Navigation Systems for Special User Groups

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Bernhard Frederic Schmitz

aus Stuttgart-Bad Cannstatt

Hauptberichter:	Prof. Dr. rer. nat. Dr. techn. hc. Dr.-Ing. E.h. Thomas Ertl
Mitberichter:	Prof. Dr. rer. nat. habil. Gerhard Weber

Tag der mündlichen Prüfung: 9. Februar 2015

Institut für Visualisierung und Interaktive Systeme der Universität Stuttgart

2015

Acknowledgements

A work such as this is never the product of a single person alone, there are always those who contribute in one way or another to its inception, development and eventual completion.

I would therefore like to thank my primary advisor Thomas Ertl, who made my time at the Institute of Visualization and Interactive Systems possible and provided helpful guidance when I needed it. I also thank Gerhard Weber, who acted as co-examiner for this thesis.

My colleagues at the institute all contributed to the atmosphere that made working there a joy. I thank all of them for interesting discussions and for the fun we had together. Andreas Hub, Steffen Koch deserve a special mention for reading this thesis and providing helpful feedback. I also want to thank them and Hermann Pflüger for many in-depth discussions, both about research and about other topics. Martin Falk and Michael Wörner provided the typography and layout template that I adapted for this thesis.

I benefited from the immediate and invaluable feedback provided by Andrea Berghammer. Without her and Klaus Bosse, some of the systems described in this thesis would never have seen the light of day.

I would also like to thank all the colleagues in the collaborative research center Nexus, especially those with whom I had the pleasure of working together on specific projects: Susanne Becker, André Blessing, Matthias Großmann and Oliver Siemoneit.

Contents

Abstract

With the advent of smartphones and apps, navigation systems have become one of the most widely used mobile applications on the planet. At the same time there are some user groups, especially among people with disabilities, for whom the benefit of navigation systems is even greater than for the average user. Even though navigation systems for those smaller user groups have become available in recent years and are of great help to their users, those navigation systems are currently not as great a tool as they could potentially be, as has been shown by prototypes in smaller research projects.

From a user perspective, navigation systems give feedback about the current location, receive input about a desired destination, and guide the user to this destination. In addition to that, the system itself needs to determine the position and calculate a route to the destination. For all of these navigational tasks, the systems needs a world model, be it a map or another representation of the world.

Many existing projects have concentrated on adapting these navigational tasks to the specific requirements of a user group, and even though this thesis contributes to these efforts, especially for blind users, its main contribution lies in a different approach. The thesis supposes that the world model provides a great leverage for adapting navigation systems to the specific requirements of the users, as the world model has influence on all navigational tasks and therefore is an integral part of all modern navigation systems. Due to this importance, the world models of currently existing special navigation systems for people with disabilities often suffer from one of two distinct problems: If the model is specifically built for the intended purpose, e.g. a navigation system for a defined disability, it is only available in a confined area. This is mostly the case with research systems. Commercially available systems on the other hand strive to cover as large an area as possible, but have to accept certain drawbacks regarding the world model's applicability for the specific purpose. Ideally, a world model for special navigation systems is both available world-wide and specifically built or at least adapted for the intended purpose.

This thesis introduces a way of integrating both requirements. A world model that is widely available and used by many people builds a common base for the navigation system. This ensures the availability and currentness of the data. This world model is then changed individually with Map Content Transformations, which were developed specifically for this pur-

pose. These Map Content Transformations combine data that is implicitly present in the base data with user specific requirements encoded into the transformation rules and thus adapts the world model according to both the requirements of the user and of the intended navigational tasks.

It is shown that these adaptations of the world model can have positive influence on all navigational tasks and can, together with the incremental advances regarding the navigational tasks themselves, create an important step towards individualized navigation systems that optimally support their users in their spatial tasks.

Even though in this thesis Map Content Transformations and their usages are applied to the field of navigation for people with disabilities, they have the potential to be used in a variety of applications based on spatial data.

Zusammenfassung

Begünstigt durch die weite Verbreitung von Smartphones und Apps sind Navigationssysteme zu einer weitverbreiteten mobilen Anwendung geworden. Gleichzeitig gibt es einige Nutzergruppen, insbesondere unter behinderten Menschen, für die die Vorteile von Navigationssystemen sogar noch größer als für den durchschnittlichen Nutzer sind. Auch wenn in den letzten Jahren Navigationssysteme für diese kleineren Nutzergruppen erhältlich geworden sind und ihren Nutzern oft eine große Hilfe sind, so können sie doch ihr volles Potential als Hilfsmittel nicht entfalten. Nur einige Forschungsprototypen ließen bisher dieses volle Potential erkennen.

Aus der Sicht eines Benutzers geben Navigationssysteme Rückmeldung über den aktuellen Standort, erhalten eine Zieleingabe und führen den Nutzer dann zum gewünschten Ziel. Darüber hinaus muss das System selbst den aktuellen Standort feststellen und den bestmöglichen Weg zum Ziel berechnen. Für all diese Navigationsaufgaben benötigt das System ein Umgebungsmodell, sei es in Form einer Karte oder einer anderen Repräsentation.

Viele existierende Projekte konzentrieren sich darauf, diese Navigationsaufgaben an die speziellen Anforderungen einer bestimmten Nutzergruppe anzupassen. Auch die vorliegende Arbeit wirkt an den Bestrebungen in dieser Richtung mit, insbesondere für blinde Nutzer, ihr Hauptbeitrag jedoch liegt in einen anderen Ansatz. Diese Arbeit geht davon aus, dass das Umgebungsmodell einen großen Einfluss auf die Navigation hat, und daher dazu genutzt werden kann, Navigationssysteme an die spezifischen Anforderungen der Benutzer anzupassen, weil das Umgebungsmodell alle Navigationsaufgaben beeinflusst und daher ein integraler Bestandteil aller modernen Navigationssysteme ist.

Aufgrund dieser Wichtigkeit leiden aktuelle Navigationssysteme für Menschen mit Behinderung oft an einem von zwei gegensätzlichen Problemen: Wenn das Umgebungsmodell speziell für den angedachten Einsatzzweck, beispielsweise ein Navigationssystem für eine bestimmte Behinderung, erstellt wurde, ist es nur für ein eng umrissenes Gebiet verfügbar. Dies ist meist bei Forschungsprototypen der Fall. Kommerziell erhältliche Systeme hingegen versuchen, ein möglichst weiträumiges Gebiet zu umfassen, müssen dafür aber einige Nachteile bezüglich der Eignung des Umgebungsmodells für den gewünschten Einsatzzweck in Kauf nehmen. Im Idealfall ist ein Umgebungsmodell sowohl weltweit verfügbar als auch für den gewünschten Einsatzzweck erstellt oder zumindest angepasst.

Diese Arbeit stellt eine Möglichket vor, diese beiden Anforderungen vereinbar zu machen. Ein Umgebungsmodell, das weltweit verfügbar ist und von vielen Menschen genutzt wird, bildet die allgemeine Basis für das Navigationssystem. Dadurch wird sichergestellt, dass die Daten verfügbar und so aktuell wie möglich sind. Dieses Umgebungsmodell wird dann anhand von regelbasierten Transformationen des Karteninhalts, die speziell zu diesem Zweck entwickelt wurden, transformiert. Diese Transformationen des Karteninhalts kombinieren dabei Daten, die implizit in den Basisdaten vorhanden sind, mit nutzerspezifischen Anforderungen, die in den Transformationsregeln hinterlegt sind. Dadurch wird das Umgebungsmodell an die Anforderungen des Benutzers und der intendierten Navigationsaufgabe angepasst.

Es wird gezeigt, dass diese Anpassungen des Umgebungsmodell einen positiven Einfluss auf alle Navigationsaufgaben haben können und zusammen mit den anderen vorgestellten Fortschritten bei den Navigationsaufgaben einen wichtigen Schritt hin zu individualisierten Navigationssystemen darstellen, die ihre Nutzer optimal unterstützen.

Auch wenn die Transformationen des Karteninhalts hier für die Navigation von Menschen mit Behinderung verwendet werden, haben sie doch das Potential in verschiedenen anderen ortsabhängigen Anwendungen eingesetzt zu werden.

1

Introduction

Electronic assistance systems have become an invaluable aid to many people with disabilities in the last decades. In some areas, these systems have become a natural part of everyday life for their users, whereas in others, they are not yet as common, mostly due to technological limitations.

Navigation systems have become even more widespread, especially with the advent of smartphones. As can easily be imagined, navigation systems could be of even greater help to people with disabilities than they are for the average user. For this to happen, navigation systems must be specifically tailored to the needs of the user. Specific disabilities require specific changes in all aspects of the navigation system.

One specific user group that benefits greatly from a navigation system that is tailored to their needs, are blind people. Many of them are unable or reluctant to navigate environments that they are not accustomed to [6, p. 4]. Navigation systems can alleviate this reluctance and thus provide greater independence to their users. Accordingly, much work has been done in the field of navigation systems for blind people, but, so far there is no solution that really satisfies all the needs of the users. For this reason, this thesis covers all aspects of navigation systems and their adaptation to specific needs of special user groups.

1.1 Navigation Systems

A navigation system has three major stages that the user interacts with: Positioning, wayfinding and guidance (see Figure 1.1). First, the position and normally also the heading of the user needs to be established. The wayfinding algorithm can then compute the best possible route from the current position to the intended destination. Finally the user needs to be guided to

that destination. User input (the destination) is needed for the wayfinding component, if it is to produce any meaningful results, whereas positioning and guidance give feedback to the user, usually about his position, and about where to go. There are systems without a wayfinding component and therefore also without a guidance component. Those systems just establish the position of the user and give feedback about this position. They should possibly better be called “positioning systems” to avoid confusion with systems that have a wayfinding capability. The feedback about the user’s position is optional, as it is not strictly necessary for systems with guiding capabilities. It should be noted that position feedback and guiding instructions cannot always be distinguished clearly, as guiding instructions can include information about the environment, e.g. the instruction “Turn right into St. Deny’s Road”. Another possible distinction of feedback is between push and pull interaction. Push interaction presents information to the user without a prompt, whereas pull interaction requires an explicit request by the user that triggers information presentation. Mostly, pull interaction is used for position feedback and push interaction is used for guiding instructions, but exceptions are possible as well.

All three stages of navigation can again be divided into the algorithmic part and the user interaction. The determination of the position is independent of the feedback about the current position transmitted to the user and the wayfinding algorithm is independent of the way the user inputs his desired destination. For guidance, this distinction might not be as immediately obvious as for the other two stages, however, there is still a distinction of how the description of the guidance, whether purely directional or context-sensitive, is generated and how these instructions are transmitted to the user.

In Figure 1.1 all three components make use of a world model. This is also optional, as there are systems that work without any world model. A world model can be used in positioning by using reasonable assumptions (e. g. the user cannot walk through walls) to refine the computed position. Even if not used by the positioning algorithm, a world model can still be useful for the user feedback part of the positioning component, as it can convey an immediate idea about the current environment. World models, mostly in form of maps, have an obvious application in wayfinding, although there are systems that do not rely on maps for wayfinding. Those rely on hard-coded directions at specific points. Finally, a world model can be used to guide the user, as is well established with maps in car navigation systems.

This rough overview holds true for all kinds of navigation systems, but the focus of this thesis is on navigation systems for special user groups. In this context, “special user groups” refers to groups of people for whom

widely available navigation systems are insufficient for some reason. Mostly, this reason is a disability, but other user groups, for example bike riders, could well profit from some of the approaches presented in this thesis. Blind people are an example of a group for whom the need of a specifically adapted navigation system is immediately obvious and in many places this thesis refers to blind users as a prime example of a group that profits from such a system. Whenever pedestrians are mentioned in this thesis, the term is used in a broad sense and includes wheelchair users, conforming to the use of the term in many legal systems, e.g. the California Vehicle Code [25].

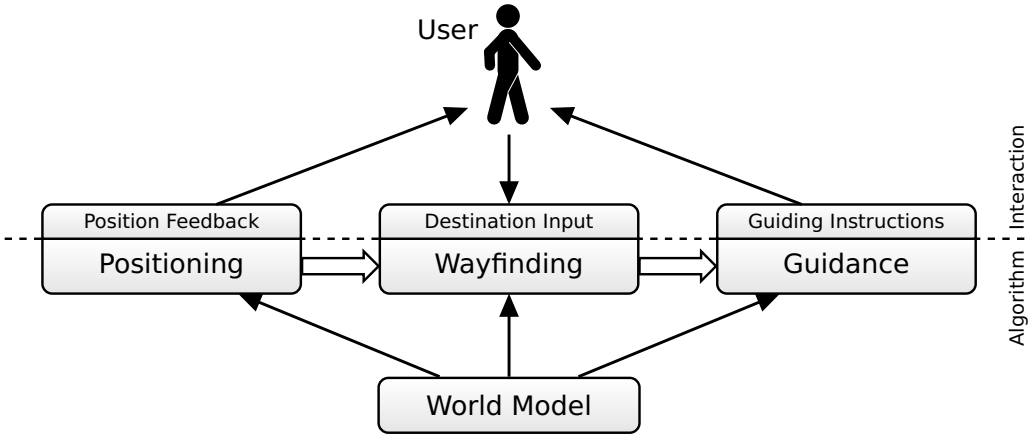


Figure 1.1: Schematic Depiction of a Navigation System. The three stages of navigation are supported by a world model, in most cases a map. The user receives feedback about the current position, can enter a destination and is instructed on how to reach that destination.

1.2 Problem Statement

The specific needs of the special user groups can be roughly separated into two classes: Information and modality. The users might need information that is missing from a standard navigation system, e.g. a wheelchair user needs information about slope and curbstones, or the user might need a special mode of transporting information, e.g. a blind user cannot use a visual map. Looking again at Figure 1.1, the modality affects the interaction of the user with positioning, wayfinding and guidance, whereas the need for additional information can mostly be satisfied by adapting the world model, but informs the three stages of navigation.

1.2.1 Individualization Challenges

Special user groups can profit from individualization in all stages of navigation. This individualization can again happen on the information level and on the modality level. Adapting the modality of the position feedback means that a fitting mode for the user is found, e.g. haptic or audio feedback for blind users. Adapting the positioning itself may seem counterintuitive at first. However, the requirements of the user for a positioning algorithm differ, e.g. a navigation system for blind users requires a greater accuracy, because the difference between being positioned in the middle of a street vs. being located on a sidewalk is not as immediately obvious to a blind user as it is to a sighted user. This does not immediately presuppose individualization, because a better positioning is desirable per se; however, it will be shown in Chapter 6 that some positioning algorithms can profit from individualization information about the user's behavior. Adapting the input to the wayfinding requires finding a method in which the user can easily enter his desired destination. Adapting the wayfinding itself can lead to totally different routes for different users, due to their personal preferences and abilities, e.g. one user might prefer a shorter route that includes steps, while another might be able to take stairs, but only willing to do so under certain circumstances, whereas yet another user might simply be unable to. Adapting the guidance stage's modality (e.g. vibration or sound for blind users) again differs from adapting the guidance algorithm, which can, for example, include descriptions about certain landmarks that can be helpful for navigation depending on the user's abilities.

