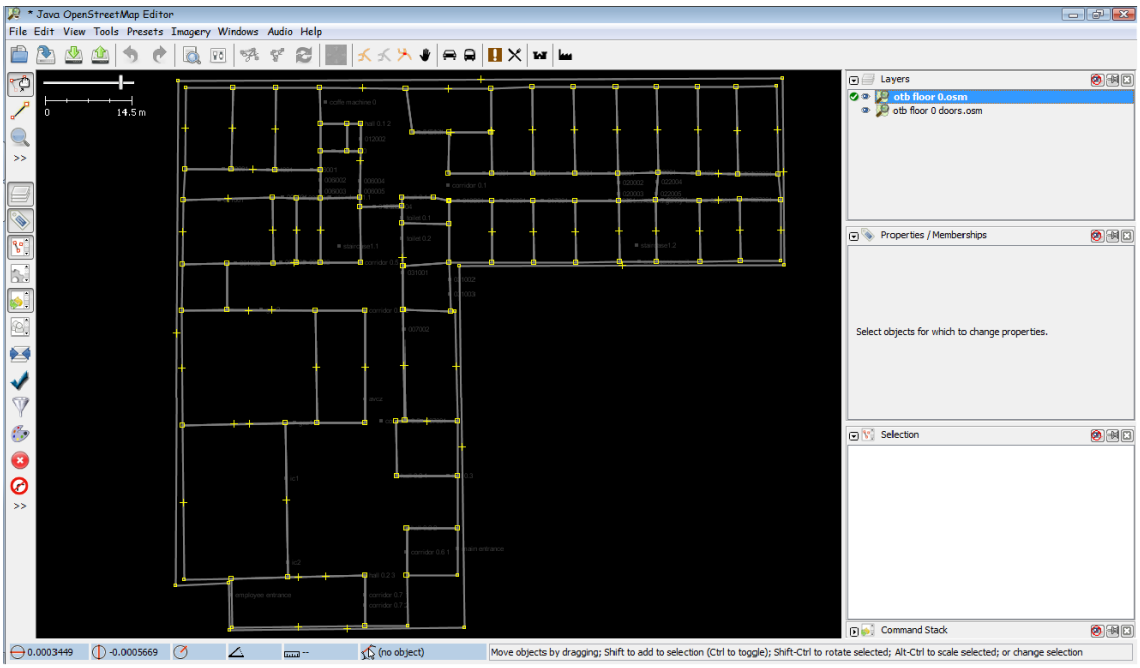


Figure 4.1: *JOSM editor with floor 0 geometrical data with door and spaces.*

package[4]. After that we implemented also a function for dataset export into XML format, since our data persistence system is based on xml plain text file (without using Spatial database due to time reason and out of scope according to the route direction generation's aim of this thesis). In Appendix D you can find a portion of the generated dataset in Code snippet 7. The complete dataset consists of 150 semantic navigable spaces, 200 nodes and 1400 edges (file size: 326 KB) and it's not fully shown in this thesis due to space reasons.
Using this dataset file as input for the route generation process we can now introduce in next section our tests conducted.

4.2 Tests

With the dataset of the building we performed a two-part study based on four different routes, trying to answer the question: are the route directions generated by IndoorNav understandable, allowing someone to find his way? The tests were

[4]https://github.com/zxing/zxing

conducted involving five people of the OTB staff (PhDs and students), submitting them the tests in Appendix D printed on different papers. We provided them an Android Smartphone[5] with running software IndoorNav for route guidance. We provided a printed QRCode library of all nodes in the building imagining that the codes were printed in front of the respective doors. To start each test the user was led to the starting point and invited to follow route test paper instructions by scanning the QRCode as starting point and filling the ending desired point's field as pointed out in the test. After that the users were not helped during travels, our collaborator followed from the back the users while performing the test annoting all the instructions(floors, rooms and single-edge instructions) needed to reach the destination and the number of position updates, after losing direction.

All the routes instructions start with the floor directions, so every test used all the floor instructions. At first, it's interesting to analyze how many floor directions have been sufficient to guide the user to the next decision point, and how many of them required detailed rooms instruction to better understand the direction to follow. Secondly we want to collect also how many room directions covered the desired additional route information and how many of them required also single edge direction to reach the next decision node.The testing route are resumed below in Table 5.1.

Route test	*Starting Node*	*Ending Node*	*Elevator usage*	*N floor instructions*	*N of room instructions*	*N of singleedge instr.*
N 1	Main entrance	Room 1.240	NO	4	10	10
N 2	Room 1.240	Room 2.120	NO	3	6	6
N 3	Room 2.170	hidden	NO	3	6	6
N 4	Room 3.240	hidden	NO	3	5	5

Table 4.1: *Routes resume*

[5]HTC Desire C

The first part of the study consists of two routes that roundly show the destination room's name, so that the users were helped in the last part of the travel by finding the room name on the doors. In the second part we provided two routes in which the ending room names were replaced by the room ID, so that the ending rooms were unknown and users needed more accuracy to find exactly the ending room. In the following we analyze only the first Route, the next three can be founded in the Appendix D.

4.2.1 Test 1

To explain the first test, we start showing the paper given to the user as guideline for the test n1(Figure 4.2). the route to follow is explained in details in the following Figure 4.4. Black arrows highlight the resumed path by Floor Instructions, the red arrows resume the path of each Room Instruction, the blue arrows are the real path resumed by Single Edge Instructions.

Figure 4.2: *User test n 1*

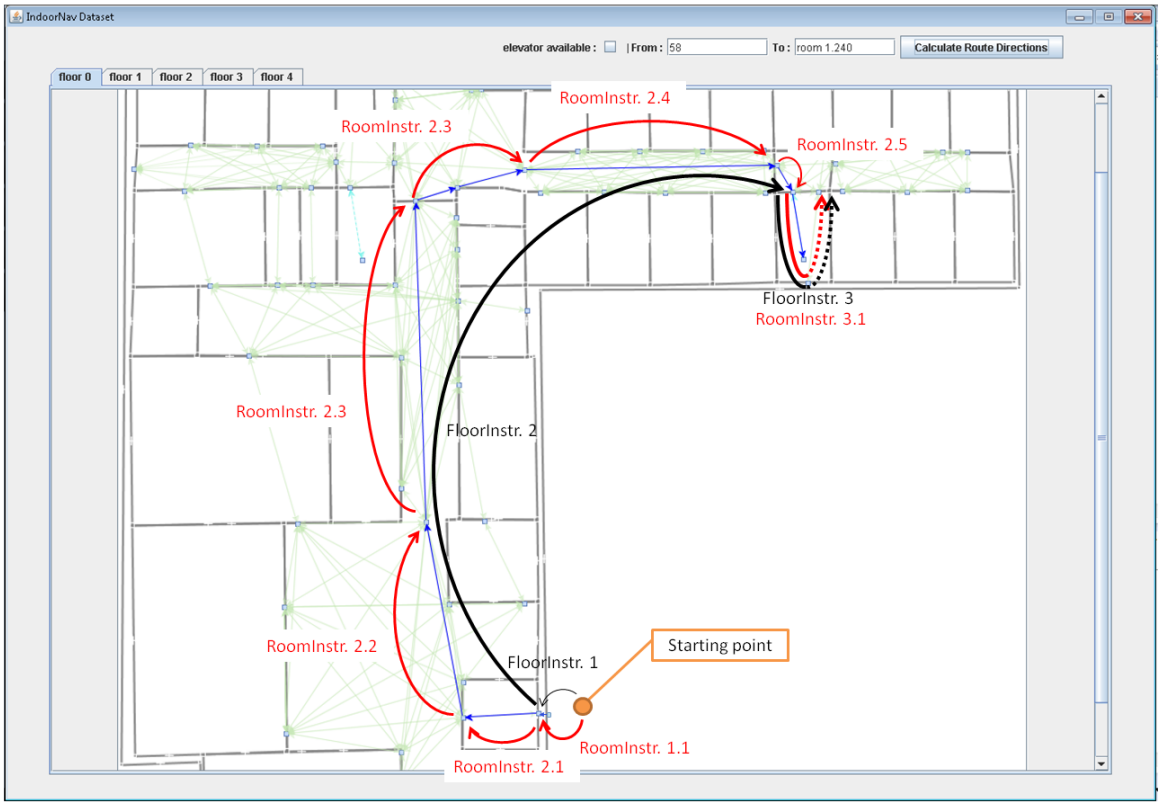


Figure 4.3: *User test 1, ground floor and part of the staircase planned path(singleEdges in BLUE), with floor Instructions(BLACK) and room Instructions(RED)*

Figure 4.4: *User test 1, first floor and part of the staircase planned path(singleEdges in BLUE), with floor Instructions(BLACK) and room Instructions(RED)*

The output of the Natural Language generation process applied to generated directions is resumed in the following table 5.2 .