Many navigation systems for special user groups have been developed, both as research projects and as commercial products. The most successful of these adapt both the modality and the world model. The necessity for adaptations of the modality is in many cases immediately obvious, e.g. for blind people, and there has been much previous work that concentrated on adapting the modality for one specific user group, as will be shown in Chapter 2. All of those systems were adapted to a specific user group as defined by the creators of the system, normally by identifying a specific disability. These groups are called "requirements groups" here, because they can be identified by easily distinguishable different requirements. However, as shown by Williams et al. [116], subgroups or "navigation personas" with individual requirements and preferences exist inside such requirements groups. A similar study by Banovic et al. [6] interviewing only blind people on their navigation preferences confirmed these findings of subgroups. As an example, one participant stated: "I'm goal oriented, so for example somebody says: 'Come and meet me at the certain spot', then I would do that. But I tend not to look around." [6, p. 3] In contrast to

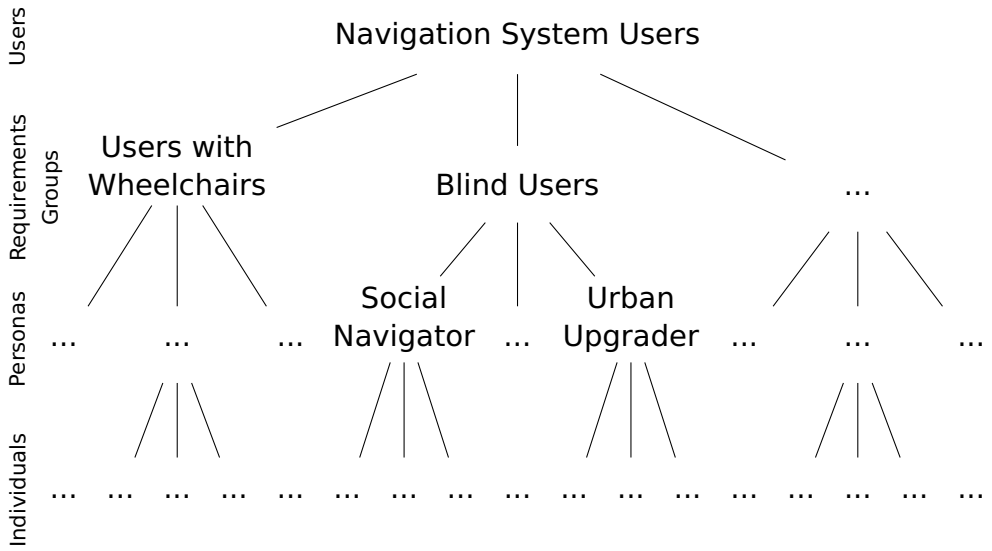


Figure 1.2: User Group Hierarchy. Groups get more specific, but group size becomes smaller towards the bottom of the hierarchy. The examples of personas are chosen according to Williams et al. [116]. Groups from all levels of the hierarchy are referred to as “user groups” throughout the thesis.

that, another participant said: “I’d like to know as much as I could about my environment without having to tax somebody else. Anything that brings me closer to experiencing my environment the way a sighted person would, would be useful for sure.” [6, p. 4] Figure 1.2 shows a classification derived from these findings.

Both Williams et al. [116] and Banovic et al. [6] clearly identify the need for a further individualization at lower levels of the user group hierarchy. However, with each step down the hierarchy, the groups get more specific, but the group sizes get smaller.

As a result, the adaptation of both the information and the modality suffers from a common dichotomy: The need for individualization is becoming greater the more specific the user group gets, yet at the same time it becomes less and less probable that such an individualization exists, because any such individualization becomes less commercially viable, the smaller a user group gets.

The individualization of the modality often requires specific hardware, but, once developed, it would work independent of the current location of the user. On the other hand, the information that is required in a navigation system does not involve the development of specific hardware, but is location-dependent. Thus, previous efforts in research that were successful in adapting to their user group have often been confined to a designated

area that was selected as the test area for the prototype. Commercially available systems concentrated on adapting the modality and relied on widely available map data that is not adapted to the specific needs of the users and thus do not provide as much benefit as would be possible with a more sophisticated and specialized world model [116, p. 2]. Furthermore, almost all previous work provided adaptation on the “requirements group” level, but few work has been done towards individualization at lower levels.

1.2.2 Specific Challenges in Navigation for Blind People

In addition to the problem of individualization Williams et al. [116] and Banovic et al. [6] identify further specific challenges in navigation for blind users, including:

Integration of Information Sources. All smartphone users were using several navigation applications, due to the availability of different functionality and information. All users regretted the need for switching applications [116, p. 4].

Street Crossings and Construction Areas. Those were identified as particularly challenging navigational tasks by almost all participants [116, p. 5].

Up-Front High Level Information. Participants reported that they gather high level information about an area before visiting it, mostly by communicating with friends, family or Orientation and Mobility (O&M) specialists, but expressed a preference for not having to rely on anyone [6, p. 3].

Sharing Information. A desire and willingness to share useful information about the environment was reported by the participants [6, p. 7].

1.3 Research Questions

Taking into account all of the above, several important questions arise:

Large-Scale Adaptation. Is it possible to create a system that allows adaptation and individualization of the world model on a large scale, possibly world-wide?

Adaptation Flexibility. Can these adaptations be designed so flexible that they can be applied on every level of the user group categorization?

Algorithmic Benefit of Adaptation. Can the three stages of navigation profit from such an adaptable world model, i.e. is an adapta-

tion of the algorithmic part of the stages possible by adapting the world model?

Integration with User Interaction. How do the interaction parts of the three stages of navigation integrate with the adaptations to their algorithmic counterparts?

Benefit for Navigation for Blind People. Does such a system help to meet the challenges listed in Section 1.2.2?

1.4 Contribution

This thesis demonstrates that adaptations of the world model are indeed possible on a large scale. It presents two different approaches to adapted world models.

The more traditional approach integrates different information sources and harmonizes their internal representation so as to simplify their use by the stages of navigation [97].

The second approach has a more general outlook. It is based on the idea that there is no optimal way of storing information in world models or maps, and that a specific way of mapping that is of benefit to one user group could be an annoyance or even a hindrance for another group. In order to give consideration to both groups, the proposed solution consists of a model that—implicitly or explicitly—contains all the data for all users, and transformations that create individualized representations for each task or user group. It therefore becomes easier to create individualizations even for levels further down the user group hierarchy (Figure 1.2), as there is no need to create a new world model from scratch. Instead a new transformation—or the modification of an already existing one—should suffice.

As a concrete implementation of this proposed solution, this thesis presents Map Content Transformations. OpenStreetMap was chosen as a common world model. The collaborative nature of OpenStreetMap has the advantage of enabling each user group to contribute specific information that is only relevant to their group. A rule-based scripting language, which has been developed specifically for this purpose, is used to transform the common OpenStreetMap data, and the resulting map is better fit for the specific purpose because it combines data that was only implicitly present in the base data with the requirements encoded into the rules. The result can thus be individualized according to the specific requirements of both the user group and the stage of navigation that it is intended to be used for.

The thesis discusses the effects of this adaptable world model on the three stages of navigation and shows how they can benefit from it by

presenting concrete implementations that improve the current state of affairs in each of these stages. It demonstrates that Map Content Transformations in conjunction with OpenStreetMap can be used to overcome the challenges listed in Section 1.2.2 by providing specific examples improving the situation for each challenge. It discusses the effects of the adaptable world model on those three stages and shows how they can benefit from it by presenting concrete implementations that improve the current state of affairs in each of these stages. It presents advances in the stages themselves as well as adaptations of their modality and explores how those adaptations of modality interact with the output of the algorithmic level of each stage.

1.5 Thesis Structure

This thesis is structured according to the model of navigation presented in Figure 1.1. After an introduction to the state of the art in Chapter 2, the adaptation of world models and maps is explained in chapters 3 and 4 and serves as a foundation both in theory and in practice for the later chapters, which thoroughly deal with each stage of navigation. Chapter 5 is focused on positioning. It explains the basics of particle filters for positioning and demonstrates the particle filter that was implemented in the navigation system used throughout the thesis. With those basics explained it is demonstrated in Chapter 6 that a positioning system that uses a particle filter can profit from individualized maps created with Map Content Transformations. After the position has been determined, the system can give feedback about the current environment, which is explored in Chapter 7. The same chapter also discusses stationary systems for map feedback, as the underlying principles are the same, and those systems can profit just as well from personalized maps as mobile navigation systems. In Chapter 8, it is shown how Map Content Transformations can improve this map feedback. Route planning is treated in Chapter 9. Route planning is considered as the entirety of this stage of navigation, i.e. the combination of destination input and wayfinding. The chapter describes how both can be individualized by using maps that are adapted for the specific needs of the user. Chapter 10 covers guidance as the last stage of navigation. Again, both parts of this stage are looked at, with an emphasis on the information part and how it can be enhanced by individualized maps created with Map Content Transformations.

This thesis is based in parts on the following previously published documents:

Bernhard Schmitz. The ViibraCane – a white cane for tactile navigation guidance. In *Proceedings of the California State University, Northridge Center on Disabilities' 25th Annual International Technology and Persons with Disabilities Conference (CSUN 2010)*, 2010. Extended Abstract.

Bernhard Schmitz and Thomas Ertl. Making digital maps accessible using vibrations. In *Proceedings of the 12th International Conference on Computers Helping People with Special Needs (ICCHP 2010)*, pages 100–107, 2010. DOI:10.1007/978-3-642-14097-6_18

Bernhard Schmitz, Susanne Becker, André Blessing, and Matthias Großmann. Acquisition and presentation of diverse spatial context data for blind navigation. In *12th IEEE International Conference on Mobile Data Management (MDM 2011)*, pages 276–284, 2011. DOI:10.1109/MDM.2011.66

Bernhard Schmitz and Thomas Ertl. Rule-based transformation of map data. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops 2012)*, pages 584–589, 2012. DOI:10.1109/PerComW.2012.6197581

Bernhard Schmitz, Attila Györkös, and Thomas Ertl. Combination of map-supported particle filters with activity recognition for blind navigation. In *Proceedings of the 13th International Conference on Computers Helping People with Special Needs (ICCHP 2012)*, 2012. DOI:10.1007/978-3-642-31534-3_78

Bernhard Schmitz and Thomas Ertl. Interactively displaying maps on a tactile graphics display. In *Proceedings of the Workshop on Spatial Knowledge Acquisition with Limited Information Displays*, pages 13–18, 2012

Bernhard Schmitz and Thomas Ertl. Creating task-specific maps with map content transformations. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on MapInteraction (MapInteract '13)*, pages 84–90, 2013. DOI:10.1145/2534931.2534945

Bernhard Schmitz and Thomas Ertl. Individualized route planning and guidance based on map content transformations. In *Proceedings of the 14th International Conference on Computers Helping People with Special Needs (ICCHP 2014)*, pages 120–127, 2014. DOI:10.1007/978-3-319-08599-9_19

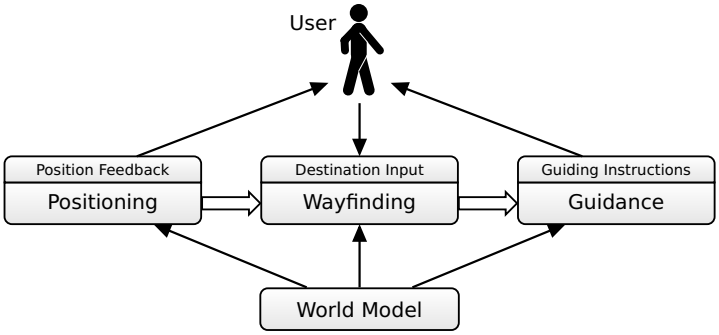
The following published documents were not directly used for this thesis:

Andreas Hub and Bernhard Schmitz. Addition of RFID-based initialization and object recognition to the navigation system TANIA. In *Proceedings of the California State University, Northridge Center on Disabilities' 24th Annual International Technology and Persons with Disabilities Conference (CSUN 2009)*, 2009. Extended Abstract.

Bernhard Schmitz and Andreas Hub. Combination of the navigation system TANIA with RFID-based initialization and object recognition. In *Proceedings from 7th European Conference of ICEVI*, 2009. Poster Presentation.

Oliver Siemoneit, Christoph Hubig, Bernhard Schmitz, and Thomas Ertl. Mobiquitous devices and perception of reality. A philosophical enquiry into mobile and ubiquitous computing devices that alter perception using the example of TANIA – a tactile acoustical indoor and outdoor navigation and information assistant for the blind, deafblind, and visually-impaired users. In *Proceedings of the 5th Asia-Pacific Computing and Philosophy Conference*, pages 123–130, 2009

Ahmed El-Safty, Bernhard Schmitz, and Thomas Ertl. An openstreetmap editing interface for visually impaired users based on geo-semantic information. In *Proceedings of the 14th International Conference on Computers Helping People with Special Needs (ICCHP 2014)*, pages 116–119, 2014. DOI:10.1007/978-3-319-08599-9_18



2 State of the Art

Past research has been considering all parts of a navigation system as laid out in Figure 1.1. Some work is especially focused on one specific aspect, while other groups developed and tested complete navigation systems. Those complete systems have to cover all parts, but some concentrate on one or two specific aspects, while mentioning others only in passing. This chapter gives an overview of the state of the art and presents how previous work has tackled the different challenges regarding all stages of navigation. The stages themselves are classified according to the techniques that are being used, and the previous work is categorized according to that classification. If necessary, systems covering several aspects of navigation are mentioned more than once, in the appropriate sections. As this thesis is particularly concerned with navigation for specific user groups, this chapter does not cover general navigation systems that are not tailored to a certain user groups. As in the entire thesis, a specific focus is put navigation systems for blind people.

Due to this focus, a short explanation of terms in this context is in order. A blind person trying to get from one position to another has to overcome two major difficulties: How to get to this other position, and how to avoid obstacles on the way there. The difference between these two tasks has been identified for a long time.

Obstacles are classically avoided by using a long cane or a guide dog. The task is normally called obstacle avoidance [16, 17] or clear path guid-

---

Parts of this chapter have previously been published in:

Bernhard Schmitz, Susanne Becker, André Blessing, and Matthias Großmann. Acquisition and presentation of diverse spatial context data for blind navigation. In *12th IEEE International Conference on Mobile Data Management (MDM 2011)*, pages 276–284, 2011. DOI:10.1109/MDM.2011.66

ance [54]; the term micro-navigation has also been employed [79]. A whole class of electronic systems trying to help the user with this task have been developed for a long time and have also become commercially available in the last decade. These systems are commonly called Electronic Travel Aids (ETAs) [122]. Neither the entire thesis nor this chapter are concerned with ETAs, except in cases where specific solutions can be applied in full-fledged navigation systems.

2.1 Positioning

Computing the current position of the user is a task faced by all navigation systems and not exclusive to those for special user groups. However, demands on accuracy and availability vary with the purpose of the system, e.g. there is no need for a car navigation system to operate indoors. In fact, the field of location sensing is even broader than the application of navigation systems, as location sensing is also of use in the field of smart factories, in emergency situations and many other applications.

The development in this field is quite fast, leading Heathcote [37] to predict in 2004 that “In 10 years’ time there will be no concept of lost”. This prediction has not come true, though, and while it is reasonable to assume that one day it will, it is not clear when that will be.

Hightower and Borriello [42] have identified 15 different categories of location sensing technologies. Besides measures such as accuracy, scale and cost, they propose four categories: Physical position vs. symbolic location, absolute vs. relative positioning, localized computation and recognition. Physical position vs. symbolic location refers to whether a system places objects in a physical frame of reference (such as latitude and longitude) or uses abstract ideas such as “on the desk” or “in the mailbox”. Absolute vs. relative refers to whether all objects are placed in the same frame of reference or whether each object has its own frame of reference. Localized computation means that the computation of the position occurs at the location of the user (or object) (e.g. in the Global Positioning System (GPS) device) in contrast to computing the location at some central location, e.g. the servers of a service provider. Localized computation ensures privacy, as long as the located position is not broadcast. Recognition describes whether a system is not only able to localize an object, but also to recognize which object has been localized.

This classification is useful for the general field of ubiquitous computing, but less so for navigation systems. As an example, almost all systems presented here apply localized computation and use absolute position-

ing. Furthermore, navigation systems should be categorized according to aspects relevant to the users and the operator, if possible.

The area of operation is one of the primary concerns to the user. One obvious aspect is whether a system works indoors or outdoors. There are systems that work only outdoors (e.g. GPS-based systems) as well as systems that work only indoors (e.g. systems based on ceiling-mounted infrastructure). Obviously, there are also systems working both in- and outdoors. This is normally achieved by either combining several positioning methods, by having an infrastructure that is applicable both in- and outdoors, or by being completely independent from any infrastructure.

The other aspect of the area of operation is the size of the area. GPS-based systems work worldwide, while pure indoor systems obviously have a very limited area of operation. The size of the area of operation corresponds with in- and outdoor applicability, but is not identical to it. However, the one thing that normally limits the area size (at reasonable cost) is the dependency on installed infrastructure, i.e. the necessity to install infrastructure is equivalent to a small area of operation. The installation of infrastructure is of great relevance to the operator of a system and therefore used for the categorization. It could be argued that GPS-based systems are reliant on infrastructure, and indeed the satellites are a very cost-intensive infrastructure. However, as this infrastructure is available without any cost to a potential operator, those systems are categorized as not needing infrastructure, corresponding to their global applicability.

The different positioning systems, which are described in detail below are summed up and categorized in Table 2.1.

2.1.1 Outdoor Systems

Systems that work only outdoors are usually based on a technology that is relying on signals which could be blocked or uncontrollably reflected by building walls. The prime example for this kind of technology is GPS.

2.1.1.1 GPS-based Systems

GPS is used in many outdoor navigation systems, both commercially and in research. GPS is universally available and a well-established and reliable technology. Unfortunately its accuracy is dependent on conditions such as weather and satellite visibility, making it less accurate in street canyons. Many systems try to reach a better accuracy by using Differential GPS, however this does not solve problems such as blocked signals or multipath errors.

The Drishti system [38] is one of the early and most well-known systems in this area. It combines differential GPS with an electronic compass for heading information. Later versions added an indoor component (Section 2.1.2.3).

The Personal Guidance System [64, 65, 66, 70] has also been using differential GPS since its inception, as has the MoBIC system [79]. The system by Sánchez and de la Torre [88] uses only a standard GPS receiver.

2.1.1.2 RF Beacons

Radio Frequency (RF) Beacons have been used in a system for the very restricted task of assisting a blind runner to keep in his running lane [101]. The beacons are placed along the running lane with additional receiver beacons placed at a fixed distance for calibration. The position is then determined by looking up the RSSI attenuation levels of the received signals in a previously calculated table that includes the corresponding positions. It is one of the best system with regards to both accuracy and speed, which is paid for by its restricted applicability.

2.1.2 In-/Outdoor Systems

Many systems which were developed for indoor use can also be applied for outdoor use in restricted areas. Other systems were deployed outdoors for the experimental setup, but could be used indoors without problems.

2.1.2.1 RFID Tags

Radio Frequency Identification (RFID) technology can be used for positioning if the tags are placed at fixed locations and the correspondence between each tag id and its position is known. Active RFID tags need an energy source, but normally have a higher range than passive RFID tags. Those do not require an energy source themselves, but rely on the active reader providing enough energy for identification.

Amemiya et al. [43, 2] use a dense grid (1.2m between tags) of active RFID tags and averaging over the positions of all received tags as well as the previous position. To minimize the influence of the human body blocking RF signals, two readers, one placed on each shoulder, are used. A standard deviation of 400mm for the computed position compared to the actual position was reported. The experimental environment of the project was set up outdoors, but the system should also work indoors.

If passive RFID tags are used, the grid needs to become even denser. Willis and Helal [117] integrate passive RFID tags into the floor at a dis-

tance of 30.5cm between them. To solve the problem of the short range of the employed tags, the reader is integrated into either the shoe or the cane of the user. Willis and Helal tested the setup with two different walking speeds. With the faster speed (between 2 and 2.5 km/h) they were able to read about half of the possible tags. A similar setup was used by Ienaga et al. [53].

Ghiani et al. [30, 31] propose a system for the specialized task of navigation in museums. The system employs active RFID tags with a range of 5m, one for each exhibit, and links the id of each tag with information about the exhibit and positional information.

Bessho et al. [11, 12, 13] combine many similar space identification techniques, namely active and passive RFID tags, infrared beacons and QR code (a 2D barcode that can be read with cameras) to a “space identifiying ubiquitous infrastructure”. Each beacon identifies a place, the system then uses a spatial model connecting the places for further processing.

2.1.2.2 WiFi-Based

The current location of a user can be computed by using the signal strength of WiFi networks [89]. Bowen et al. [18] used the technology in a navigation system tailored to the needs of people with disabilities, mostly wheelchair users. The main advantage of using WiFi networks is that WiFi is often already available and must just be complemented if necessary. The technology is often used in conjunction with other techniques. The WiFi-based positioning is then used for a coarse location before further refining by the other technique. This is especially useful in cases where the needed computation power can be reduced significantly if a rough estimation of the location is known [50, 118]. WiFi-based positioning was popularized by the PlaceLab initiative [62] and can be found in every smartphone today.

2.1.2.3 Ultrasound

Positioning by using ultrasound can be very precise but requires expensive infrastructure. A version of Drishti extended for indoor use employs a commercially available ultrasound positioning system for indoor navigation [80]. A positioning error of less than 22 cm was reported.

2.1.2.4 Inertial Tracking

Inertial tracking tracks the movement of the user relative to a starting position. The main sensors are inertial sensors, although gyroscopes and compasses are also frequently used as parts of an inertial tracking system.

Inertial tracking works both in- and outdoors and is independent of a pre-installed infrastructure. However, inertial tracking does not work without further pre-knowledge or a complementary positioning system, as it is only relative to a starting point that needs to be determined. Additionally, small errors that occur during tracking add up over time and need to be corrected.

The TANIA system [50, 51, 52] uses inertial tracking and detailed maps of the environment. The inertial sensor is placed in the neck, and a calibration has to be done for each user independently. Furthermore it uses GPS (or user input) for initial positioning and correction of errors.

In a similar system by Godha et al. [32], the inertial sensor is placed on one foot. Furthermore the GPS signals are not only used for an initial position and to correct the tracked position from time to time, but also to compute the stride length, which makes individual calibration unnecessary.

In order to refine the tracking, a particle filter can be used [118]. Combined with a map of the environment, the system provides quite accurate tracking, if an initial position is known. Woodman and Harle report an accuracy of 0.5m 75% of the time and 0.73m 95% of the time. The system is even able to determine the position in a building correctly after some time of walking around if no initial position is known. However, this cannot be done in real-time yet, which is why a WiFi-based system to narrow down the initial position was used.

2.1.3 Indoor Systems

The reason that some systems work only indoors is that they either rely on ceiling-mounted infrastructure or that they need walls as part of their positioning algorithm (normally by calculating the distance to the wall).

2.1.3.1 RFID Tags

Kulyukin et al. [61] tag all doors, turns and destination objects and use a mobile robot to guide the user. The robot can read the tags but is able to locate itself more accurately, due to the knowledge of its own motion actors and a laser range finder. Because of the construction of the robot on wheels and the use of the laser range finder, the system can only be used indoors, in contrast to the other RFID-based systems mentioned in Section 2.1.2.1.

2.1.3.2 Barcode-based

Barcodes are omnipresent in everyday life on commercial products. Their field of application differs little from that of passive RFID tags, and there-

fore they can also be used in a similar way. The Metronaut system [103] uses what is referred to as a stickernet of barcodes and a standard barcode scanner.

2.1.3.3 Infrared

Positioning with infrared transmitters works according to the same basic principle as do RFID tags and barcodes, as there is a unique ID, in this case provided by infrared transmitters, which is linked to a known position.

The system by Sonnenblick [104] relies on infrared transmitter units which are mounted on the ceiling above important landmarks (e.g. in front of doors). The receiver unit will then play a customized description of the current location. The system can only be used to identify the current location and has no ability to guide the user to a desired destination.

A similar but more sophisticated system was developed by Ertan et al. [28]. In this case the ceiling-mounted infrared transmitters are installed in such a way that the whole path is covered, thus allowing a continuous position estimation.

Talking Braille uses a network of small, solar-powered beacons called cyber-crumbs, which interact with a badge worn by the user via Infrared [86]. To compute the distance between user and beacon, ultrasound is used, whereas for his heading the difference in received signal strength between two infrared receivers with an angle of 90° between them is used.

2.1.3.4 Visible Light

Somewhat similar to the infrared approach is the idea to convey information in the non-visible flicker of visible light. The electronic ballast circuit of standard fluorescent lamps can be modified to change the light frequency, thus encoding information [24]. The advantage of this system is that many buildings are already equipped with fluorescent lamps and only a low-cost adaptation of the electronic ballast circuits is necessary.

2.1.3.5 Vision-Based

Vision-based positioning should not be confused with vision-based obstacle avoidance. Vision-based positioning determines the position of the user, but needs pre-knowledge in order to achieve this. Hub et al. [47, 48, 49] use a stereo camera to compute a picture with distance information, which is then compared to a 3D model of the building. By determining the current room with a WiFi positioning system and the heading of the helmet-placed camera with a 3D compass the computing power needed for the comparison

	indoor	outdoor
without infrastructure	[47] [48] [49] [32] [50] [51] [52] [118] [107]	[64] [65] [38] [80] [66] [32] [50] [51] [52] [88]
with infrastructure	[103] [104] [2] [80] [86] [117] [18] [50] [11] [51] [52] [12] [13] [24] [30] [31] [53] [57]	[2] [117] [61] [18] [11] [12] [13] [101]

Table 2.1: Positioning Systems. Categorization of the systems mentioned in this chapter according to the need for infrastructure and application area. Entries are sorted according to year of publication, numbers indicate alphabetical order according to first author.

is reduced. The location in a room can then be calculated very exactly. Other systems use markers, which are placed in the environment and detected by a camera, such as QR-codes or color-coded markers [57], or work purely computer-vision-based, without markers [107].

2.2 Wayfinding

There have been relatively few publications in the field of wayfinding for navigation systems for specific user groups, presumably because researchers consider the problem solved with Dijkstra's algorithm or because wayfinding is a broader problem that is attended to by a large research community not directly involved in the field of assistance systems. A notable exception is the work by Völkel and Weber [111, 112] that explores multimodal wayfinding, which means wayfinding dependent on the disabilities of the user. As a basic example, a wheelchair-bound user possibly

needs another route than a blind user, who can walk staircases. The Drishti system has a special wayfinding component, which uses traffic congestions on walkways observed during different times of the day as weight functions [38]. Kulyukin et al. [61] also give a detailed description of their wayfinding algorithm, which is a breadth-first-search in a manually engineered connectivity graph between the RFID tags they use.

2.3 Guidance

When observing the task of guiding the user to a specific location, three different aspects of the information given to him have to be considered: The sensory system to which the information is fed, the reference system, in which the information is given, and the form, in which the information is presented. As this applies to the position feedback as well, this chapter includes information about position feedback, especially in systems where the boundaries between position feedback and guiding instructions are blurred.

According to Aristotle, man has five senses. Since human smell and taste are not fine-grained enough to be applicable for a navigation system, they will not be regarded here. In contrast to that, sight, hearing and touch can be used to give a user visual, auditory or tactile cues regarding his whereabouts and/or about where to go. Most standard car navigation systems today use both visual and auditory cues. Visual cues are used in the form of simple arrows, maps, or a schematic 3D depiction of the immediate environment. Audio cues are mostly provided by speech output, announcing the next turn or freeway exit to take, or similar events. In navigation systems for blind people the information that is normally presented visually or can be recognized immediately from the environment must be presented by other means. This fundamentally changes the nature of the audio cues. It is not sufficient to say “next right”, as the user might not be able to differentiate between a gateway and a walkway that come up to his right. Furthermore, navigational assistance is also needed in large open spaces without any inherent tactile cues such as curbstones.

In all three sensory systems (i.e. visual, auditory or tactile) the form of the information can be map-based, text-based or purely direction-based. Map-based systems convey information about the (normally surrounding) area and the user’s position in that area in form of a map and a marker. Speech output may describe the area or give navigational information in the form of natural language. Purely direction-based information can only be used for concrete navigational tasks, i.e. the guiding to a point, be it a waypoint or the final destination.

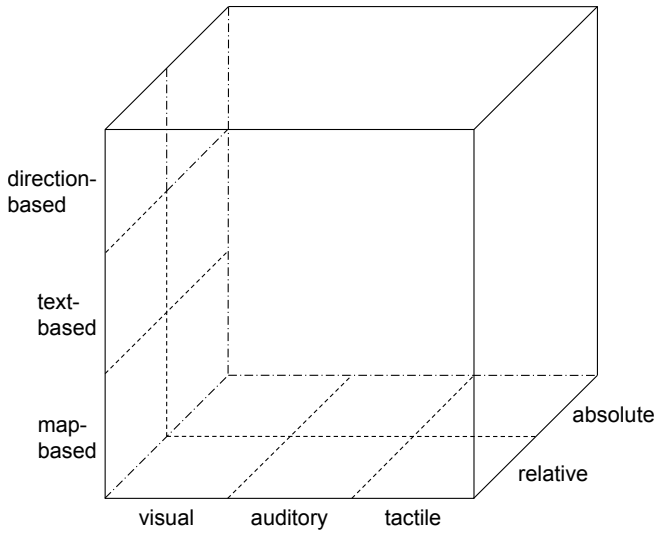


Figure 2.1: Guidance in navigation systems. Guidance can be classified according to three criteria, forming three dimensions for classification: The sensory system, the reference system, and the form. A categorization of the systems mentioned in this chapter can be found in Table 2.2

The reference system of the information given can be absolute (e. g. “northwest”) or relative (e.g. “left”). Relative systems can be further distinguished by whether they define the heading of the user by his torso, his head, or his hand (in the last case the user is normally holding a handheld pointing device) [66]. It could be assumed that purely direction-based information (i.e. any way of conveying nothing but a direction) is always relative. However, the SpaceSense system by [119] uses absolute directions to give directions (i.e. “head north”) and to indicate nearby places of interest.

Generally, in order to give relative information, the heading of the user needs to be known, either by specific sensors or by computation of the positional change over time.

Figure 2.1 gives a graphic overview of the different aspects of guidance. It will be filled in Table 2.2 with the different navigation systems that are presented in this chapter.

2.3.1 Visual Cues

Visual cues include maps, directional arrows (as on street signs), photos, arrows in a (stylized) 3D environment as well as written text. They are being used in almost all navigation systems for sighted people. Applications for special user groups include the system by Chang et al. [23], which uses arrows imposed on pictures to guide persons with cognitive disabilities, the

tour guide by Bessho et al. [12] that uses a combination of maps and photos, and the system by Cheok and Li [24] that uses arrows on a head-mounted display.

2.3.2 Auditory Cues

Auditory cues include speech output, direction-giving sound and auditory maps.