Direction Level	**Literal Instruction**
Floor Instruction 1	Starting from you position in front of the QRCode, Enter into the QRCode's door
Room Instruction 1.1	Enter into the QRCode's door
SingleEdge Instruction 1.1.1	Enter into the Door of the ground floor
Floor Instruction 2	Start walking going straight on and reach the Stairs north
Room Instruction 2.1	Go Straight on, walk for 4 meters, pass through the corridor 0.6 and reach the hall 0.2
SingleEdge Instruction 2.1.1	Go Straight on, walk for 4 meters and reach the Corridor 0.6 door
Room Instruction 2.2	Turn Right, walk for 11 meters, pass through the hall 0.2 and reach the corridor 0.5
SingleEdge Instruction 2.2.1	Turn Right, walk for 11 meters and reach the Corridor 0.5 access
Chunking Instructions 2.2.1	Next node is the first visible corridor from your Right side
Chunking Instructions 2.2.1	Next node is the first visible corridor from your Left side
Chunking Instructions 2.2.1	Next node is after 4 visible Door counting from Left side
Room Instruction 2.3	Go Straight on, walk for 18 meters, pass through the corridor 0.5 and reach the hall 0.1
SingleEdge Instruction 2.3.1	Go Straight on, walk for 18 meters and reach the Door hall 0.1
Chunking Instructions 2.3.1	Next node has the Door toilet 0.1 on the Right side
Chunking Instructions 2.3.1	Next node is after 1 visible Door counting from Left side
Chunking Instructions 2.3.1	Next node is after 4 visible Door counting from Right side

Continued on Next Page...

Direction Level	**Literal Instruction**
Room Instruction 2.4	Turn Right, walk for 6 meters, pass through the hall 0.1 and reach the corridor 0.1
SingleEdge Instruction 2.4.1	Turn Right, walk for 2 meters and reach the corner
Chunking Instructions 2.4.1	Next node has the Corridor Door on the Left side
Chunking Instructions 2.4.1	Next node is the first visible SingleCorner from your Right side
Chunking Instructions 2.4.1	Next node is the first visible SingleCorner from your Left side
Chunking Instructions 2.4.1	Next node is after 3 visible Door counting from Left side
SingleEdge Instruction 2.4.2	Turn Right, walk for 3 meters and reach the Corridor 0.1 access
Chunking Instructions 2.4.2	Next node is the first visible corridor from your Right side
Chunking Instructions 2.4.2	Next node is the first visible corridor from your Left side
Chunking Instructions 2.4.2	Next node is after 1 visible Door counting from Right side
Chunking Instructions 2.4.2	Next node is after 3 visible Door counting from Left side
Room Instruction 2.5	Go Straight on, walk for 14 meters, pass through the corridor 0.1 and reach the corridor 0.3
SingleEdge Instruction 2.5.1	Go Straight on, walk for 14 meters and reach the Corridor Door
Chunking Instructions 2.5.1	Next node has the Door 0.200 on the Left side
Chunking Instructions 2.5.1	Next node has the Door 0.190 on the Right side
Chunking Instructions 2.5.1	Next node is after 4 visible Door counting from Right side
Chunking Instructions 2.5.1	Next node is after 4 visible Door counting from Left side
Room Instruction 2.6	Veer Right, walk for 1 meters, pass through the corridor 0.3 and reach the staircase 1.2

Continued on Next Page...

Direction Level	**Literal Instruction**
SingleEdge Instruction 2.6.1	Veer Right, walk for 1 meters and reach the stairs access 1.2
Chunking Instructions 2.6.1	Next node has the emergency exit on the Left side
Chunking Instructions 2.6.1	Next node is the first visible corridor from your Right side
Chunking Instructions 2.6.1	Next node is after 1 visible corridors counting from Left side
Chunking Instructions 2.6.1	Next node is after 2 visible Door counting from Left side
Floor Instruction 3	Take the stairs, go Upstairs and reach the floor 1
Room Instruction 3.1	Take the stairs, go Upstairs, walk for 8 meters, pass through the staircase 1.2 and reach the corridor 1.3
SingleEdge Instruction 3.1.1	Go Upstairs, walk for 4 meters and reach the corner.
Floor Instruction 4	Start walking veering Right and reach the room 1.240
Room Instruction 4.1	Veer Right, walk for 1 meters, pass through the corridor 1.3 and reach the corridor 1.2
SingleEdge Instruction 4.1.1	Veer Right, walk for 1 meters and reach the Corridor Door
Chunking Instructions 4.1.1	Next node has the Door 1.220 on the Left side
Chunking Instructions 4.1.1	Next node is the first visible Door from your Right side
Chunking Instructions 4.1.1	Next node is after 2 visible Door counting from Left side
Room Instruction 4.2	Veer Left, walk for 2 meters, pass through the corridor 1.2 and reach the room 1.240
SingleEdge Instruction 4.2.1	Veer Left, walk for 2 meters and reach the Door 1.240
Chunking Instructions 4.2.1	Next node has the Door 1.250 on the Right side

Continued on Next Page...

Direction Level	**Literal Instruction**
Chunking Instructions 4.2.1	Next node is the first visible Door from your Left side
Chunking Instructions 4.2.1	Next node is after 5 visible Door counting from Right side

Table 4.3: *Test Route 1 directions resume*

4.3 Results

The analysis of the results must take into account the fact that we test not only evaluate the generated route directions but also Natural Language translation and visual prototype interface, so it's affected to errors not related to the directions but to the testing prototype. The testing results are resumed below in Table 5.2. All the test had a positive outcome. Floor directions are the most useful for textual guidance(60%), particularly in vertical movements (floor changes) in which no one needed other details(100%). In spaces with non convex shapes, room-instructions are frequently used(total percentage 25.5%). Users also lose orientation in floor directions due to not clear first direction to follow, and needed to update position scanning a near QRCode in 20% of route tests. As expected in the second part of the tests, user needed more accurate instruction due to destination room's name hidden(Single edge instruction usage has risen from 10% to 30%).

Route number	*N Floor instr. per route*	*Just Floor*	*Floor + Room*	*Floor + Room + SingleEdge*	*Updated position per user*
N 1	4	12/20	7/20	1/20	0/5
N 2	3	10/15	4/15	1/15	1/5
Summary part 1		63%	31%	6%	10%
N 3	3	10/15	3/15	2/15	1/5
N 4	3	7/15	3/15	5/15	2/5
Summary part 2		57%	20%	23%	30%

Table 4.4: *Tests resume of Instructions usage (all users data together) referring to number of floor instructions per route*