2.3.2.1 Speech Output

As already mentioned, speech output is used in most current car navigation systems and is an obvious choice for navigation systems for blind people. It is therefore little surprise that synthetic speech has been used since the very beginning of the research in this field [64] and has been in continuous use since then. It will not be completely replaced by any other technique, as it has the advantage of being able to not only transfer directions but also additional information, e.g. about buildings/rooms in the vicinity. Speech output can be used for directions of varying complexity, such as simple directions (left/right), more complex directions (45° left) and detailed descriptions of the environment and the route to follow. Often the choice of descriptions depends on the information available and as such also on the type of the system. In contrast to a system that is based on street maps, a system that is designed for the use in a restricted area can contain hand-crafted (i.e. more exact) route descriptions. Bradley and Dunlop [19] have made an observation which should be taken into account when designing navigation systems with speech output: Their study compared the results a group of sighted and a group of blind people performing different navigational tasks. Both groups were tested with descriptions created by blind and sighted persons. Their results indicate that all of the blind participants could navigate faster and with less cognitive load with the help of the directions created by blind persons compared to those created by sighted persons. The results were not quite as clear for the group of sighted participants, however most of them performed better with the descriptions created by sighted people.

Other systems relying on speech output are the MoBIC system [105, 79] and both systems by Hub et al. [49, 51, 52]. Many groups use a simple text to speech output of directions in setups that are focused on other aspects of navigation, e.g. the systems by Treuillet et al. [107], Sánchez and de la Torre [88], Ienaga et al. [53], Ko et al. [57], and Yatani et al. [119].

2.3.2.2 Direction-Giving Sound

Direction-giving sound can be classified into two distinct categories: Sound coming from the desired direction in virtual 3D space (i.e. a pull-strategy) and sound that only corrects the user if he goes off-track (i.e. a push-strategy).

Experiments with direction-giving sound have also begun quite early, in the case of Loomis et al. [64] in combination with speech output, i. e. as speech coming from a certain direction.

In a study that compared auditory and haptic cues it was shown that a simple binary cue with a push strategy is sufficient to guide a user to a desired location, and additionally that auditory and haptic cues perform about equally for this task [70]. In a similar study auditory cues and a handheld pointing device were compared. Both setups were able to guide the user, but none performed considerably better than the other [69].

Another study compared direction-giving sound, speech output and a tapping device. Both head- and body-oriented directions were compared for all three methods. Again, no device performed unequivocally better than the other devices [84, 85].

A study on different kinds of direction-giving sound compared three different kinds of sound (a sonar-like sound, a pure sine wave at 1000 Hz and a pink noise) and three different sizes of waypoints (0.5m, 1.5m and 15m), i.e. the area that needs to be reached during a navigation in order to consider a waypoint as passed [113]. As the study took place in a virtual environment with head tracking it can be assumed, that the sound beacons were head-oriented. While the pink noise performed slightly better, there was no significant preference of one sound type. Hence the main result of the study is that a waypoint radius of approximately 1.5m is a reasonable size.

Other examples of systems using simple direction-giving sound are those by Treuillet et al. [107], and by Ienaga et al. [53].

In his dissertation, Lutz [67] developed a system for the evaluation of auditory cues. In this system, the position of the user is represented by a 4D mouse on a drawing tablet. The mouse reports its 2D coordinates on the tablet as well as its 2D orientation, which maps to the position and orientation of the user. Different auditory cues can then be presented to the user and his feedback can be tracked. The purpose is to evaluate auditory cues in a laboratory environment without the need for the complete setup and test of the navigation system, reducing the effort of the study and eliminating other effects (such as varying GPS signals) on the navigation of the user.

2.3.2.3 Auditory Maps

Auditory maps have not yet been used in navigation systems. However, auditory maps have been explored as a means to teach the spatial environment of a certain area to blind people. Different types of auditory maps exist for different purposes. Heuten et al. [39, 40] use sound sources placed at key points in the map, e.g. a park being represented by bird sounds, a lake by water sounds. Those sounds are then played from the correct point in the 3D space relative to the virtual position of the user. By moving around the virtual position (in this case with a digitizer tablet) the user can thus develop a mental model of the map. The user can determine the range around the virtual position in which he still wants to hear the sounds.

Cohen et al. [26] try to convey graphs through auditory means. To achieve this they represent edges and vertices through sounds. As the user moves the cursor along an edge the distance to the next vertex is conveyed through a modulation in the vibrato of the tone. If the user moves away from the edge, the loudness of the tone is diminished. As Cohen noticed in his studies, this way of conveying graphs is especially useful for graphs that do have underlying spatial (e.g. geographic) information and could, for instance, be used for maps of public transportation systems.

2.3.3 Tactile Cues

Tactile cues have the advantage of not cluttering or blocking the hearing of the user, enabling him to filter the environmental noises for other information such as street traffic. Tactile cues include braille displays, tactile maps and tactile direction-giving.

2.3.3.1 Braille Displays

Braille displays provide the same functionality as speech output for either deafblind users or users who do not want to wear headphones or draw attention to themselves by using a loudspeaker that produces synthetic speech. Both systems by Hub et al. [49, 51, 52] use Braille as an optional alternative to speech output.

2.3.3.2 Tactile Maps

Tactile maps in their original form are one of the most widely used device categories in the area of navigation aids for the blind. Made from metal, wood or plastics those tactile maps are static in their appearance and normally fixed at a location (e.g. at a blind center or a train station), where they represent the immediate environment. In this form, tactile maps are

unusable for mobile navigation systems, yet a possible future importance should not be underestimated because of their intuitive and established functionality. Maps can also be displayed on a pin-matrix display [87]. The system by Zeng et al. [121, 123] uses a pin-matrix display of reduced size and can thus integrate tactile maps into a mobile navigation system.

Other systems try to use quasi-tactile maps as a cheaper alternative. The TANIA system uses a quasi-tactile map which consists of a touchscreen augmented with tactile strips as points of the compass and speech output or a braille display for the name of the touched region [50, 51, 52]. This allows the user to explore the map, by touching different adjacent regions and learning their spatial relations. The TANIA system uses these maps both for the simple exploration of maps and as an output method during navigation.

2.3.3.3 Tactile Direction-Giving

Tactile direction-giving can be seen as the tactile counterpart to direction-giving sound. The current direction is conveyed to the user by tactile means. It is little surprise that studies have included a binary vibrating device, which vibrates only if the user is off-track. Marston et al. [70] place the vibrotactile stimulator in the wristband of one hand. A bit more information is given by those systems that have two distinguishable signals, normally indicating left and right. The actuators have been fixed at several different locations, such as the thumb and index finger [30], in two vibrating bracelets worn on the left and right arm [101], or on the shoulders [84, 85]. The user is tapped on the left shoulder to indicate a left turn, on the right shoulder to indicate a right turn and on the center to indicate that he is on track. In all these papers it has been shown, that a user can be guided successfully by one or two tactile actuators. The SpaceSense system by Yatani et al. [119] uses an array of nine vibration motors embedded into the sleeve of a smartphone, where the vibrating motor conveys the direction, while the strength of the vibration encodes the distance to a destination.

Amemiya et al. [2] use an interface that was originally developed to convey finger-braille to deafblind users [43]. In this interface actuators are placed on the index, middle and ring finger of each hand, representing the six dots of the Braille alphabet. These actuators are then used to give a navigation guidance, by vibrating in a certain pattern. Amemiya et al. found a pattern of stimuli wandering from the ring finger of one hand over the other fingers to the ring finger of the other hand the most usable. The direction is signaled by the direction in which the actuators are activated, while the angle is indicated by the pause between the activation of two actuators. The main advantage of this setup is, that one and the same

device can be used to convey textual information (through Braille) as well as directional information.

Actuators can also be placed on the back of the user. Ertan et al. [28] use a 4×4 array of actuators. Those are used to convey the four cardinal directions and a stop signal. A direction is given by successively activating rows (or columns) of actuators in the desired direction. This method can give the user the feeling of a continuous motion on his back, even if implemented with very few actuators [106].

A haptic belt with actuators around the waist has been studied by several different groups. In the most basic form the activation of one actuator signals the desired direction. This limits the number of actually displayable directions to the number of actuators [108, 110]. To increase the number of directions without increasing the number of actuators, Heuten et al. [41] activate two actuators at once, modifying their intensity. Thus, by increasing the intensity of one actuator and decreasing the intensity of the other, theoretically every possible direction can be displayed. Yet, as is to be expected, the user cannot distinguish between fine-grained differences. Heuten et al. report a mean and median deviation of 15° in observed direction for a belt with six actuators (i.e. a belt that could display in steps of 60° with the conventional setup mentioned above).

Buchmann et al. [21] identify a difficulty with haptic belts that seems to be overlooked by other studies (although briefly mentioned by Tsukada and Yasumura [108]): If the actuators are meant to convey angular information, they need to be placed in irregular distances around the waist due to the form of the human body. Furthermore the position of the actuators needs to be individually adjusted for each person – and possibly even from time to time for one and the same person – as a simple adjustment of the size as it is done with ordinary belts would displace the actuators.

The principle of the GuideCane, which has been developed as an obstacle avoidance device, could easily be applied to a full-fledged navigation system. The GuideCane is a cane that has a small, two-wheeled cart fixed at its distal end [16, 109, 99]. The cart is normally pushed forward by the user, but the system can influence the direction of the cart by steering its wheels with servo motors. The user will then automatically follow the lead of the cane. While the GuideCane is an innovative and promising idea, its main problem is that it cannot overcome steps and curbstones by itself.

Kulyukin et al. [61] proposes a robot-assisted navigation system. The robot moves independently and the user handles a leash that is connected to the robot. The user is thus able to follow the lead of the robot. As the robot is wheelbound, the same restrictions regarding steps and curbstones as for the GuideCane system apply.

Amemiya et al. [3, 4, 1] have proposed a kinesthetic device which would

	relative		absolute	
	auditory	tactile	auditory	tactile
direction-based	[64] [84] [85] [69] [113] [70] [107] [67] [53]	[16] [28] [84] [109] [85] [43] [99] [2] [108] [3] [110] [61] [70] [4] [21] [30] [41] [101]	[119]	[119]
text-based	[105] [79] [84] [85] [107] [53] [88] [57]		[50] [51] [52] [121] [123]	[50] [51] [52]
map-based			[26] [39] [40]	[50] [51] [52] [121] [123]

Table 2.2: Directioning Systems. The three-dimensional representation from Figure 2.1 has been flattened in order to maintain readability. Entries are sorted according to year of publication, numbers indicate alphabetical order according to first author.

give the user holding it the illusion of being drawn into a certain direction.

2.4 The TANIA Navigation System

The TANIA system by Hub et al. [50, 51, 52] deserves a special mention here, as it forms the basis on which much of the work presented in this thesis is built. Many of the designs presented in chapters 3–10 were first developed as enhancements to the TANIA system and could thus be tested easily. As the TANIA system predates the work presented in this thesis, it is not elaborated upon in the respective chapters. Instead, in order to ease the understanding of later developments, a short overview of the system is given here.

The TANIA system is built of an Ultra-Mobile PC, equipped with a GPS sensor, a combined 3D compass, gyroscope and accelerometer (XSens

Figure 2.2: The TANIA Navigation System. The displayed map, acceleration sensor (on strap), RFID reader (left) and GPS receiver (top) can be seen.

MTx), and an RFID reader (Figure 2.2). The MTx is placed on a strap and positioned at the neck of the user when the device is worn.

2.4.1 Positioning

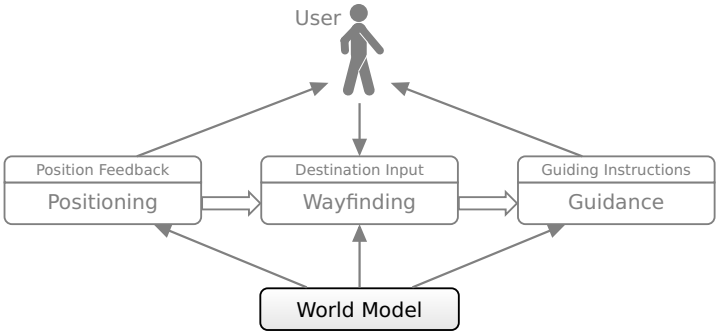
Positions inside a building are tracked by direction and accelerometer information, which is used to estimate the length of each step, by measuring vertical acceleration. As the tracking is only relative to a starting point and its accuracy deteriorates gradually, the position can be corrected at RFID tags placed at certain points in the building. As the range of the RFID reader is approximately one meter, the tags need to be placed at strategic positions, like doorways. Positions can also be corrected by the user if he is sure of his current whereabouts. Outside, positioning can be synchronized with GPS positions, either constantly or upon request.

2.4.2 The Tactile-Acoustical Map

After the position has been determined, feedback about the surroundings can be given to the user. For this purpose TANIA features a tactile-acoustical map. The touchscreen of the UMPC is augmented with tactile strips, which indicate the map center and the four cardinal directions and help the user to orient his finger on the screen. Any map information can then be projected onto the screen, and a tap with the finger will provide the user with text-to-speech feedback about the underlying map information and the distance to his current position. Thus, the user can gain a spatial impression of the environment by exploring the map. The map allows the display of hierarchically structured information, which is communicated outwards from the center, i.e. if the user taps a specific location, the system informs him about the nearest enclosing map structure, e.g. the room number. If he taps the same location again, the system will announce increasingly larger enclosing structures, e.g. the building wing and then the building itself. To support access to the map, we provide two modes: The virtual and the navigation mode. In navigation mode, the user is always at the center of the (zoomable) map and can therefore gain an immediate impression of the area surrounding him. In virtual mode, the user can zoom and scroll freely and therefore explore other areas that he is planning to visit.

2.5 Summary

As the research in the past 20 years has shown, it is an attainable goal to provide blind people and other special user groups with a navigation system that allows for independent navigation in unknown environments. Past research has often concentrated on two necessary parts: Positioning and guidance. Positioning is still a problematic area, as there is so far no system that provides for ubiquitous positioning and has the required accuracy for a longer period of time. On the other hand, it has been shown that accurate guidance can be achieved with several different techniques. This thesis contributes to the efforts in positioning and guidance, but concentrates on areas that have not yet been the focus of research in this area, especially the world model and its effects on the stages of navigation.



3 Aggregation of Spatial Data

As stated in Section 1.3, one of the main problems with navigation systems for specific user groups is that an individualization is required on a large scale. Until now, the underlying data model of a navigation system has mostly been referred to as “world model”, which is in many cases consists of maps. However, a homogeneous and untransformed world model provided by a single source is by its very nature in conflict to the aspired goal of individualization.

Aggregating information from different sources is one possible way to tackle this problem. Different sources are likely to emphasize different aspects of the data, and are therefore of different relevance to specific users or tasks. Additionally, data aggregation can help to integrate information that is not directly available in maps, but which could still be beneficial for the users, such as up-to-date information on the state of the environment (e.g. defunct elevators or construction sites, see Section 1.2.2) or timetables of public transport. Often, the desired information can be found in the world wide web. However, the unstructured nature of the web makes it difficult to locate the specific information, especially for a blind user who is at the same time trying to navigate in an unknown environment.

For this reason, a system was developed, which automatically integrates a number of different information sources, and uses several kinds of maps, depending on their availability for a certain area. In addition to these maps text-based information is integrated into the system. Section 3.1

Parts of this chapter have previously been published in:

Bernhard Schmitz, Susanne Becker, André Blessing, and Matthias Großmann. Acquisition and presentation of diverse spatial context data for blind navigation. In *12th IEEE International Conference on Mobile Data Management (MDM 2011)*, pages 276–284, 2011. DOI:10.1109/MDM.2011.66

explains the classification of data that can be used for navigation. The concrete architecture that is used in the system is presented in Section 3.2. Section 3.3 provides examples of information sources that profit from such an architecture and are integrated into the system. A typical scenario for such a system is discussed in Section 3.4.

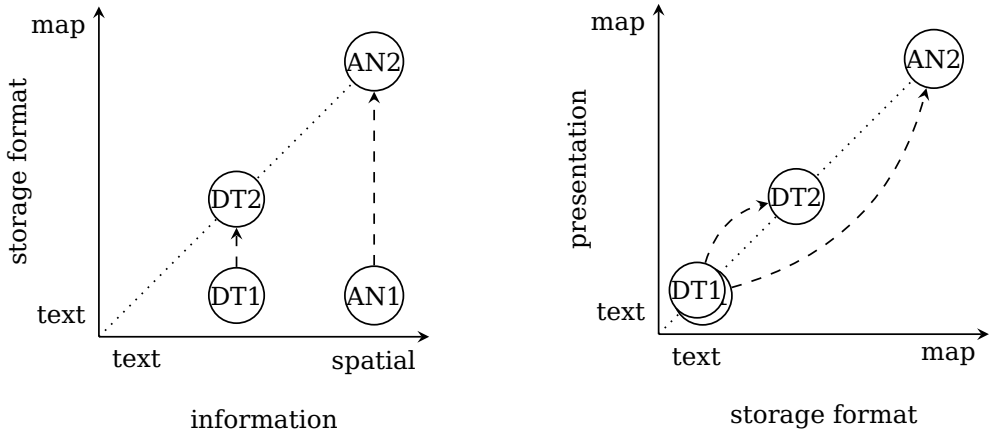
3.1 Data Classification

Static maps are unfit to store additional non-static information about the environment that could be of great benefit to the user. As already mentioned, much of the required information can be found on the world wide web, such as timetables, information about construction sites, etc. However, this information is distributed across lots of different and often inaccessible websites making it impossible—especially for a blind person—to get the required information when it is needed.

This information is normally tied to a specific location, and therefore a navigation system is a very appropriate device to present it. However, there are fundamental differences between different kinds of text-based location dependent information. These differences cover the format they are stored in, their inherent spatiality, and the way in which they need to be presented to a user.

The inherent spatiality requires some additional explanation. It is not always self-evident, if a piece of information is primarily a textual information or a spatial information. As an example, information about an ordinary street normally consists of its name, location and course. However, the name as such is normally not the information a user is looking for, and it might even be quite useless if not tied to additional information. It could therefore be said that the inherent spatiality of the information about the street is relatively high. On the other hand, a timetable, e.g. at a bus stop, even though tied to a location, is not mainly a spatial information, the user is interested in the departure times, and the location is a means to ensure that the correct timetable is selected. Accordingly, its inherent spatiality is relatively low.

Normally, the spatiality of an information, its storage format and the way it is presented to a user are directly related, i.e. an information of high spatiality is stored and presented as a map, whereas an information that is highly textual is stored and presented as text (see Figure 3.1). However, as will be seen in Section 3.3.2.1, this is not always the case and sometimes, information of high spatiality is stored in text format. Nevertheless, the presentation to the user should reflect the inherent spatiality and not the storage format. As the presentation is usually directly related to the storage



(a) Relationship between inherent spatial information and storage format.

(b) Relationship between storage format and presentation.

Figure 3.1: Relationship between Inherent Spatial Information, Storage Format, and Presentation. The storage format should be chosen along the dotted line of (a). Normally, the presentation follows automatically along the dotted line of (b) (a text file cannot be presented as a map). In the case of colloquial area names (AN), the existing information is stored in pure text format, however, the information is highly spatial (AN1). By applying the method explained in Section 3.3.2.1 the storage format is transferred to a polygonal map (AN2) thereby allowing a corresponding presentation, as seen in (b). In the cases of Departure Times (DT, Section 3.3.2.2) and Acquired Live Content (Section 3.3.2.3), the information is again stored in pure text format. However its inherent spatiality is not as high as that of area names (DT1). This corresponds to the fact that the result of the transformation is still textual in nature and just tied to a specific map object, and not polygonal itself (DT2).

format, it is a logical decision to change the storage format in order to change the presentation. Because of this decision it is possible to use a single server-side data provider (the Nexus platform, see Section 3.2.1) as the sole world model for the system while still using a variety of data sources and thus allowing at least some adaptation for different scenarios.

3.2 System Architecture

Figure 3.2 shows the architecture of the system. It correlates with the general schema presented in Figure 1.1 on page 3. The concrete implementation uses a client-server architecture, in which the server provides the world model. As a server, the Nexus platform, which was first presented

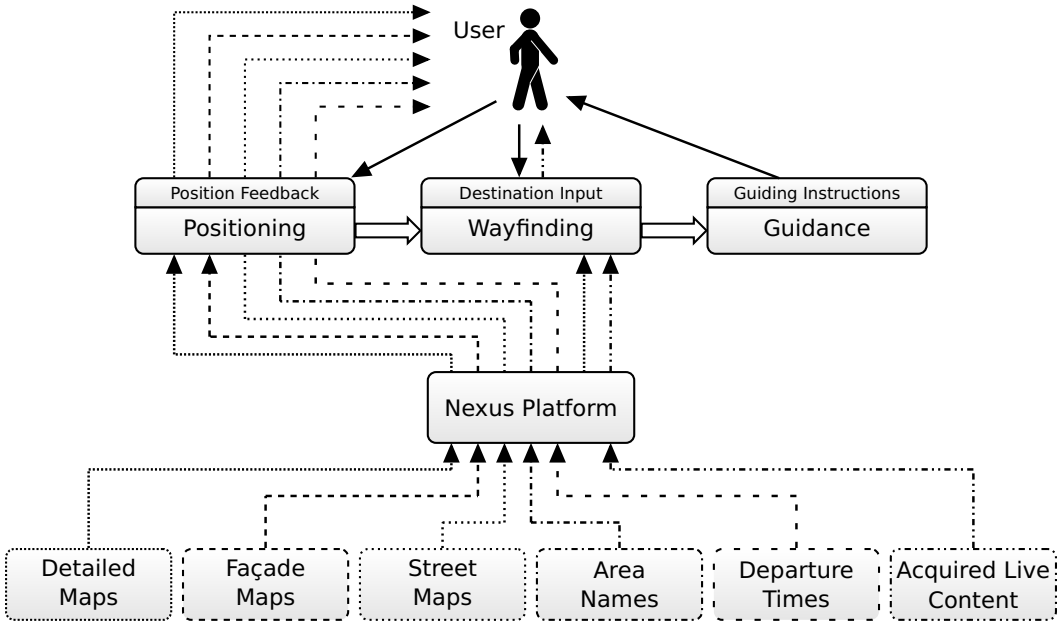


Figure 3.2: Correlation of the General Navigation System Schema with the Architecture of the Aggregation Approach. The uniform world model is replaced by a variety of information sources, with differing influence on the system. The user can influence the positioning component and may receive feedback from the wayfinding component, if the acquired live content necessitates it. Detailed maps and façade maps can influence positioning by collision detection.

by Nicklas et al. [77], is used. It integrates different information sources, which will be described in detail in Chapter 3.3. Because of this architecture, the integration needs to happen only once and the resulting information can be used by any number of clients. Positioning, wayfinding and guidance are handled by the client system, and will be explained in the respective chapters of this thesis.

3.2.1 The Nexus Platform

The Nexus platform was used as the base for this system due to its being designed with extensibility and different data sources in mind [71]. It is a platform that was designed to support many different types of location-based aplications. The three-tier architecture of the platform and how the navigation system ties in to it is shown in Figure 3.3. The spatial data is stored on context servers, which constitute the service tier. A context server registers the area that it covers with the area service register in

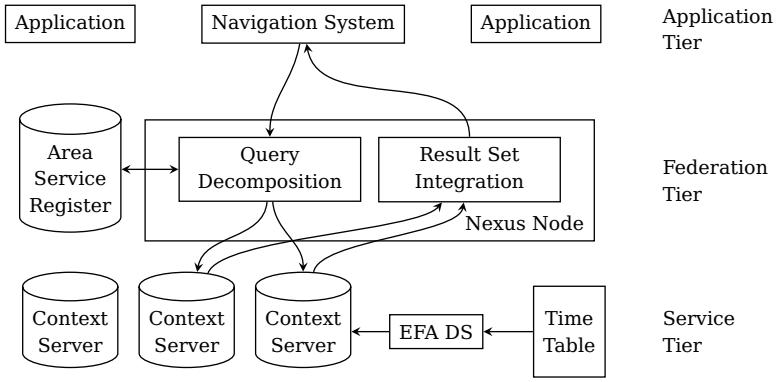


Figure 3.3: Architecture of the Nexus Platform. It consists of three tiers, the applications on the top, the systems providing data at the bottom and a federation tier in between, which distributes queries and integrates results.

the federation tier. The Nexus Nodes in the federation tier can therefore distribute a spatially restricted query to the relevant context servers and combine the result sets of the context servers. The single combined result is then sent back to the application that triggered the original request.

The Nexus platform uses the Augmented World Model (AWM) to represent spatial data [76, 45]. The AWM allows developers to define their own schemas, which are called Extended Class Schema (ECS). All types of an Extended Class Schema are derived from types of the Standard Class Schema (SCS), which is provided by the AWM. The ability to create a custom ECS was used heavily for the federation of the different data sources described in Section 3.3. Additionally, the support for multi-attributes by the AWM was used. An object can contain several instances of the same attribute, distinguished by meta-data that describes the applicability of the attribute, e.g. its temporal validity.

3.3 Information Sources

The information that is federated by the Nexus platform can be classified into two distinct categories: Maps and text-based spatial information. Maps have a very high inherent spatiality according to the classification in Section 3.1, but are normally stored in an appropriate format. Text-based spatial information, on the other hand, can be of varying inherent spatiality and also varies in storage format.

3.3.1 Map-Based Information

As mentioned in Section 1.2 (Problem Statement), one of the main difficulties for navigation systems for special user groups is the non-availability of specialized maps or world models on a larger scale. The design of the system in this chapter makes it possible to alleviate the problem by employing the strategy of providing specialized maps wherever they are available, but having fallback solutions available for other areas.

3.3.1.1 Detailed Maps

Detailed Maps are the core of the system. They were originally created for the TANIA system described in Section 2.4. The TANIA system used these maps, which were stored locally in the Geography Markup Language (GML) format, as their only world model. They are comprised of polygons which are annotated with information (normally the name) that can be presented to the user. For use with the Nexus platform, an ECS was created, which reflects the objects that were represented in GML. In addition to the objects present in the real world, the maps include purely virtual navigation areas, which are areas connected to certain additional information that might be helpful for a blind user, for example areas denoting a coordinate system in an open space without other navigation cues. The maps were built manually, often based on maps received from institutions, such as the University of Stuttgart. Figure 3.4 shows one example of such a map.

As these maps are the main means of navigation, they are cached locally on the navigation system. This means that a request about all map objects in a certain configurable area around the user is sent to the Nexus platform, the response is parsed and all interaction with the map objects occurs locally. Whenever the user gets near the border of the cached area, a new request is started. All parsing is done by a separate background thread, and only once a new area is loaded completely, it replaces the previously loaded area.

All map objects have an attribute that designates the floor to which the object belongs. It was decided to use a string attribute in order to be able to use floor names from the real world, which are not always integers. Consequently, the layering of the floors is not inherent to the attribute and thus unknown to the system. Therefore, “hypergates” have been introduced, which are polylines that have an attribute which references the connected floor and are placed at staircases, ramps, etc. Upon crossing the polyline, the system will automatically switch to the respective floor. As floor-switching can (in contrast to leaving the cached area) occur unpredictably, all floors of an area are cached indiscriminately. Additionally, as

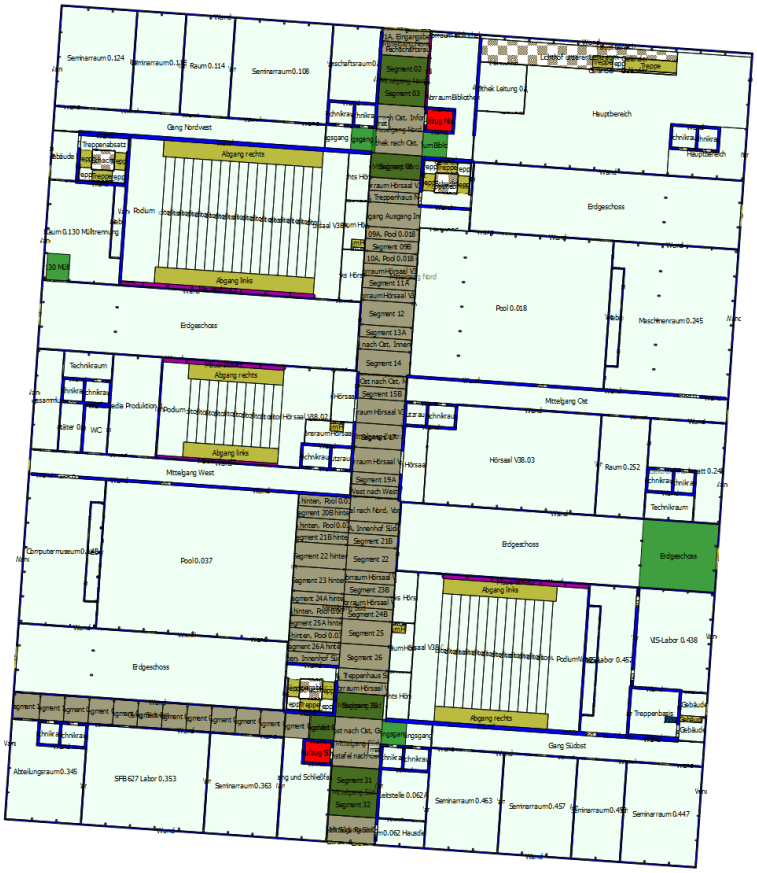


Figure 3.4: Example of a Detailed Map. The map displays the ground floor of the computer science building of the University of Stuttgart (north at the top). The main floor has been divided into segments that also contain information about adjacent rooms.

with the TANIA system (Section 2.4) floors can be automatically switched at RFID tags, which is necessary for situations where hypergates are not applicable, e.g. at elevators.

As indicated by the arrow tips in Figure 3.2, the detailed maps are used for positioning in addition to user feedback. A collision detection is used to make the positioning more accurate. For this purpose, the map polygons have an additional flag that indicates a possible collision (i.e. a real object like a wall) in contrast to purely virtual polygons.

Due to the manual creation process, these maps are a good example for the dichotomy between usefulness of the maps for specific users and their widespread availability, as mentioned in Section 1.2.1: They work very well in a navigation system for blind people, but are only available at specific location. It is therefore necessary to substitute them with other information as well as possible.

3.3.1.2 Street Maps

Standard Street Maps are commonly used, be it in car navigation systems or commercially available GPS navigation systems for blind people. Although they are inferior to the detailed maps, they can still provide valuable information when these are not available. By using the street maps the user is at least able to roughly know his current whereabouts, for example when leaving a bus. Therefore, when there is no detailed map information for the inquired location, a simple nearest-neighbor-query to the Nexus platform is started. The street's name is then parsed from the response and read to the user.

3.3.1.3 Generated Façade Maps

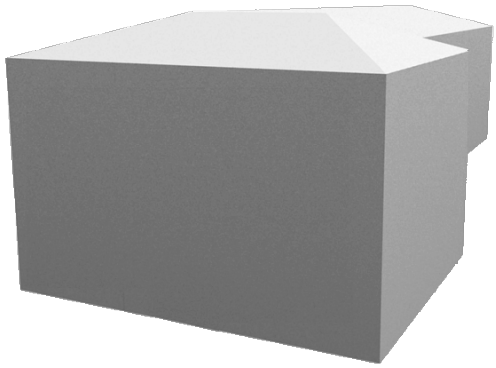
Even though accurate buildings maps are available for some buildings, it is still desirable to have information about other buildings, even if the information is not as detailed as with the manually built detailed maps. Therefore, generated façade maps were also included in the system. The 3D façade reconstruction algorithm by [10] was used to create a 3D model of a building from 3D point clouds obtained by terrestrial laser scanning. From this 3D model, a 2D map can be derived that includes annotated windows and entrance polygons. This can be helpful information for blind people looking for the entrance of a particular building. Figure 3.5 shows an example of such a generation of a 2D map.