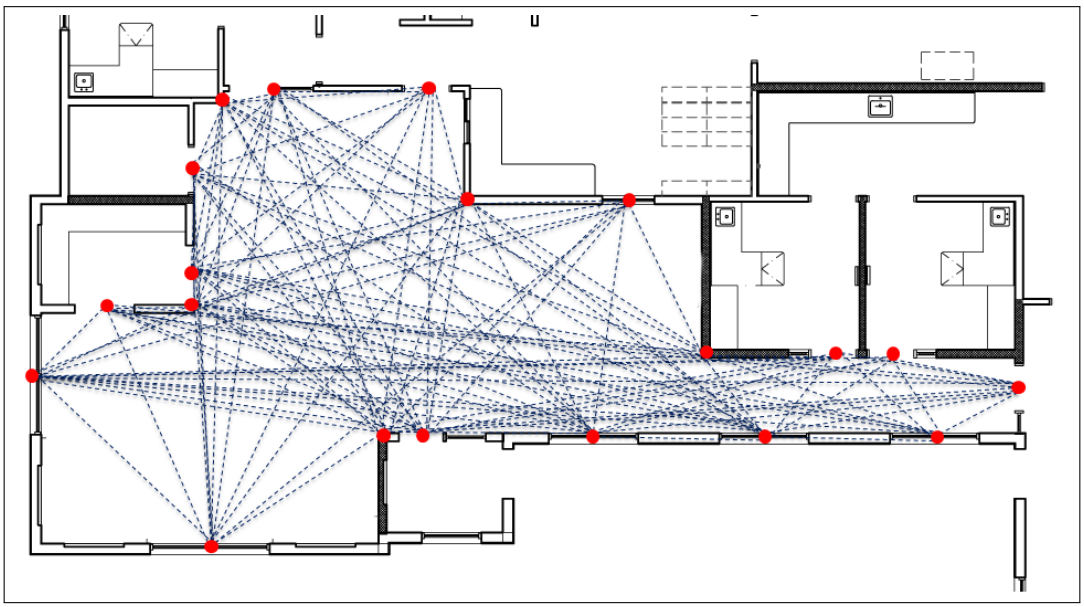
Chapter 5

Conclusions and Future work

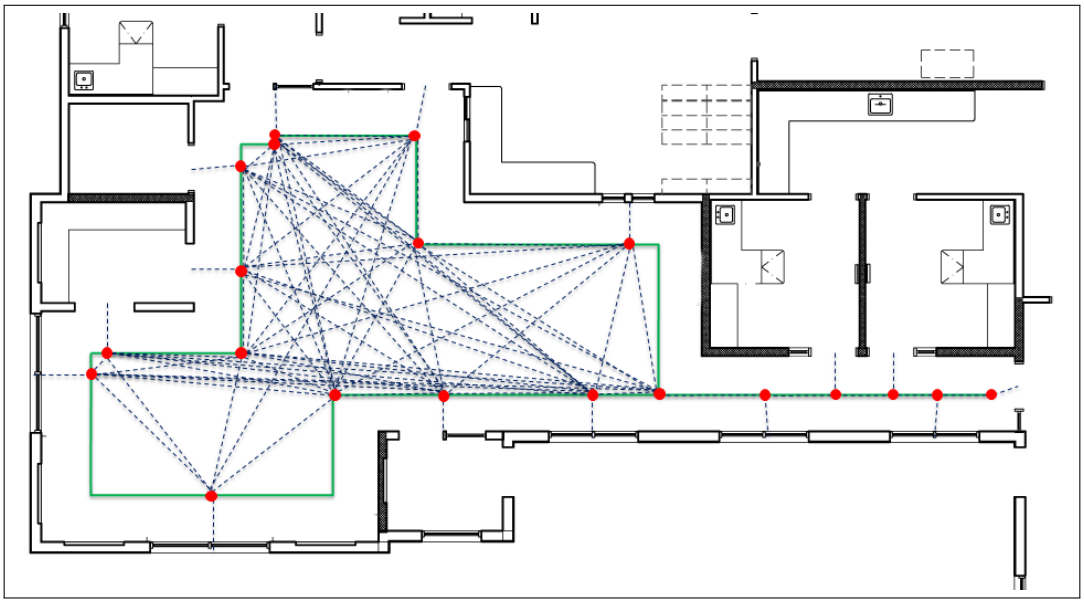
In this work we concerned with automatic generation of route directions, the proposed and implemented solution is able to produce route directions as has been demonstrated in the evaluation tests. Due to the modularity of the generation process, with low coupling between dataset,route generation and presentation form, this work may well serve as a test-bed for further empirical studies about Language generation, dataset refinement or route directions comparison. In the next paragraphs we'll analyze what's good or bad in the proposed solution and we'll denote future improvements to solve them.

This is a final comparison between our IndoorNav prototype and the initial objectives of the thesis,to analyze and propose improvements.

- **Open spaces modeling**: the choice of visibility approach solved most of the problems of data modeling, but according to Stahl et Al. research [Sta08] we can improve our model by adding a buffer(for instance 0.50m) to all the wall of the rooms, so as graph's edges aren't in certain cases overlapped to the walls/corners, but maintain an appropriate distance from them as human movements. With this solution also narrow corridors could be better modeled with an hybrid approach similar to MAT as shown in Figure 5.1.
- **Route planner**: whit the Shortest path we solved this objective but we could improve this system with kshortest path usage to generate more than one feasible path and use the output to compute the simplest path by using a weighting function that can be a mix of: Grum's approach [Gru05], calculating



(a)



(b)

Figure 5.1: *Comparison of visibility approach(a) and an hybrid approach(b).*[1]

the percentage of possible user errors in each decision node; and Duckham's one [DWR10], based on landmark's saliency using Raubal Winter framework [RW02].

- **Positioning system**: the objective of a positioning system with low in-

frastructural costs is achieved, but we should integrate the static positioning system (QRcodes) with some dynamic system for a more accurated guidance while traveling. The ideas are two: at first we can improve Android Application using built-in accelerometer and gyroscope for an estimation of the position, secondly we can introduce other direct sense positioning systems in some crucial points(previously decided by the software management) in order to catch the position while traveling and update the directions if needed.

- **Landmarks usage**: fully integrated in the route directions, an improvement could be the careful selection of the landmarks to use within a route direction without using all the visible objects, deepening selection policies.

Since no commercial indoor navigation system has been released yet, this thesis certainly is a good contribution to the scientific community. We hope that this work is a concrete step in the direction of autonomous indoor guidance systems.

ACKNOWLEDGMENTS

This master thesis research has been done at the OTB research Institute of TUDelft (The Netherlands), from February,2013 to August,2013 with the supervisory of Prof. Zlatanova Sisi. I'd like to thank you for all the support given to me in these six months with advice and encouragement, and for all the time spent on meeting and papers reviews. I'd like also to thank Prof Van Oosterom P., Mrs Fendel E. and all OTB staff and PhDs for the great hospitality shown. I would like to thank Professor Clementini Eliseo and University of L'Aquila that has given to me the possibility to make an international experience of research traineeship in one of the best universities in the world. A special thanks to my work colleague and travel companion in this experience, Mortari Filippo, and all the others University colleagues of Engineering Faculty in L'Aquila(Italy). Finally, thanks to my family and all my friends who supported me throughout the academic career!

This page intentionally left blank.

Bibliography

[Aeb12] Frric Aebi. Autonomous indoor navigation - implementation of an autonomous indoor navigation system on android. Master's thesis, University of Fribourg (Switzerland), 2012.

[All97] Gary L. Allen. From knowledge to words to wayfinding: Issues in the production and comprehension of route directions. In *Proceedings of the International Conference on Spatial Information Theory: A Theoretical Basis for GIS*, COSIT '97, pages 363–372, London, UK, UK, 1997. Springer-Verlag.

[All99] Gary L Allen. Spatial abilities, cognitive maps, and wayfinding. *Wayfinding behavior: Cognitive mapping and other spatial processes*, pages 46–80, 1999.

[All00] Gary L Allen. Principles and practices for communicating route knowledge. *Applied Cognitive Psychology*, 14(4):333–359, 2000.

[APBC08] S Aparicio, J Perez, AM Bernardos, and JR Casar. A fusion method based on bluetooth and wlan technologies for indoor location. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 487–491. IEEE, 2008.

[AW92] Anthony J. Aretz and Christopher D. Wickens. The mental rotation of map displays. *Human Performance*, 5(4):303–328, 1992.

[B+67] Harry Blum et al. A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form*, 19(5):362–380, 1967.

[BCK05] Jurg Baus, Keith Cheverst, and Christian Kray. A survey of map-based mobile guides. In *Map-based Mobile Services*, pages 193–209. Springer,

2005.

[BKW02] Jrg Baus, Antonio Krger, and Wolfgang Wahlster. A resource-adaptive mobile navigation system. In *IUI*, pages 15–22, 2002.

[CBM09] E Chan, George Baciu, and SC Mak. Using wi-fi signal strength to localize in wireless sensor networks. In *Communications and Mobile Computing, 2009. CMC'09. WRI International Conference on*, volume 1, pages 538–542. IEEE, 2009.

[CDR+10] Heriberto Cuayáhuitl, Nina Dethlefs, Kai-Florian Richter, Thora Tenbrink, and John Bateman. A dialogue system for indoor wayfinding using text-based natural language. *International Journal of Computational Linguistics and Applications*, 1(1-2):285–304, 2010.

[CKK95] Eric Chown, Stephen Kaplan, and David Kortenkamp. Prototypes, location, and associative networks (plan): Towards a unified theory of cognitive mapping. *Cognitive Science*, 19(1):1–51, 1995.

[CTW08] Yao-Jen Chang, Shih-Kai Tsai, and Tsen-Yung Wang. A context aware handheld wayfinding system for individuals with cognitive impairments. In Simon Harper and Armando Barreto, editors, *ASSETS*, pages 27–34. ACM, 2008.

[DD98] Marie-Paule Daniel and Michel Denis. Spatial descriptions as navigational aids: A cognitive analysis of route directions wegausknfte als navigationshilfen - eine kognitive analyse. *Kognitionswissenschaft*, 7(1):45–52, 1998.

[DGP02] Robert Dale, Sabine Geldof, and Jean-Philippe Prost. Generating more natural route descriptions. In *Proceedings of the 2002 Australasian natural language processing workshop*, pages 41–48, 2002.

[DGP03] Robert Dale, Sabine Geldof, and Jean-Philippe Prost. Coral: Using natural language generation for navigational assistance. In *Proceedings of the 26th Australasian computer science conference-Volume 16*, pages 35–44. Australian Computer Society, Inc., 2003.

[DK03] Matt Duckham and Lars Kulik. ŝimplestṗaths: Automated route selection for navigation. In Werner Kuhn, Michael F. Worboys, and Sabine Timpf, editors, *COSIT*, volume 2825 of *Lecture Notes in Computer Science*, pages 169–185. Springer, 2003.

[DWR10] Matt Duckham, Stephan Winter, and Michelle Robinson. Including

landmarks in routing instructions. *J. Location Based Services*, 4(1):28–52, 2010.

[DYZJ07] Bin Ding, Haitao Yuan, Xiaoning Zang, and Li Jiang. The research on blind navigation system based on rfid. In *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 2058–2061. IEEE, 2007.

[FABF13] Navid Fallah, Ilias Apostolopoulos, Kostas Bekris, and Eelke Folmer. Indoor human navigation systems: A survey. *Interacting with Computers*, 25(1):21–33, 2013.

[FMG+90] SM Freundschuh, DM Mark, S Gopal, MD Gould, and H Couclelis. Verbal directions for wayfinding: Implications for navigation and geographic information and analysis systems. In *4th International Symposium on Spatial Data Handling, Brassel, K. and Kishimoto, H., Eds. Zurich: Department of Geography, University of Zurich*, pages 478–487, 1990.

[Fre91] Christian Freksa. Qualitative spatial reasoning. *Cognitive and linguistic aspects of geographic space*, (63):361–372, 1991.

[GD02] Sabine Geldof and Robert Dale. Improving route directions on mobile devices. In *ISCA Tutorial and Research Workshop (ITRW) on Multi-Modal Dialogue in Mobile Environments*, 2002.

[Gol99] Reginald G Golledge. Human wayfinding and cognitive maps. *Wayfinding behavior: Cognitive mapping and other spatial processes*, pages 5–45, 1999.

[Gru05] Eva Grum. Danger of getting lost: Optimize a path to minimize risk. In *10th International Conference onUrban Planning & Regional Development in the Information Society (CORP 2005), Vienna, Austria*, 2005.