3.3.2 Text-Based Location-Dependent Information

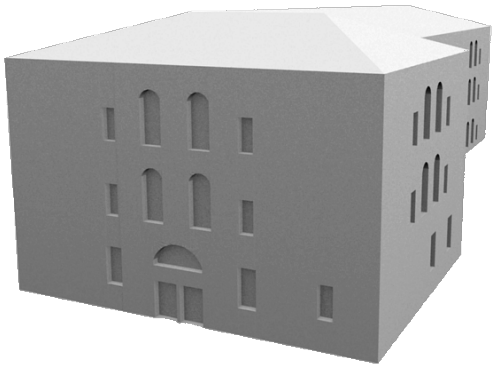
As mentioned before, the inherent spatiality of textual information varies. The general approach for transforming the textual information into something that can be stored on a Nexus server is called Text-Sensor [15]. Either a specific web site has to be set as the source for the information, or information retrieval techniques can be used to find the relevant information. After that, Natural Language Processing methods are used to retrieve the actual information. The following sections give a short overview of three information sources and the respective algorithms that were used to further enrich the world model stored in the Nexus platform.

3.3.2.1 Colloquial Area Names

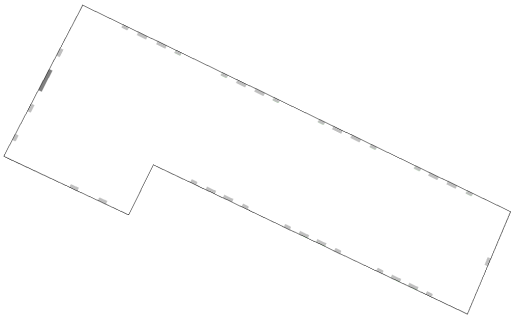
In many cases, available maps represent a specific, "official" convention of naming of areas. Often, these refer to geo-political entities (suburbs, cities, states), but neglect more unofficial names used by people, such as functional descriptions (e.g. financial district), vernacular names (e.g.



(a) Façade before reconstruction.



(b) Façade with windows and entrances, after reconstruction.



(c) 2D map of the building, created from the reconstructed 3D model.

Figure 3.5: Façade Reconstruction. A 3D model of the building created by terrestrial laser scanning (a) is augmented with windows and doors created by a combination of laser scanning and façade grammars (b). The result can be flattened and use as an indication of building entrances where better maps are not available (c).

Bohnenviertel) and spatial names (e.g. Stuttgart-West). Integrating those names into the navigation system allows the user to connect conversational information with the map presented by the system. The method used in the system to extract such names from textual descriptions was developed by Blessing and Schütze [14]. The method uses standard web search engines to search for such names. The textual content of the resulting pages is then stored, and compared with street maps (Section 3.3.1.2), in order to collect all street names in the text for a specific colloquial area name. The street names need to be disambiguated, because some names are equal to common nouns. Afterwards, the streets are clustered. The convex hull of the largest cluster of streets can then be assumed to approximately represent the area associated with the colloquial name and can be added

to the world model. It should be noted that in some instances several clusters could actually be the correct solution, such as in cities with several University campuses, such as Stuttgart.

3.3.2.2 Departure Times

In order to be able to present departure times at public transport stops, it is necessary to use information from the responsible public transport organization. In the case of the system described in this chapter, information from Stuttgart's transport association, the Verkehrsverbund Stuttgart (VVS), was used. For users of mobile phones with small displays, the VVS provides Wireless Markup Language (WML) pages for each public transport stop, with information about the departure times of trains, trams and buses for the near future. Those pages are calculated according to the current information level of the VVS itself and are not to be confused with static timetables.

Another ECS was created that defines departure tables, which are simple objects with a position and attributes holding the information about departure times. A small daemon program on the context server (EFA DS) retrieves the WML page, parses it updates the respective departure table object.

3.3.2.3 Acquired Live Content

Navigation is partially dependent on the status of utilities such as elevators and escalators. In some cases, relevant information about their current status is available on the web, for example the VVS provides a web site with information about non-working elevators and escalators and other blocked accesses to train stations and tram stops.[1] Those messages are written in natural language and therefore not easily parsed. Natural Language Processing Methods are needed. In this case a grammar for a finite state transducer was developed to acquire the information. The method used here was developed by [15]. The extracted status is then added as an additional attribute to the object in the World Model that was determined to be referred to by the message. Additional information from the textual content, which is linked with information from the world model is used to deal with ambiguities, i.e. to determine the most probable object being referred to. Nonetheless, an approach such as this can never be error free, and it has to be decided by the user how to handle such uncertain information.

[1] http://www.vvs.de/aktuelles/verkehrsmeldungen/aufzuege-und-zugaenge/ (last accessed 2014-10-03)

3.4 A Typical Application Scenario

To demonstrate how the different information sources work together to enable the user in his navigational tasks, a typical scenario, in which the system can be used, is presented in this section. In order to emphasize the general benefit of the system, the scenario is not located at a specified area. The scenario is therefore fictitious in regard to the mapped locations, however, the technology can be used in the described way.

Suppose that a blind user of the system is in a hotel in a city unfamiliar to him and wants to visit a museum[2]. Knowing that the city has a museum quarter, the user would search for it on the map of the system. Now the museum quarter might not be a part of the provided maps under that inofficial name. However, the algorithm for colloquial area names (Section 3.3.2.1) has identified the area that is commonly referred to as the museum quarter and added it to the Nexus platform. Therefore, the user can easily identify it. By exploring the area that is marked as the museum quarter with his fingers, he learns that the area has not been fully mapped yet. So he will have to rely on simple street maps to get from one location to another (Section 3.3.1.2). He can also notice that most of the museums only have façade outlines, while one has a completely mapped interior. Let us assume that the user decides to visit one museum that is of particular interest to him, and additionally the fully mapped one, because of the convenience.

As the museum quarter is too far away from the hotel he has to use the commuter train to get there. The system allows to search for the nearest commuter train station and query the departure times (Section 3.3.2.2). If an adequate service is provided by the commuter train operator, this information is not just based on timetables but actual approximated departure times depending on the current position of the train in the network. Based on that information, the user can for example decide to have another coffee in the hotel lobby, instead of having to wait at the station. In order to find the way to the station, the user has to rely on standard street maps, as there is no special mapping for this area. This certainly is something for the more adventurous users and similar to the use of a current commercial GPS navigation system for blind people. Arriving at the station, the user is informed that the elevator that leads to the platform is out of order. This piece of information was present in textual form on the website of the commuter train operator and automatically integrated into the Nexus platform by the live content algorithm (Section 3.3.2.3). On the platform

[2] Either for touchable exhibits or because a user prefers the social interaction to hearing the explanatory texts at home

the user can again rely on the departure times in order to catch the correct train.

Once arriving at the museum quarter the user can find the way to the entrance of the façade-mapped museum (Section 3.3.1.3). At the museum that is completely mapped, he can also use the system inside. Here, the system can not only be used for navigation, but also to convey information about the items on display. The user can listen to additional information about the items on display at any location that has been enriched with metadata by the museum.

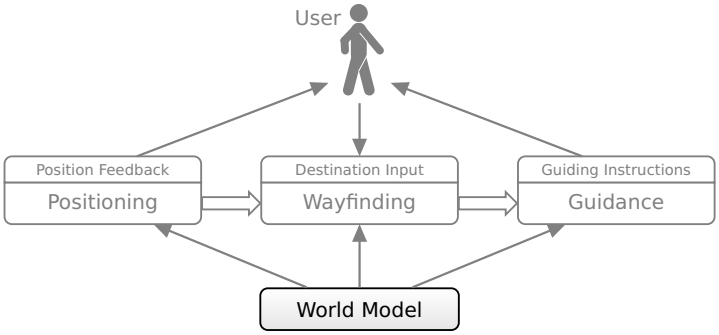
3.5 Discussion

The system presented in this chapter tries to solve some of the problems discussed in Section 1.2 by providing a common platform that aggregates information from various sources and thus makes it possible to create a navigation system that acts as a single interface to this varied spectrum of information. A quantitative evaluation of these novel aspects is rather difficult due to the design goals of the system: A success—data from different sources displayed seamlessly and without a possibility of ascertaining their source for the user—cannot be measured in any meaningful way by a user study.

However, the approach has several problems:

- Individualization is only possible by adding specialized information for specific user groups into the Nexus platform. This means that unless a specific requirement is considered by the person adding the information to the platform, the requirement cannot be satisfied.
- The system employs an approach with specialized data and fallback solutions to overcome the dichotomy between having specialized maps for specific areas and having more general maps on a wide scale. It thus combines the two distinct approaches mentioned in Section 1.2 that were available previously, but does not offer a solution that allows for specialized maps on a wide scale.
- The system in its current state uses only detailed maps and some acquired live content for wayfinding. Street maps could easily be integrated into the wayfinding as well. However, each navigational task can only be performed on objects designed for that task, it is not possible to create variants of one and the same object with differences tailored to specific tasks.

The next chapter will therefore introduce a different approach to the same problem that tries to overcome these problems.



4 Map Content Transformations

Even if the system presented in the last chapter was able to tackle some of the challenges and questions put forth in Chapter 1 (Introduction), there are still several aspects missing, as mentioned in Section 3.5. This chapter presents a new approach to world models in navigation systems for special user groups, which is designed to address many of these problems and provides benefit to all stages of navigation. The specific benefits for the stages of navigation will be dealt with in the respective chapters later on.

The general idea of the approach is that everybody uses the same world model as a basis. A preprocessing step is introduced before the use of the world model, in which rule-based transformations are applied to the data. These transformations take the original data, which is equivalent for everyone, and transform it into a new world model that is individualized for the specific user and navigational task (Figure 4.1). With this approach, there is a unified world-model, that is ideally available world-wide, yet the world model for an individual user is still adapted to his needs and can fit specific requirements.

OpenStreetMap was chosen as a community based world model for this approach. This has several advantages. OpenStreetMap is available world-wide, and is often a better alternative than commercially available maps, especially for pedestrian navigation. In 2011 OpenStreetMap provided a

---

Parts of this chapter have previously been published in:

Bernhard Schmitz and Thomas Ertl. Rule-based transformation of map data. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops 2012)*, pages 584–589, 2012. DOI:10.1109/PerComW.2012.6197581

Bernhard Schmitz and Thomas Ertl. Creating task-specific maps with map content transformations. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on MapInteraction (MapInteract '13)*, pages 84–90, 2013. DOI:10.1145/2534931.2534945

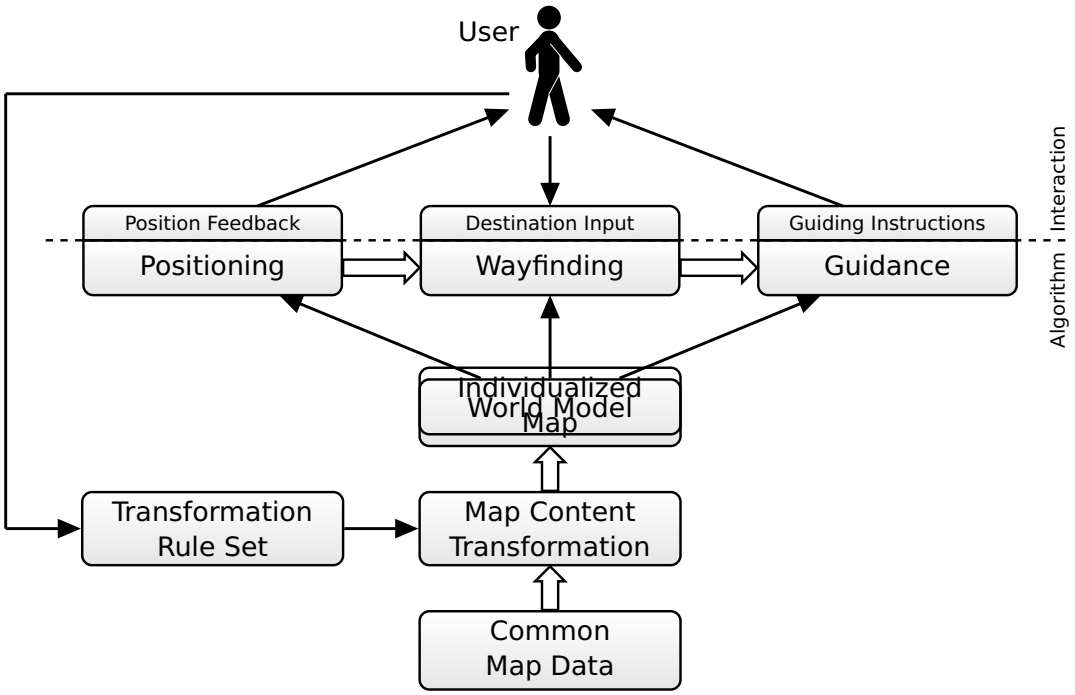


Figure 4.1: General Idea of Map Content Transformations. The common map data is the same for all users. A rule set, which can either be one of many predefined sets for a variety of groups, or specifically individualized, is chosen and transforms the content of the world model so that the resulting map is individualized according to the requirements of the user or adapted to a specific task.

more than 30% larger street network for pedestrian navigation in Germany than the commercial TomTom Multinet 2011 database [74]. And even though commercial databases are also continuing to grow, using OpenStreetMap has the added bonus that users can submit additional data at any time.

Another advantage relevant to the topic of this thesis is that anyone can enter additional data into OpenStreetMap, and the format of the data is open enough that even information that was not originally intended to be entered can nevertheless be added. That makes OpenStreetMap a good choice as a world model for a navigation system for disabled people: Information that is useless to other people, could be quite relevant for specific user groups. It is even possible that people without a specific disability might not know which information is relevant for the users with that disability. A similar effect has been observed by Bradley and Dunlop [19], who showed that navigational information created by blind people is better suited to help other blind people navigate than information created

by sighted people. A single-sourced world model is therefore likely to miss such specific information, whereas with OpenStreetMap it can always be added later on.

This chapter describes Map Content Transformations, a new, rule-based scripting language, which was developed as a concrete implementation of the aforementioned approach for OpenStreetMap. Nonetheless, the general idea is in theory applicable to all kind of world models and not limited to the specific case of OpenStreetMap.

4.1 Background

There are several existing techniques that deal with the adaptation of maps for specific circumstances. There are map renderers such as Mapnik and Maperitive that convert map data into graphical output, normally pixel- or vector-based graphics [82]. These renderers feature specific languages that let the user determine the style of the output. The style can be heavily customized, resulting in different maps for different activities, such as OpenStreetMap's road, cycling and public transport maps, or according to regional differences, where different frontends for different countries display the road network according to the country's established mapping style. As an example, OpenStreetMap.com renders motorways blue, whereas OpenStreetMap.de renders them in red and yellow—adhering to conventions of drawing motorways on English and German paper maps, respectively[1]. There are two important aspects with these renderers: First, they only change the style, i.e. the outward appearance, of the map, whereas the layout and the topology of the map stays the same. Second, the output is a graphic and can therefore not be used in applications that rely on map data in vector format.[2]

Another technique is map generalization. Map generalization in its original form deals with adapting maps to different scales by reducing complexity while retaining essential characteristics [115]. As such, it has been around even before the advent of geographic information systems. Map generalization can be seen as an optimization problem. A predetermined algorithm is applied to a specific set of map features while trying to fulfill certain constraints, e.g. a maximal distance from the original data. In this form, the optimization problem is local. Several generalization algorithms can also be combined, where the sequence of their application

[1] last accessed: 2014-10-03

[2] Vector-based graphics created by renderers are optimized for visual impression, not for further computational use.

becomes important, thus creating a global optimization problem calling for combinatorial optimization techniques [75].