[HE04] Mark Hampe and Birgit Elias. Integrating topographic information and landmarks for mobile navigation. 2004.

[HG10] Haosheng Huang and Georg Gartner. A survey of mobile indoor navigation systems. In *Cartography in Central and Eastern Europe*, pages 305–319. Springer, 2010.

[KDK+04] Alexander Klippel, Carsten Dewey, Markus Knauff, Kai-Florian Richter, Dan R Montello, Christian Freksa, and Esther-Anna Loeliger.

Direction concepts in wayfinding assistance systems. In *Workshop on Artificial Intelligence in Mobile Systems*, pages 1–8, 2004.

[KH06] Krzysztof Kolodziej and Johan Hjelm. *Local Positioning Systems: LBS Applications and Services*. CRC Press, erste edition, Mai 2006.

[KHRW09] Alexander Klippel, Stefan Hansen, Kai-Florian Richter, and Stephan Winter. Urban granularities - a data structure for cognitively ergonomic route directions. *GeoInformatica*, 13(2):223–247, 2009.

[KMB+04] Benjamin Kuipers, Joseph Modayil, Patrick Beeson, Matt MacMahon, and Francesco Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 4845–4851. IEEE, 2004.

[KRH09] Alexander Klippel, Kai-Florian Richter, and Stefan Hansen. Cognitively ergonomic route directions. *Handbook of Research on Geoinformatics*, pages 230–238, 2009.

[KTC09] Tai-Wei Kan, Chin-Hung Teng, and Wen-Shou Chou. Applying qr code in augmented reality applications. In Stephen N. Spencer, Masayuki Nakajima, Enhua Wu, Kazunori Miyata, Daniel Thalmann, and Zhiyong Huang, editors, *VRCAI*, pages 253–257. ACM, 2009.

[LHM99] Kristin L. Lovelace, Mary Hegarty, and Daniel R. Montello. Elements of good route directions in familiar and unfamiliar environments. In Christian Freksa and David M. Mark, editors, *COSIT*, volume 1661 of *Lecture Notes in Computer Science*, pages 65–82. Springer, 1999.

[LPW79] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, October 1979.

[Lyn60] Kevin Lynch. *The image of the city*, volume 11. the MIT Press, 1960.

[LZ11] L Liu and S Zlatanova. A" door-to-door" path-finding approach for indoor navigation. In *Proceedings of GeoInformation For Disaster Management Conference 2011*, pages 3–8, 2011.

[Mac] Matt MacMahon. Handling errors and omissions in route instructions.

[Mar86] David M Mark. Automated route selection for navigation. *Aerospace and Electronic Systems Magazine, IEEE*, 1(9):2–5, 1986.

[MHC08] Inderjeet Mani, Janet Hitzeman, and Cheryl Clark. Annotating natural

language geographic references. In *proc. LREC 2008-W13 Workshop on Methodologies and Resources for Processing Spatial Language*, pages 11–15. Citeseer, 2008.

[Mon05] Daniel R. Montello. Navigation. In Cambridge University, editor, *The Cambridge Handbook of Visuospatial thinking*, pages 257–294. Cambridge University Press, 2005.

[MS06] Daniel R Montello and Corina Sas. Human factors of wayfinding in navigation. 2006.

[NKB+09] Emmanouil Nikoloudakis, Manolis Kritsotakis, Antonis Bikakis, Theodore Patkos, Grigoris Antoniou, and Dimitris Plexousakis. Exploiting semantics for indoor navigation and user-tracking. In *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC'09. Symposia and Workshops on*, pages 149–154. IEEE, 2009.

[PJS+96] Helen Petrie, Valerie Johnson, Thomas Strothotte, Andreas Raab, Steffi Fritz, and Rainer Michel. Mobic: Designing a travel aid for blind and elderly people. *Journal of Navigation*, 49(01):45–52, 1996.

[Rad03] Verena Radoczky. *Kartographische Unterstützungsmöglichkeiten zur Routenbeschreibung*. 2003.

[Rad07] Verena Radoczky. How to design a pedestrian navigation system for indoor and outdoor environments. In Georg Gartner, William E. Cartwright, and Michael P. Peterson, editors, *Location Based Services and TeleCartography*, Lecture Notes in Geoinformation and Cartography, pages 301–316. Springer, 2007.

[Red99] A David Redish. *Beyond the cognitive map [electronic resource]: from place cells to episodic memory*. MIT Press, 1999.

[Ret04] Gunther Retscher. Pedestrian navigation systems and location-based services. In *3G Mobile Communication Technologies, 2004. 3G 2004. Fifth IEE International Conference on*, pages 359–363. IET, 2004.

[RHM04] Lisa Ran, Sumi Helal, and Steve Moore. Drishti: An integrated indoor/outdoor blind navigation system and service. In *PerCom*, pages 23–32. IEEE Computer Society, 2004.

[Ric07] Kai-Florian Richter. A uniform handling of different landmark types in route directions. In Stephan Winter, Matt Duckham, Lars Kulik, and Benjamin Kuipers, editors, *COSIT*, volume 4736 of *Lecture Notes in*

Computer Science, pages 373–389. Springer, 2007.

[Ric08] Kai-Florian Richter. *Context-specific route directions: generation of cognitively motivated wayfinding instructions.* PhD thesis, University of Bremen, 2008. http://d-nb.info/987640062.

[Ric13] Kai-Florian Richter. Prospects and challenges of landmarks in navigation services. In *Cognitive and Linguistic Aspects of Geographic Space*, pages 83–97. Springer, 2013.

[RK04] Kai-Florian Richter and Alexander Klippel. A model for context-specific route directions. In Christian Freksa, Markus Knauff, Bernd Krieg-Brckner, Bernhard Nebel, and Thomas Barkowsky, editors, *Spatial Cognition*, volume 3343 of *Lecture Notes in Computer Science*, pages 58–78. Springer, 2004.

[RMT04] Tracy Ross, Andrew J. May, and Simon Thompson. The use of landmarks in pedestrian navigation instructions and the effects of context. In Stephen A. Brewster and Mark D. Dunlop, editors, *Mobile HCI*, volume 3160 of *Lecture Notes in Computer Science*, pages 300–304. Springer, 2004.

[RW02] Martin Raubal and Stephan Winter. Enriching wayfinding instructions with local landmarks. In Max J. Egenhofer and David M. Mark, editors, *GIScience*, volume 2478 of *Lecture Notes in Computer Science*, pages 243–259. Springer, 2002.

[SH99] Molly E. Sorrows and Stephen C. Hirtle. The nature of landmarks for real and electronic spaces. In Christian Freksa and David M. Mark, editors, *COSIT*, volume 1661 of *Lecture Notes in Computer Science*, pages 37–50. Springer, 1999.

[SSO08] Edgar-Philipp Stoffel, Korbinian Schoder, and Hans Jrgen Ohlbach. Applying hierarchical graphs to pedestrian indoor navigation. In Walid G. Aref, Mohamed F. Mokbel, and Markus Schneider, editors, *GIS*, page 54. ACM, 2008.

[Sta08] Christoph Stahl. New perspectives on built environment models for pedestrian navigation. *Spatial Cognition 2008 Poster Proceedings*, pages 016–08, 2008.

[Ste13] Horst Steuer. High precision 3d indoor routing on reduced visibility graphs. In *Progress in Location-Based Services*, pages 265–275.

Springer, 2013.

[Sto09] Edgar-Philipp Stoffel. *Hierarchical graphs as organisational principle and spatial model applied to pedestrian indoor navigation.* PhD thesis, Ludwig Maximilians University Munich, 2009. http://d-nb.info/100055080X.

[SW75] Alexander W Siegel and Sheldon H White. The development of spatial representations of large-scale environments. *Advances in child development and behavior*, 10:9–55, 1975.

[TAK+05] Vassileios Tsetsos, Christos Anagnostopoulos, Panayiotis Kikiras, P Hasiotis, and Stathes Hadjiefthymiades. A human-centered semantic navigation system for indoor environments. In *Pervasive Services, 2005. ICPS'05. Proceedings. International Conference on*, pages 146–155. IEEE, 2005.

[TAKH06] Vassileios Tsetsos, Christos Anagnostopoulos, Panayotis Kikiras, and Stathes Hadjiefthymiades. Semantically enriched navigation for indoor environments. *IJWGS*, 2(4):453–478, 2006.

[WH05] Scooter Willis and Sumi Helal. Rfid information grid and wearable computing solution to the problem of wayfinding for the blind user in a campus environment. In *Proceedings of the ninth annual IEEE International Symposium on Wearable Computers, Osaka, Japan.* Citeseer, 2005.

[Yen71] J.Y. Yen. Finding the k shortest loopless paths in a network. *management Science*, pages 712–716, 1971.

Appendix A

Appendix A - UML diagrams

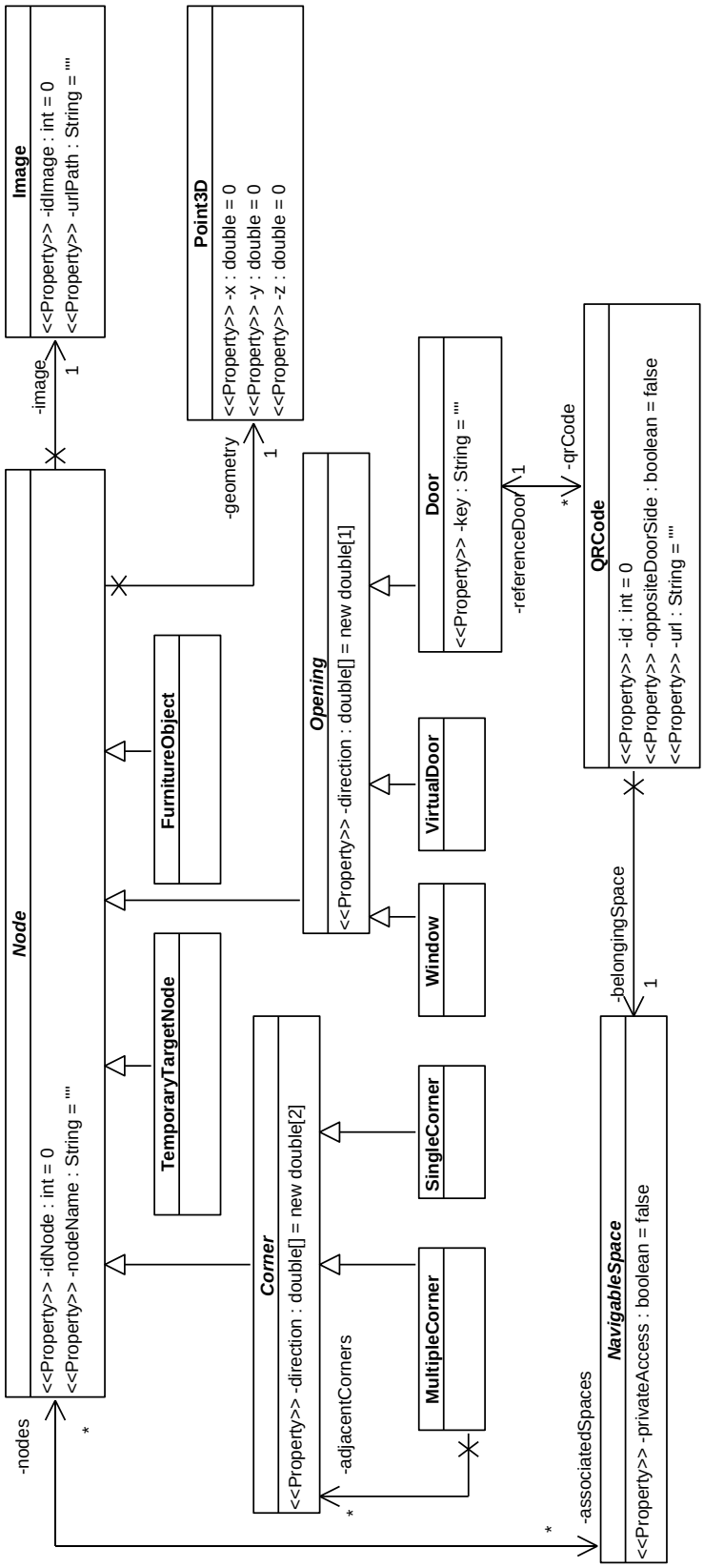


Figure A.1: *UML class diagram of domain-model for graph nodes*

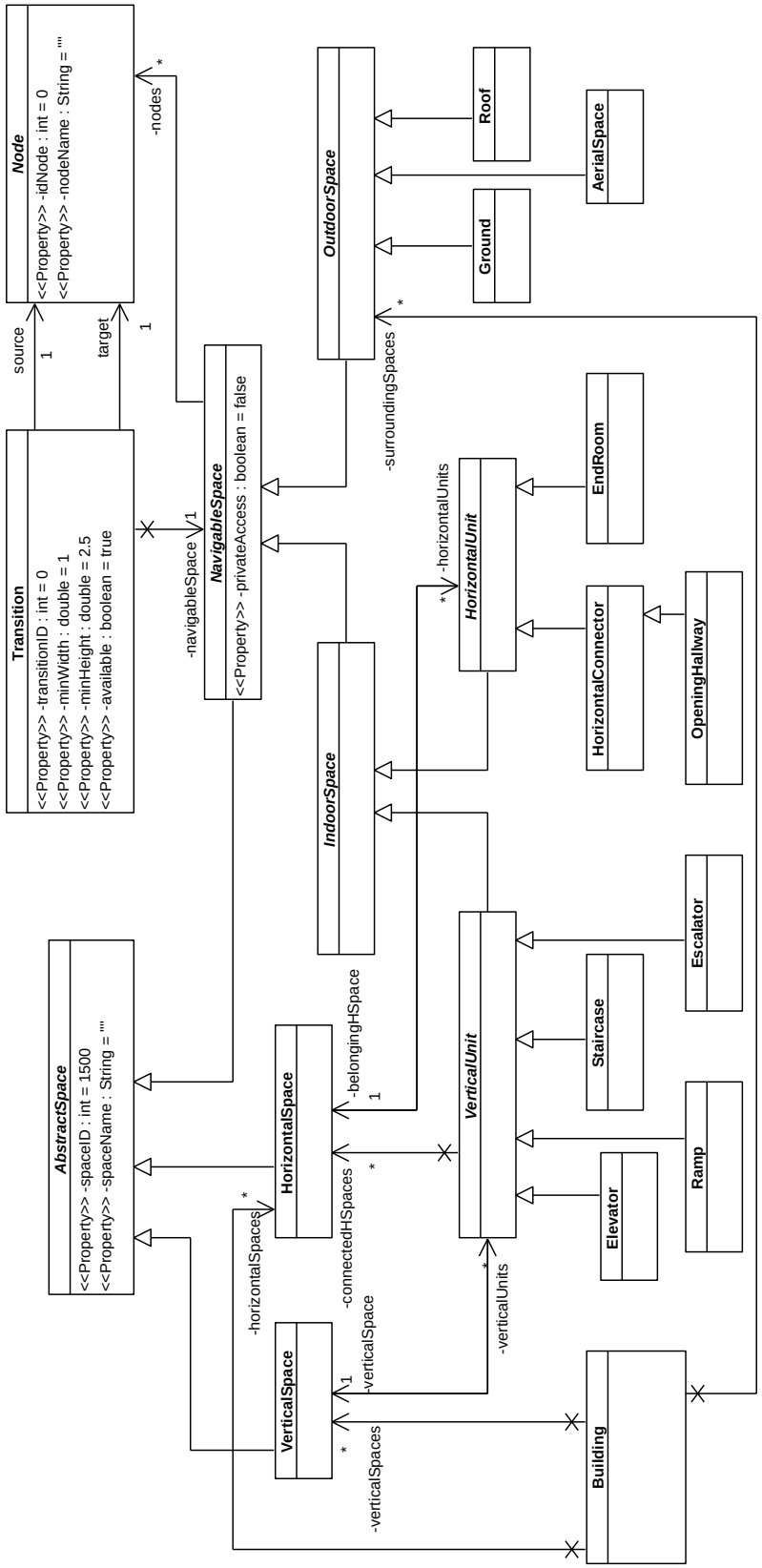


Figure A.2: *UML class diagram of domain-model for semantic spaces*

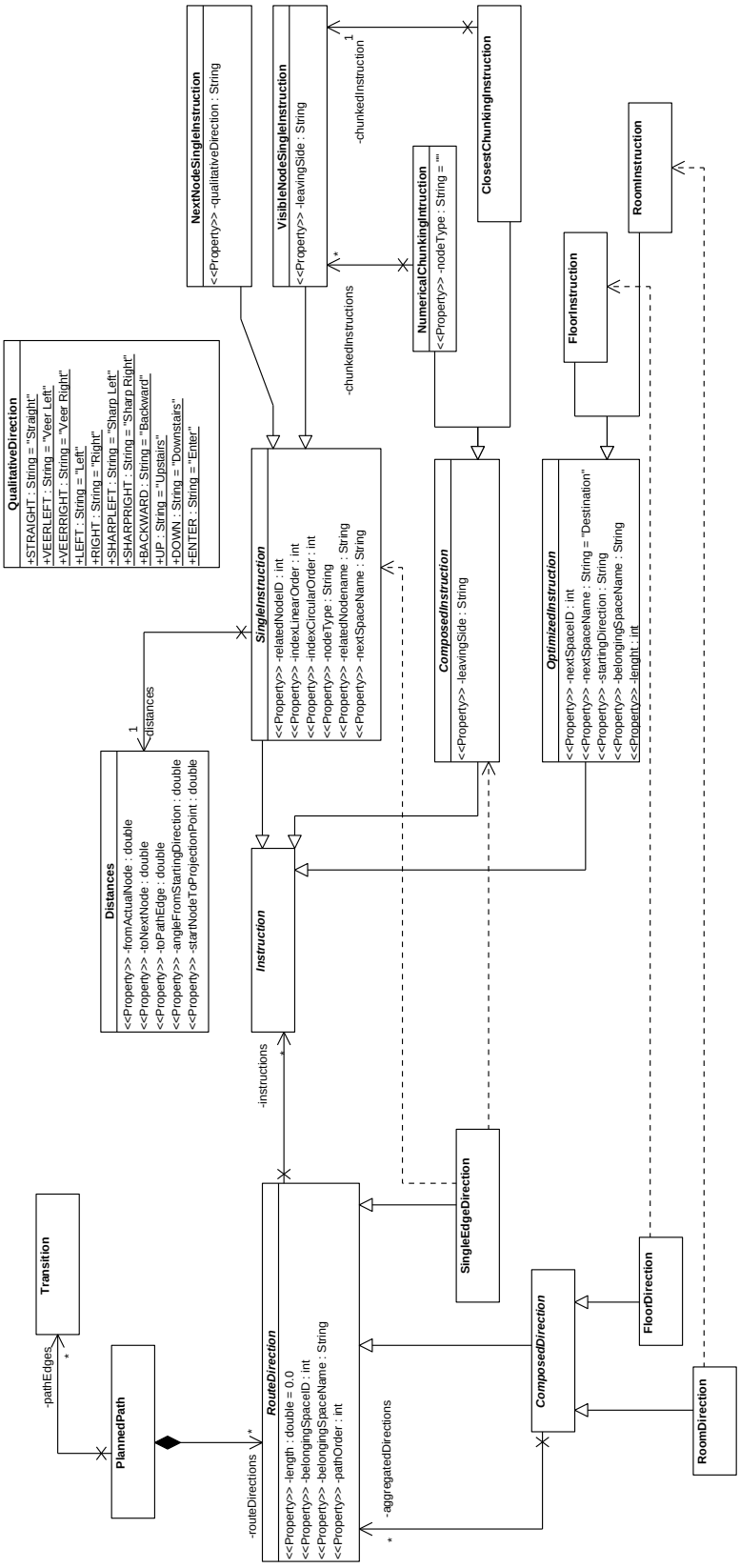


Figure A.3: *UML class diagram of domain-model for Route Instructions*

Appendix B

Appendix B - Pseudo Algorithms

Algorithm 2: *Extraction of Route Instructions*

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  Function: extractSingleInstruction
3  Input: graph, plannedPath
4  Output: routeDirections (Collection of ↩
       SingleEdgeDirections)
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7  extractSingleInstruction{
8    create routeDirections collection;
9    foreach (pathedge in plannedPath){
10     create SingleEdgeDirection and add it to ↩
         routeDirections;
11     store pathedge's NavigableSpace into ↩
         SingleEdgeDirection;
12     create Instruction collection into ↩
         SingleEdgeDirection;
13     create NextNodeSingleInstruction and add it to ↩
         Instruction;
14     store ending node of pathedge in ↩
         NextNodeSingleInstruction;
15     compute QualitativeDirection of pathedge and store it ↩
         into NextNodeSingleInstruction;
16     foreach (visibleNode from pathedge.startingNode ↩
         belonging to the same pathedge.NavigableSpace) {
17       create VisibleNodeSingleInstruction and add it to ↩
           Instruction;
18       store visibleNode in VisibleNodeSingleInstruction;
19       store qualitative belonging leavingSide of ↩
           visibleNode - referring to pathedge - into ↩
           VisibleNodeSingleInstruction;
20     }
21   }
22   return routeDirections;
23 }
```

Algorithm 3: *Closest Chunking of Route Instructions*

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  Function: closestChunking
3  Input: graph, routeDirections (Collection of ↩
       SingleEdgeDirections), radius
4  Output: routeDirections (Collection of ↩
       SingleEdgeDirections)
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7  closestChunking{
8    foreach (SingleEdgeDirection in routeDirections){
9      analyze all VisibleNodeSingleInstruction of ↩
         SingleEdgeDirection;
10     divide them into qualitative belonging space: Left and↩
         Right as two different sets;
11     foreach not empty set {
12       take the VisibleNodeSingleInstruction with minimum ↩
           angle value ignoring Node types "Corner";
13       if (VisibleNodeSingleInstruction angle is less than ↩
           radius && ){
14         create ClosestChunkingInstruction and store it ↩
             into Instruction collection of ↩
             SingleEdgeDirection;
15         store VisibleNodeSingleInstruction into ↩
             ClosestChunkingInstruction;
16       }
17     }
18   }
19   return routeDirections;
20 }
```

Algorithm 4: *Numerical Chunking of Route Instructions*

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  Function: numericalChunking
3  Input: graph, routeDirections (Collection of SingleEdgeDirections)
4  Output: routeDirections (Collection of SingleEdgeDirections)
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7  numericalChunking{
8    foreach (SingleEdgeDirection in routeDirections){
9      analyze all VisibleNodeSingleInstruction of SingleEdgeDirection;