While most work on map generalization remains in that area, some projects have aimed to include user information or the user himself in the generalization process. The XSLT-based approach by Lehto and Kilpeläinen [63] combines simple selection and filtering methods implemented in Extensible Stylesheet Language Transformations (XSLT) with other generalization methods, implemented as hardcoded extensions. Burghardt et al. presented a platform for generalization algorithms. Algorithms can be added as plug-ins and the required input data for the algorithms are specified as XML-based schemata. Users can select the desired algorithm and enter parameters in a UI that is dynamically created from an Extensible Markup Language (XML) interface description that is part of the plug-in [22]. Kulik et al. expand regular generalization algorithms with parameters based on meta-information about the user, so that e.g. streets that are of little interest are more simplified than those of higher interest to the user [60].

Other work regarding the personalization of maps is a bit further removed from the classic map generalization technique and consists of a specific system that offers different options for adaptation, such as highlighting specific features [5], changing the graphical style [81], selectively showing or downloading specific features while hiding other features from the user [35], or automatically determining adaptations based on location and user interests and adaptively displaying additional information [68].

The approach that is implemented by Map Content Transformations is not directly related to map generalizations. There is no optimization regardings specific constraints. To the contrary, the result of a scripted map transformation could well be “worse” regarding the constraints of classic map generalization. With standard map generalizations, topological consistency between the original and the result is often required or at least strived for [60], whereas for the individualization of maps it might well be desirable to create an entirely new topology that is only connected to the original topology on a more abstract level. Additionally Map Content Transformations do not aim to be fully automated and influencable only by parameters. Instead, the whole personalization of the map is fully scriptable by the users. This means that the user can write rules that apply to specific “situations” in the map. In this context, “situations” refer not only to map entities as such, but also to combinations of map entities fulfilling certain conditions, such as “an intersection of a main road with a residential road”, “a pedestrian crossing on a main road”, “a public library near a train station”, or any combination thereof. The result of these conditions is then changed according to geometric rules predetermined by the scriptwriter. This allows for almost complete liberty on what to do with the map and does

not constrain the user to predetermined algorithms. On the other hand, the result will in all likelihood not be as good regarding aesthetic criteria as the result of a classic generalization. However, it is possible to use the output of Map Content Transformations as an input to a classic generalization or to a renderer. Thus, they are complementary to the existing map generalization techniques and could even be added to existing systems such as Burghardt et al.'s [22], if the entire script is considered as a parameter.

There is some previous work on the scripted creation of geometry [33, 55], however in these cases geometric objects are scripted from the ground up, whereas for map transformations geometric objects first need to be identified and then the scripting has to be applied in relation to these already identified objects.

4.1.1 OpenStreetMap XML format

An understanding of the data format of OpenStreetMap greatly simplifies understanding Map Content Transformations. Therefore, a short introduction into this data format is given in this section. OpenStreetMap data consists of only three basic types: *Nodes*, *ways* and *relations*. *Nodes* are simple points on the map. *Ways* are polylines, i.e. tuples of *nodes* that connect the *nodes* in the order of the tuple. Due to the ordering of the tuples, each *way* has an inherent direction, even if this direction is normally not used (except for one-way streets). *A* node can be part of several *ways* (e.g. to denote an intersection), and a *way* can even contain the same *node* several times. The most used example for this is a *way* that has the same *node* as its first and last member, thus becoming a polygon. *Relations* are ordered lists that can contain *nodes*, *ways*, and other *relations*. Each member of a *relation* can optionally have a string denoting its *role*. All three basic types can have an arbitrary number of key-value string pairs, called *tags*. In those tags the type of the object and any additional information is stored. What those key value pairs really mean, is not formally standardized. The meaning is instead determined by convention, which is discussed and defined in the OpenStreetMap Wiki[3]. The values are not checked in any way for consistency or adherence to the conventions, and mapping styles tend to differ across regions. When stored in XML format, these object types, are stored with the elements `<node>`, `<way>` and `<relation>`, with `<tag>` storing the key-value pairs in the attributes `k` (key) and `v` (value).

[3] wiki.openstreetmap.org

4.2 Requirements of Map Content Transformations

In order to be usable for the intended purpose, i.e. to transform OpenStreetMap map data into individualized forms, Map Content Transformations need to satisfy several requirements, regarding the specification of the transformations, the result, and the transformation process itself:

Flexibility. The transformations must not be hardcoded, i.e. there must be a flexible scripting system.

Ease of Use. The transformation language should enable as many people as possible to create scripts. Ideally an understanding of the OpenStreetMap data format and of High-School level geometry should be enough.

General Applicability. When specifying what needs to be done to the map, the writer of the transformation must not refer to concrete objects (e.g. by location), as the transformation should work anywhere on the map. Instead, the writer should be able to refer to a more abstract level of mapping, i.e. specific combinations of map objects and tags or “situations” on the maps that appear in many places.

Construction of Geometry. It should be possible to freely construct new geometry based on the geometry that is already existing in the map.

Shortcuts. There should be easy to use shortcuts for the creation of complex geometry that is often needed.

Consideration of Topology. It should be possible to influence the topology of the created geometry, as topology is important for some stages of navigation, e.g. wayfinding.

4.3 Existing Alternatives

If there is a solution available that fulfills all requirements, this solution could be used to achieve the desired results. The XML format of OpenStreetMap seems to imply the use of XSLT. XSLT is a language that transforms XML documents into other XML or text documents. XSLT can thus be used for transforming OpenStreetMap data, e.g. one could easily create new tags according to rules based on existing tags.

However, XSLT does not fulfill all the requirements for Map Content Transformations. Especially the creation of new geometry based on existing geometry is almost impossible with pure XSLT. As an example, a two dimensional polygon of a street should be created from the single line tagged with its width. But even more complex geometry can be created by scripts,

when being able to perform geometric calculations, e.g. measuring the distance between two points or the angle between two streets and having the ability to construct new points and lines based on those measurements. For example, in order to create a route graph that connects two sidewalks at a zebra crossing, it is necessary to build a new route perpendicular to the street at the position of the node that is tagged as a zebra crossing. The resulting line then needs to be intersected with the sidewalks that have been created parallel to the road on both sides. Unfortunately this construction of new geometry is not possible with XSLT, which is designed to deal with text data and not two-dimensional geometric data.[4] XSLT Extensions could provide a solution, which is also used by Lehto and Kilpeläinen [63]; however, these require the user to have deep programming skills and are problematic to provide by a user, if the transformation is to be done on a server.

In order to circumvent these problems it was therefore decided to implement a new scripting language for map transformations. The language is based on the XML format of OpenStreetMap in order to be easily understandable by people who use OpenStreetMap and allows scripting rule-based map transformations of arbitrary complexity.

4.4 Structure of a Transformation

Several important aspects need to be addressed by a transformation:

Transformation Target. Some transformations only need to add some additional geometry into the already existing map, while others must create completely new maps. Therefore it is necessary to be able to specify the target of the transformation: The original map, or a new one.

Identification Rules. If the transformation is to be universally applicable, it is not to be tailored to a specific map. Therefore, a transformation needs to identify geometry independent of its location, based on a specification of structures and features. Identification rules are therefore applied on the entire map.

Construction Rules. Construction Rules specify how new map features are created or existing ones are changed. Construction rules are only applied to the entities identified in the corresponding identification rule.

[4] In theory, XSLT is Turing-complete and therefore able to handle such transformations, however this would be impractical to the extent of impossibility regarding the implementation of the XSLT scripts and result in problematic performance.

Identification Source. Sometimes it is necessary to further transform geometry that has been the result of a transformation. Therefore it must be possible to specify whether the identification is to be applied on the original map, or on the result of previous transformations.

Evaluation Order. What is meant with "previous transformations" can only be well defined if the order of evaluation can be specified. Therefore, for all rules an evaluation order can be specified with the attribute `order`. Furthermore, this allows the creation of a priority of rules: Entities that have been transformed by a rule of earlier order are not touched by rules of later order. Thus it is possible to create rules that apply only to very specific situations and more general rules that apply to a wider range of situations, but do not affect entities for which specific rules exist.

As the language for Map Content Transformations is designed to be similar to the XML format of OpenStreetMap, it has to adhere by XML standards. Therefore, a single root element exists, which is called `<transformation>`. The attributes of that root element are properties that influence the whole transformation. At the moment, the only global attribute is `new_map` which can be either true or false. If it is true, the result of the transformation is stored in an entirely new map that starts blank and will contain only the results of the transformation. Otherwise, the results of the transformation are inserted into the already existing map. This can be used if it is only necessary to add some additional information.

Enclosed in this root element can be an arbitrary number of rules. Each rule describes a situation in the map, i.e. a specific combination of map objects and their attributes, and what constructions should occur based on that situation. Rules are denoted by the element `<rule>`, and the attributes of that element are properties that apply to the entire rule. As it is sometimes necessary to further transform geometry that has been the result of a transformation, the attribute `source` of the `<rule>` specifies whether the identification is to be applied to the original map, or to the result of previous transformations. Due to this possibility of applying rules to the results of previous transformations, it must be possible to specify the order of their application. This is done with the attribute `order` of the `rule`, which can be any integer number. The higher the order is, the later the rule will be evaluated. A list of possible attributes of the `<rule>` element is given in Table A.1 on page 147.

Rule 4.1 shows the XML structure of a full transformation. The transformation target is specified in the root node. The identification sources can be specified for each rule separately, together with the evaluation order.

```
<?xml version='1.0' encoding='UTF-8'?>
<transformation new_map="true">
  <rule order="10" source="original">
    <identify>
      ...
    </identify>
    <identify>
      ...
    </identify>
    <construct>
      ...
    </construct>
  </rule>
  <rule order="20" source="transformed">
    ...
  </rule>
</transformation>
```

Rule 4.1: Structure of a Full Transformation. In this example the full transformation consists of two rules. The first rule contains two identification rules. The second rule is executed after the first, due to the higher order, and works only on results of the first rule, because `"transformed"` is specified as `source`.

The identification and construction rules will be explained in sections 4.5 and 4.6 in detail.

Each `<rule>` consists of an arbitrary number of `<identify>` and `<construct>` elements. The `<identify>` elements describe the situation on the map, to which the transformation is to be applied. This is explained in detail in Section 4.5. Each combination of map objects that fulfills the specifications of the identification is called an instance of the identification. The `<construct>` elements describe, how and what is to be constructed based on what was identified. The construction is applied to each instance of the identification of the specific rule. A detailed explanation will be given in Section 4.6.

4.5 Identification

As has been explained, the identification does not identify objects in an area, instead it is meant to identify objects anywhere on the map that occur in a specified relation to each other. Objects in OpenStreetMap share an n-to-n relationship to each other, meaning that a way can contain several nodes, but each node can also be contained by several ways; and a relation

can contain several other objects, but each of these objects can also be contained by any number of relations. This is contrary to the normal use of XML, where an element can contain several elements, but is only contained by exactly one element. The OpenStreetMap XML format circumvents this by giving a unique id to each object and describing each object on the same level in the XML tree, while all references to contained objects are represented with an element containing only the type and the id of the object. Map Content Transformations use a similar system with some added flexibility.

The XML elements `<node>`, `<way>` and `<relation>` can be used to describe the objects that are being identified, and they can be nested according to the desired relationship. Any object can be given an identifier on-the-fly, by assigning a string that can be freely chosen to the attribute `id`. In any later place where an object of the same type is referenced, that string assigned to the attribute `is` will ensure that the element represents exactly the same object. As in OpenStreetMap, tags are denoted by the element `<tag>`, which has two attributes `k` and `v`, denoting the key-value pairs in the OpenStreetMap data. The value specified by `v` can be a wildcard, denoted by `*`. If a wildcard is used, all tags that have the given key will be regarded as matches, regardless of their value. If tags with a wildcard key are given the attribute `id`, the id string will refer to the value in the data that was matched to the wildcard. Using the id string in curly braces will be replaced with the matched value at any later point. If an element is a relation, its child elements can be given the `role` in order to signal the role that the object that is to be matched should have. If no role is given, or the wildcard (`"*"`) is used, an object of any role can match.

All XML elements can be enclosed in the boolean operators `<and>`, `<or>` and `<not>`. In the standard notation, i.e. if no boolean operator is given, `<and>` is presumed, so that a match for all elements needs to be found in the data. Giving an identifier via the `id` attribute is only possible where no `<or>` or `<not>` has been used on any higher level, as otherwise it becomes unclear which object will really be represented by an identifier.

At this point, it is necessary to explain the evaluation of these XML rules. The rules are matched by stepping down the Document Object Model (DOM) tree of the rules and simultaneously stepping down the hierarchy of the OpenStreetMap data. The XML element directly enclosed by the first `<identify>` will match on any object of its type inside the inspected map area, as there are no restrictions in this first step. After that, the evaluation recursively steps down the DOM tree of the rules and the hierarchy of the OpenStreetMap data. If no matching object (or objects) are found, the parent object is considered unmatched. Assuming that no identifier is involved, if a match is found for all child objects, a match to

the identification rule was found in the map. When one of the elements has an `id` tag, the matching becomes a little more complicated: If this element is matched not by one, but by n objects on the map, the parent object has to be considered matched n times as well.

As an example, the listing in Rule 4.2a defines a very simple identification that is designed to identify traffic signals on streets. The figure in Rule 4.2b shows a street containing two traffic lights. If the identification is applied to the region shown in the image, the result will be two instances of the identification, containing different traffic lights.

If the comment signs are removed, i.e. lines 8–11 are added to the identification, both possible resulting instances are just permutations of each other, and therefore it depends on the attribute `permutations` of the rule whether the result will contain both or just one of those instances (see Table A.1 on page 147). If permutations are not allowed, the selection of one of the two alternatives is coincidental.

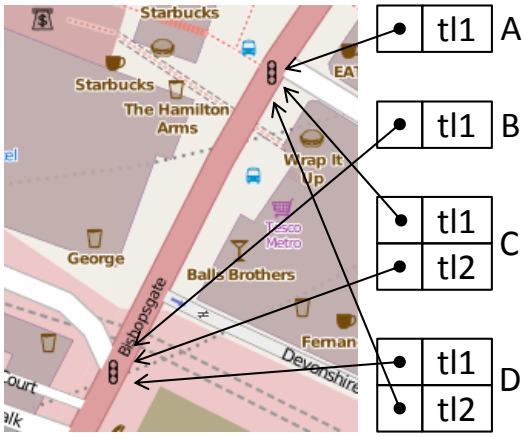
A rule can contain more than one `<identify>` element. Because of the structure of the results, this means that the number of identified results will grow rapidly if the identification elements are not related to each other in some way, as every object matched with one element has to be combined with every object matched with the other element on the whole map. This leads to a complexity of $O(n^i)$, with n being the number of objects on the map, and i begin the number of `<identify>` elements. As this complexity applies to the space of the result, there can be no algorithm with a lower time complexity.

It is therefore advisable—and in most cases desirable—to have some connection between the objects in the different `<identify>` elements. The simplest connection is of course the identity, i.e. stating that an object should be the same as one of the objects that has already been identified previously. This is achieved by the attribute `is`, which implies that the object should be the same as the object that was previously given an identifier with the `id` attribute. Additionally, it is possible to use the `near` attribute. This attribute signals that only objects that are near a previously identified object should be matched. In this case, the `distance` attribute indicates the maximum distance of the object that should be identified to the already identified object. As relations have no physical dimension, the `near` attribute cannot be used with `<relation>` elements.