10     divide them into qualitative belonging space: Left and Right as two different sets;
11     foreach not empty set {
12       divide set into NodeType creating subsets;
13       foreach subset{
14         create NumericalChunkingInstruction into Instructions Collection of SingleEdgeDirection;
15         store into NumericalChunkingInstruction all VisibleNodeSingleInstruction of the subset;
16         store into NumericalChunkingInstruction the number and nodetype elements of subset;
17       }
18     }
19     remove all VisibleNodeSingleInstruction from Instruction collection of SingleEdgeDirection;
20   }
21   return routeDirections;
22 }
```

Algorithm 5: *Room Segmentation of Route Instructions*

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  Function: roomSegmentation
3  Input: graph, routeDirections (Collection of ↩
       SingleEdgeDirections)
4  Output: routeDirections (Collection of RoomDirections)
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7  roomSegmentation{
8    create previousNavigableSpace variable;
9    create currentRoomDirection variable;
10   create newRouteDirections Collection of RoomDirections;
11   foreach (SingleEdgeDirection in routeDirections){
12     if(SingleEdgeDirection's NavigableSpace = ↩
         previousNavigableSpace){
13       add SingleEdgeDirection to aggregatedDirections of ↩
           currentRoomDirection;
14     }
15     else{
16       store SingleEdgeDirection's NavigableSpace into ↩
           previousNavigableSpace;
17       create RoomDirection and store it into ↩
           newRouteDirections;
18       store RoomDirection into currentRoomDirection;
19       create RoomInstruction into RoomDirection;
20       store SingleEdgeDirection's NavigableSpace into ↩
           RoomInstruction;
21       create aggregatedDirections Collection of ↩
           SingleEdgeDirections into RoomDirections;
22       add SingleEdgeDirection to aggregatedDirections;
23     }
24   }
25   delete routeDirections;
26   return newRouteDirections;
27 }
```

Algorithm 6: *Floor Segmentation of Route Instructions*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Function: floorSegmentation
Input: graph, routeDirections (Collection of RoomDirections)
Output: routeDirections (Collection of FloorDirections)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

floorSegmentation{
  create previousCollectorSpace variable;
  create currentFloorDirection variable;
  create newRouteDirections Collection of RoomDirections;
  foreach (RoomDirection in routeDirections){
    if(RoomDirection.NavigableSpace.collectorSpace = previousCollectorSpace){
      add RoomDirection to aggregatedDirections of currentFloorDirection;
    }
    else{
      % complete previous floor direction
      create FloorInstruction into currentFloorDirection;
      store RoomDirection.NavigableSpace.collectorSpace into FloorInstruction.nextSpace;
      % create new floor direction
      store RoomDirection.NavigableSpace.collectorSpace into previousCollectorSpace;
      create FloorDirection and store it into newRouteDirections;
      store FloorDirection into currentFloorDirection;
      create aggregatedDirections Collection of RoomDirection into FloorDirection;
      add RoomDirection to aggregatedDirections;
    }
  }
  % complete last floor direction
  create FloorInstruction into currentFloorDirection;
  store "Destination reached" into FloorInstruction.nextSpace;

  delete routeDirections;
  return newRouteDirections;
}
```

Appendix C

Appendix C - Data model explaining examples

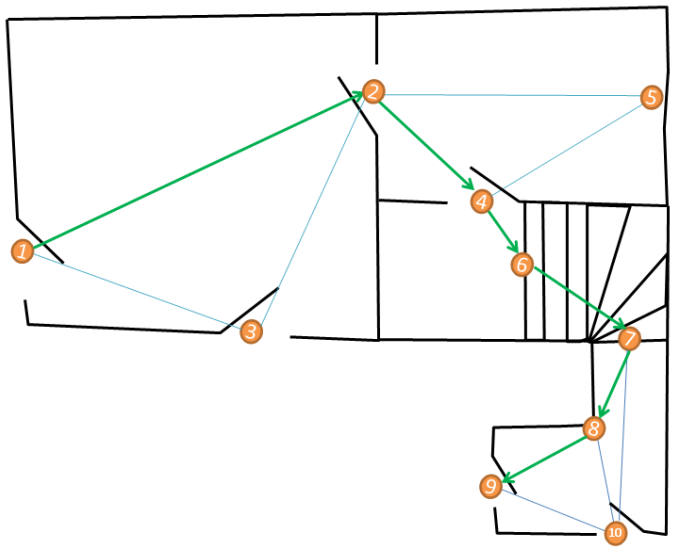


Figure C.1: *Example of a Building floor, with visibility graph: nodes are the orange dots with node number and the transitions are the blue segments. A path between nodes 1 and 9 is highlighted with green arrows.*

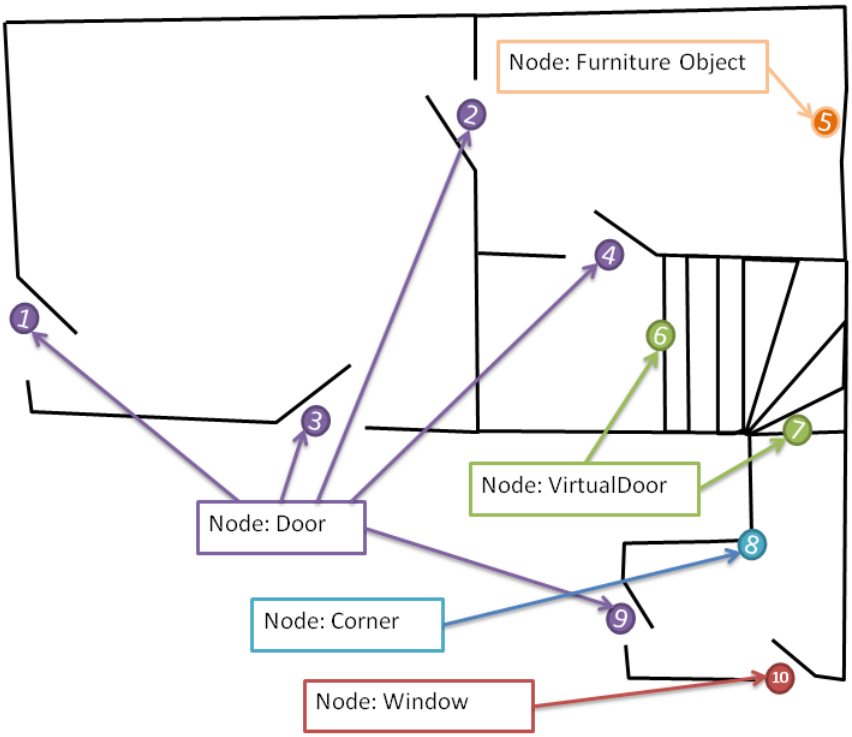


Figure C.2: *Nodes and their reference Class of object*

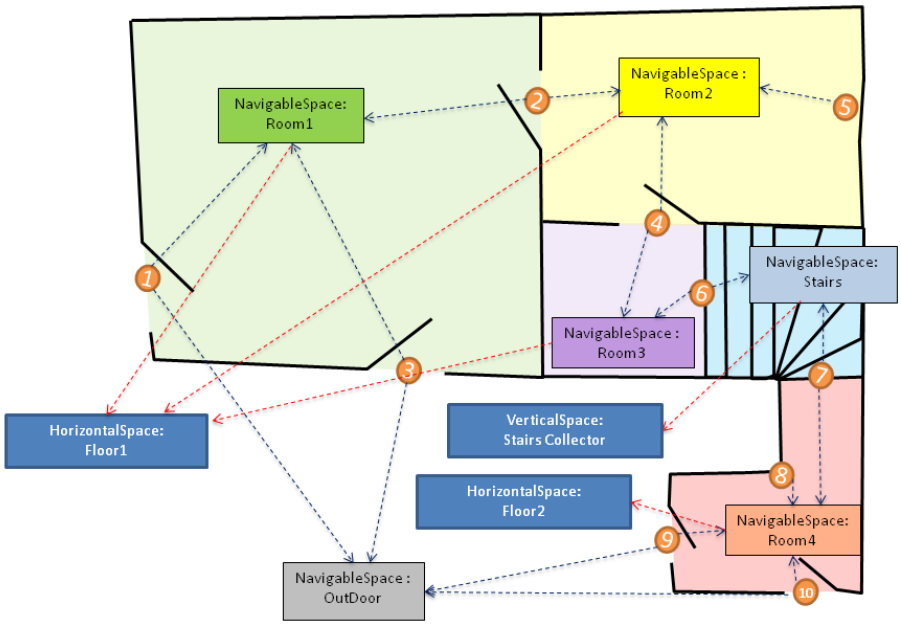


Figure C.3: *Using Notation "ClassName:ObjectDescription" is shown: Nodes and their associated Semantic Spaces marked with blue arrows, and the hierarchy of spaces Units(VerticalUnit and HorizotalUnit) and space collectors(VerticalSpace and HorizontalSpace) using red arrows.*

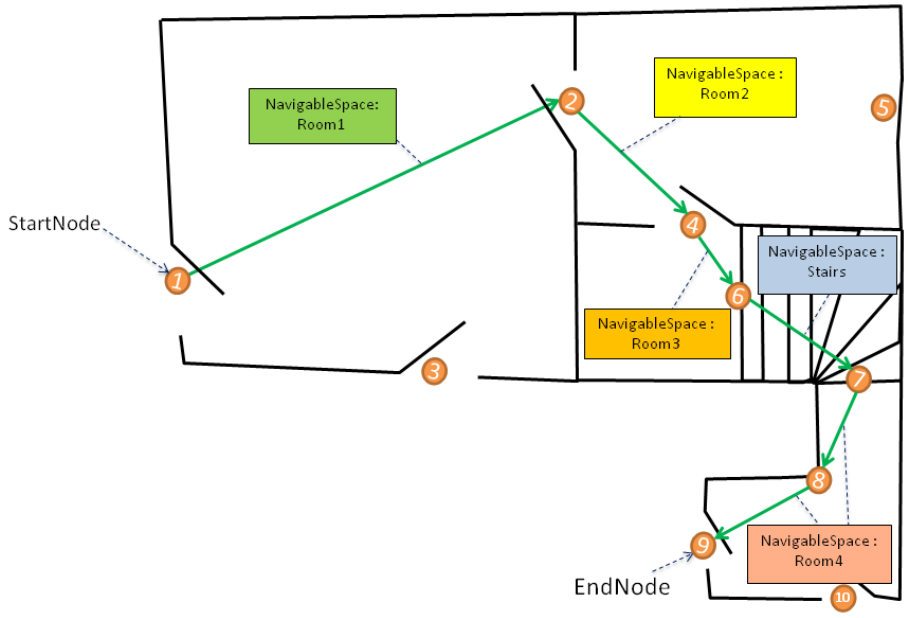


Figure C.4: *Using Notation "ClassName:ObjectDescription" is shown: Transition of a computed Path and their associated Semantic Spaces.*

Appendix D

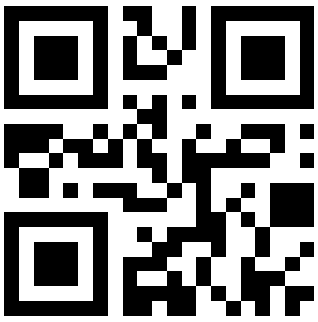
Appendix D - Route tests

Code snippet 7: *OTB dataset portion(most of data hidden due to space occupation)*

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Dataset xmlns:xsi="http://www.w3.org/2001/XMLSchema" xsi:↩
      noNamespaceSchemaLocation="file://datasetschema.xsd">
3    <Building>
4      <HorizontalSpace spaceName="floor 0" spaceID="1501">
5        <EndRoom privateAccess="false" spaceID="1523" ↩
           spaceName="toilet 0.2" belongingHSpaceID="1501" />
6        <HorizontalConnector privateAccess="false" spaceID↩
           ="1525" spaceName="corridor 0.0" belongingHSpaceID↩
           ="1501" /></HorizontalSpace>
7      <VerticalSpace spaceName="Stairs south" spaceID↩
          ="1506">
8        <Staircase privateAccess="false" spaceID="1512" ↩
           spaceName="staircase 1.1" verticalSpaceID="1506">
9          <connectedHSpaceID>1523</connectedHSpaceID><↩
             connectedHSpaceID>1502</connectedHSpaceID>
10       </Staircase></VerticalSpace>
11     <Ground privateAccess="false" spaceID="1509" spaceName↩
          ="ground" />
12   </Building>
13   <Graph>
14     <Door idNode="136" nodeName="Door 0.310" key="">
15       <Point3D x="22940.0" y="29611.5" z="0.0" />
16       <Image idImage="200" urlPath="" belongingSpaceID↩
           ="1523" />
17       <associatedSpaceID>1523</associatedSpaceID><↩
           associatedSpaceID>1512</associatedSpaceID>
18       <direction>3.141592653589793</direction>
19       <QRCode id="0" referenceNodeID="136" ↩
           belongingSpaceID="1523" />
20     </Door>
21     <Door idNode="108" nodeName="Door 0.320" key="">
22       <Point3D x="22940.0" y="29611.5" z="0.0" />
23       <Image idImage="201" urlPath="" belongingSpaceID↩
           ="1523" />
24       <associatedSpaceID>1523</associatedSpaceID><↩
           associatedSpaceID>1509</associatedSpaceID>
25       <direction>0</direction>
26       <QRCode id="1" referenceNodeID="108" ↩
           belongingSpaceID="1523" />
27     </Door>
28     <Transition minWidth="1.0" weight="1.0E29" minHeight↩
          ="2.5" available="true" belongingSpaceID="1532" ↩
          transitionID="0" fromNodeID="136" toNodeID="108" />
29   </Graph>
30 </Dataset>
```

“IndoorNav”

User Test 1



From: **Main entrance** (scan QRcode above)

To: **Room 1.240**

Without elevator

Figure D.1: *User test n 1*

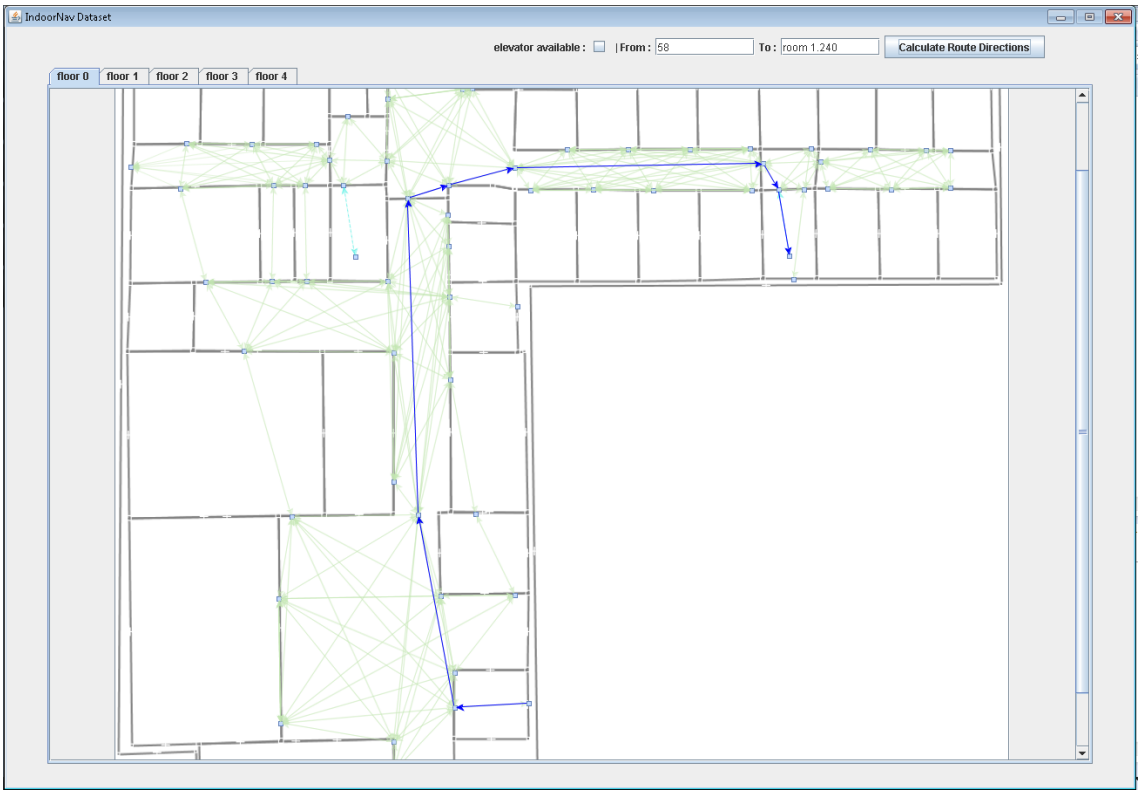
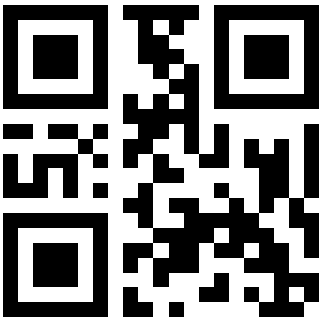


Figure D.2: *User test n 1, screenshot of path on Floor 1*

Figure D.3: *User test n 1, screenshot of path on Floor 2*

“IndoorNav”

User Test 2



From: **Room 1.240** (scan QRcode above)

To: **Room 2.120**

Without using elevator

Figure D.4: *User test n 2*

Figure D.5: *User test n 2, screenshot of path on Floor 1*

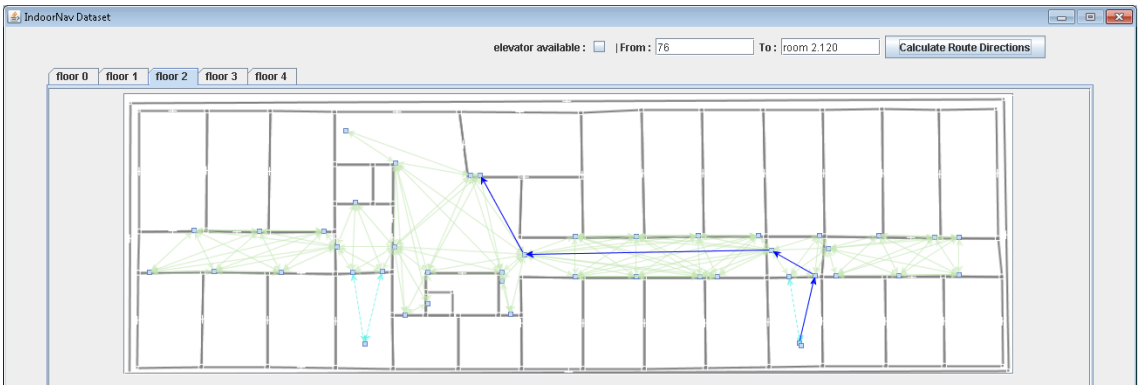


Figure D.6: *User test n 2, screenshot of path on Floor 2*

“IndoorNav”

User Test 3

From: **Room 2.170** (scan QRcode above)

To: **1160**

Without using elevator

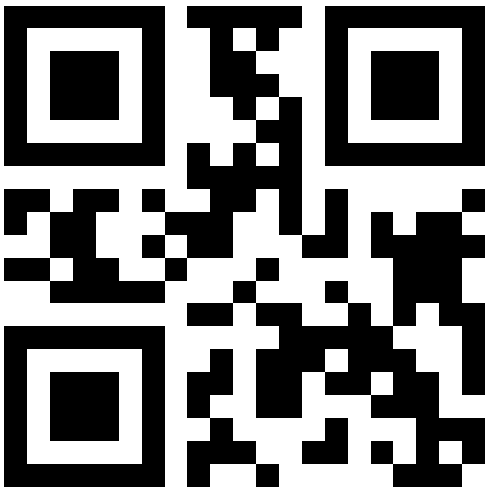
Figure D.7: *User test n 3*

Figure D.8: *User test n 3, screenshot of path on Floor 2*

Figure D.9: *User test n 3, screenshot of path on Floor 3*

“IndoorNav”

User Test 4



From: **Room 3.240** (scan QRcode above)

To: 788

Without using elevator

Figure D.10: *User test n 4*

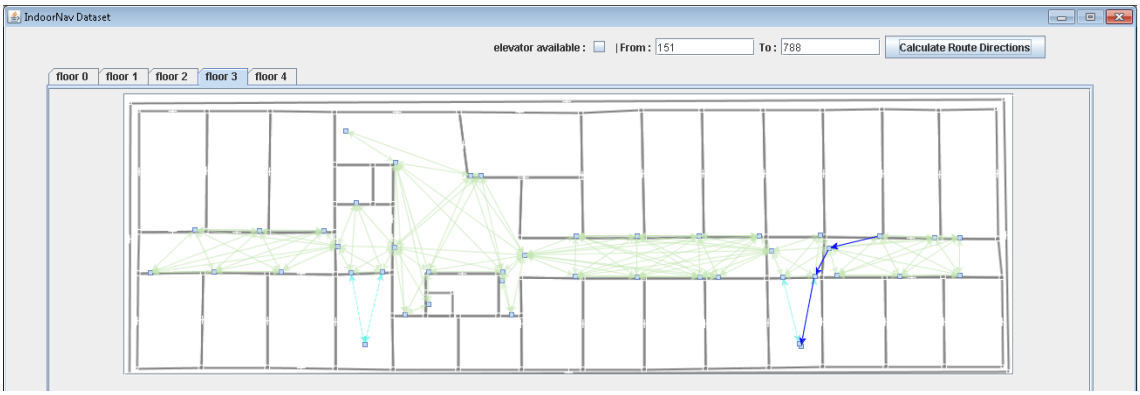


Figure D.11: *User test n 4, screenshot of path on Floor 3*

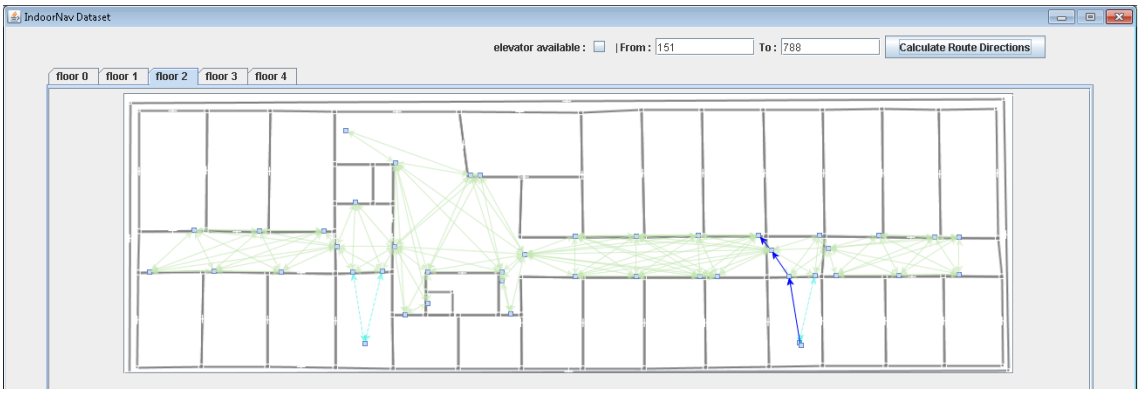


Figure D.12: *User test n 4, screenshot of path on Floor 2*