The `order` attribute of each rule does not only determine the sequence in which the rules are executed. By default, objects that have been identified with an identifier by rules of lower order are not matched on rules of higher order. This arrangement makes it possible to write specialized rules for very specific combinations of objects, and later on more general rules for objects that have not yet been transformed. It is even possible

```
<identify>
 <way>
  <tag k="highway" v="*"/>
  <node id="tl1">
   <tag k="highway"
    v="traffic_signals"/>
  </node>
<!-- <node id="tl2">
       <tag k="highway"
        v="traffic_signals"/>
      </node> -->
 </way>
</identify>
```

(a) A simple identification for nodes on streets that are tagged as traffic lights.



(b) A street with two traffic lights, and different instances resulting from applying variations of (a).

Rule 4.2: Identification of Two Traffic Lights on a Street. (b) shows the result of the identification in (a). A and B denote the instances identified by the listing in (a). C and D denote the instances identified, if the comment marks in the listing are removed. If the rule does not allow permutations, the result will consist of only either C or D.

to add a rule of highest order that leaves everything as it is, so that all objects on the map that have not yet been handled by a previous rule are transferred unchanged to the new map. However, it is sometimes desirable to allow exceptions to this arrangement. Therefore each element that denotes an object (`<node>`, `<way>` and `<relation>`) can have the attribute `ignore_order`. If `ignore_order` is `"true"`, all objects will be matched, regardless of their previous identification. If `ignore_order` is an integer, objects will match if they were identified by rules of lower or equal order than the given integer, but not if they were identified by rules of higher order.

4.6 Construction

After the identification is finished, new geometry can be constructed. The `<construct>` element is used to signal this construction. Each rule can have many `<construct>` elements, and each `<construct>` element can have many elements denoting the construction of new map data. Each construction rule is applied to each instance that was created by the identification.

In the requirements of Map Content Transformations (Section 4.2), it was established that it should be possible to create completely new geometry, which is based on the objects that were identified. On the other hand, there should be more simple rules, which allow easy specification of transformations that are used frequently.

The following pragraphs describe the rules that can be used for construction in detail.

4.6.1 Basic Construction Rules

The most simple construction rules are those that only concern tags and do not construct new geometry. As has been explained in Section 4.3, a new scripting language is not strictly necessary for simple constructions, as they could also be achieved with XML transformations using XSLT. However, in order to stay consistent with the rest of the language, those basic constructions can be realized with the elements `<node>`, `<way>` and `<relation>` in conjunction with the `is` attribute for identified objects. The attribute `as` of these elements, which can be either `"copy"` or `"reference"` determine the creation of the object: If the object is used as a copy, a new object is created for each use. That can mean that the same object might be created several times, if it is identified in several instances of the identification. If the object is used as a reference, no copy is being made, if the `new_map` attribute of the `<transformation>` is false. If the object is used as a reference and the `new_map` attribute is true, one copy will be created in the new map, but later references to the same object will reuse this newly created copy and not create another copy. The `reference` attribute is also inherited, so that all the nodes of a way that is being referenced are implicitly also referenced, even if they are not explicitly mentioned in the rule at all. Tags can be added to all of these objects, whether referenced or copied, using the element `<tag>`, just as in the OpenStreetMap data format. In the values of the `<tag>` elements, values of tags that were identified during the identification can be inserted by referring to the id string inside curly braces. The braces and their content will be replaced by the value that the id string represents. This can be used to constructively create specific strings for descriptions depending on strings that have been identified in the existing map. For example if two street names, Bishopsgate (id “sname1”) and Liverpool Street (id “sname2”) had been identified, and the desired result would be a verbose description of the intersection of those streets, the string for the intersection would be “Intersection of {sname1} and {sname2}”, resulting in “Intersection of Bishopsgate and Liverpool Street”.

Using the element <tags> in conjunction with the attribute is and an identifier will copy the tags of the referenced objects to the object where that tag is used.

It is also possible to create new objects, without directly referring to an identified object. This is done by referring to child objects, which means that only ways and relations can be created in this way, as nodes have no child objects. The creation is done by using the appropriate element, i.e. <way> or <node>, and then using the appropriate child elements, either referencing or copying objects that have been identified. The <way> element can have the attribute polygon. If its value is "closed", the first node of the way is automatically inserted as its last one, so that a polygon is created. Any sub-element of a <relation> can be given an attribute role, which again specifies the role that object is supposed to have in the relation.

4.6.2 For Loops

As both relations and ways contain child objects, it can be useful to be able to iterate over those child objects. This is done with the element <for>. Table A.2 on page 148 shows the attributes of the <for> element and the options and effects those attributes have. The identifiers that are being made available by the for loop (i.e. the value of the id attribute) are only usable inside the <for> element and its sub-elements. Standard scoping rules apply to the identifiers: If loops are nested and the same names are used, inner elements overwrite the visibility of identifiers of higher levels, but not their values—as soon as the inner for loop is left, the old value becomes usable again.

4.6.3 Geometric Construction Rules

The possibility to construct entirely new geometry, based upon specific geometric relation to the already existing geometry, was specified as one of the main requirements of Map Content Transformations. This is difficult to achieve using pure XML and would result in unnecessary verbosity. Therefore, code for geometric constructions is embedded as CDATA sections into the XML code. The language used in these sections is called Geometric Transformation Language (GTL). This language is very simple and based on only two geometric entities: points and segments, which are pairs of two points denoting a line. The general idea is that ASCII characters represent specific geometric constructions based on the geometric form of the character, e.g. an X represents an intersection. All geometric constructions create a segment as output. This might seem counterintuitive, as the result of an intersection is a single point and not a segment. Therefore, in some

cases where a point could be expected, the first point of the segment is by convention one of the points of the input, and the second represents the expected result. There is a reason for this convention: Some constructions need to create new segments, e.g. the creation of a parallel. At the same time the language accepts a segment in any position where a point is expected, using the second point of the given segment. Therefore, all geometric constructions can be considered to create points or segments, as is appropriate for the situation they are used in, and there is no need to remember the result type of each construction.

Some of the geometric constructions require real numbers as a parameter. These numbers are usually also computed from the existing geometry. In contrast to the infix character notation of the geometric constructions, these functions that return real values are realized as classic function calls with a parameter list in parentheses. Additionally real values can be supplied as hardcoded values, by variables, and correctly parenthesized basic arithmetic operations combining all those.

Listing A.1 on page 152 reproduces the Extended Backus-Naur Form (EBNF) of the GTL grammar. Table A.7 on page 153 explains what the operators that are used in the grammar actually do.

The grammar and evaluation rules might seem a little confusing at first, but because of their structure it is relatively easy to chain different expressions. For example, if there is one known segment $[a, b]$, and one wants to create a segment that starts at a, with an angle of $20°$ to $[a, b]$ and twice the length of $[a, b]$, one would simply write $([a, b]V20) * (DISTANCE([a, b]) * 2)$.

Naturally, there needs to be an exchange of data between the XML rule and the CDATA section. Therefore, any identifier that identifies a node can be used in the CDATA sections, where it represents a point. This is also true for nodes that are being identified by the iteration object of a for loop, including those defined via an offset. This is often used when GTL sections are placed inside loops and reference the sections of ways. Additionally, identifiers that identify values of tags that store real numbers can be used just as a replacement for real variables in GTL. Similarly, any points or real values that are defined in the GTL sections can be used in XML construction rules. Regarding visibility, the points and values defined in the GTL section act similar to those defined by a for loop, with the exception that their visibility is defined one element above the `<gtl>` element, so that they can be used by elements of the same or deeper levels of the DOM tree. The reason is that the `<gtl>` elements act like variable declarations for those values and points, whereas the identifiers for the iteration objects of the for loop do not make sense outside of the for loop itself.

4.6.4 Exceptions

In the description of the effects of GTL operators (Table A.7 on page 153) it is mentioned that some geometric constructions can throw exceptions. It is possible to catch these exceptions via the `<try>` element. This element can be inserted at any level of the DOM tree. Once an exception is thrown, the execution unwinds to this level and continues from there. This can be used to create objects only in situations where it is applicable, e.g. by constructing the intersection between two segments, and then only use this intersection in a way, if it exists.

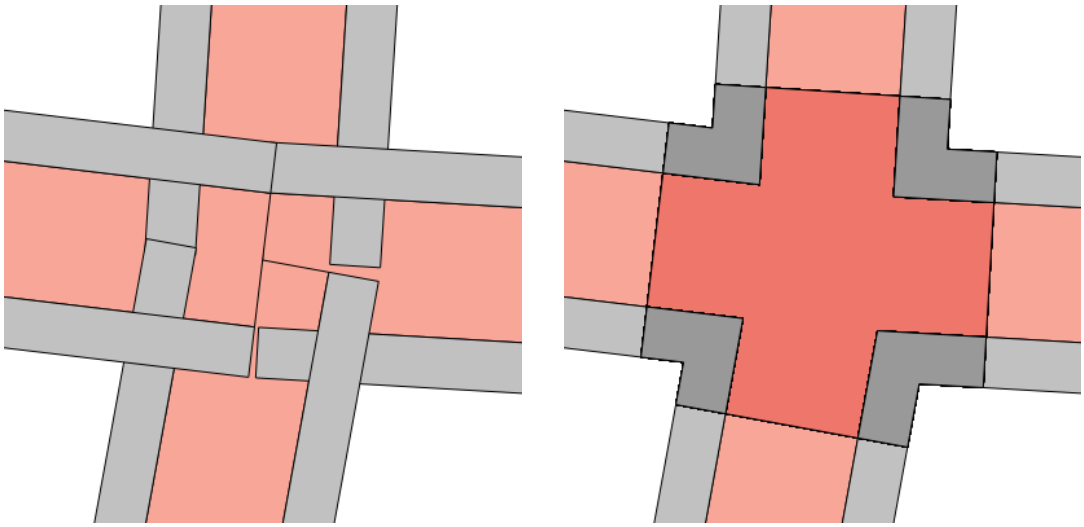
4.6.5 Shortcut Construction Rules

Creating geometries by using geometric construction rules can be rather cumbersome. During development and testing of the construction rules, it became apparent that some specific types of constructions are used frequently, mostly for creating representations of sidewalks, roads and intersections, either as two-dimensional polygons or at a certain distance from the original way. These transformations often refer to an entire way and would require iterating over the way and applying the GTL transformation in every iteration step, or are otherwise rather complicated to create with GTL. For the sake of ease of reading and writing it is therefore useful to provide shortcuts for these frequently used transformations. They are realized as XML elements and controlled by their attributes.

The attributes can either be specific strings that control certain properties of the rule, id strings of identified objects, or real values. Real values in this context can be hardcoded values, or id strings of tags whose values can be interpreted as real numbers, or any correctly parenthesized arithmetic expression containing either or both of those.

The first shortcut construction rule is invoked with the element `<hull_polygon>`. It is used to create a polygonal representation of a way or a node, and can be used to create polygonal representations of streets, sidewalks, pedestrian crossings and similar objects. Table A.3 on page 149 lists all available options for this element; Figure 4.3a gives a visual representation of the construction of such polygons.

As the polygonal representations of streets are constructed independently of one another, intersections are problematic areas. Figure 4.2a shows how constructing polygons for streets and sidewalks results in unwanted gaps and overlapping near intersections. Therefore the `<intersection_polygon>` element can be used to create representations of streets or sidewalks at intersections. Those polygons are meant to represent the intersection in a way that continues the street and sidewalk polygons.



(a) Situation at an intersection where street and sidewalk polygons meet. Gaps and outlines of the polygons are visible even where they should not be.

(b) Intersection polygons are rendered on top of the streets and sidewalks, obscuring the visible outlines. The intersection polygons of the sidewalk are not connected.

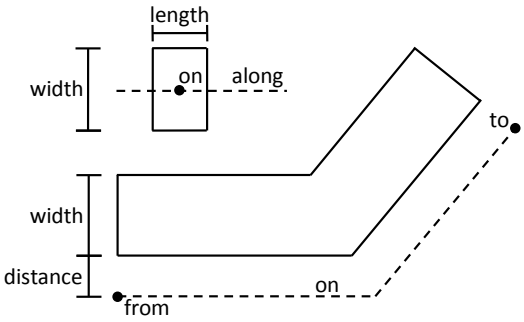
Figure 4.2: Necessity of Intersection Polygons. An intersection of polygonal streets and sidewalks without (a) and with (b) intersection polygons.

For this reason the street representation itself is a single polygon, whereas several unconnected polygons are needed to represent the sidewalks. Figure 4.2b shows the same intersection with intersection polygons. In order to ensure such a result, the intersection polygons need to be rendered on top of the polygons representing the ways, e.g. by constructing tags that represent a z-value. The available options for the `<intersection_polygon>` are listed in Table A.4 on page 150, a visual representation can be found in Figure 4.3b.

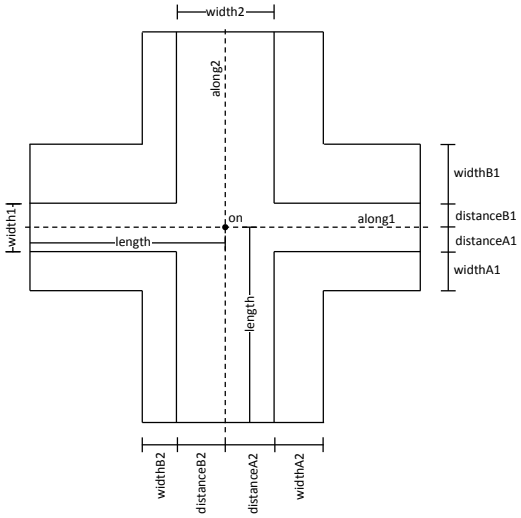
With GTL, parallels to existing lines can be created relatively easy. However, as ways are polylines in OpenStreetMap, creating simple parallels to the sections of the ways does not suffice, as those would create gaps at nodes. Creating a connected parallel, e.g. for a route graph representation of a sidewalk, can be achieved with GTL, but the element `<parallel>` can be used as a shortcut. Table A.5 on page 151 explains the possible attributes of this element.

4.6.6 Topological Construction Rules

The results of the transformations described in the previous sections are new geometries. However, the topology of those new geometries have



(a) Two hull polygons, one for a way (with a distance), and one for node. If the distance is not given, the hull polygon would be created directly around the original way.



(b) Intersection Polygons of empty subtype and subtype distance.

Figure 4.3: Attributes of Shortcut Constructions. A visual representation of the effects of the attributes of shortcut explanations for `<hull_polygon>` (a) and `<intersection_polygon>` (b). Extensive explanations can be found in Tables A.3, and A.4.

been ignored until now, even though the topology is in many cases just as important as the geometry. As an example, if a route graph is created, all the different parts need to be connected at the correct places, otherwise it will just be a collection of line segments without much actual use. Some of the topology can be created correctly by using appropriate geometric transformations. However, geometries created by different rules can never be connected directly after their creation as each rule operates completely independent of other rules. Additionally, each rule operates independently on each instance of the identification. Thus, the newly created geometries need to be connected by separate rules that are applied to them after their creation.

The simplest rule is the element `<connect>`, with the attributes `way1` and `way2`, as well as the optional attributes `from1`, `to1`, `from2`, and `to2`. If an intersection between the ways exist, a node is placed at its position and added to the ways, thus creating a topological connection between them. If the optional attributes are given, only the segments between the given nodes are considered. The values of the optional attributes adhere to the same standard as the `from` and `to` attributes of the `<for>` loop (see Table A.2), meaning that identified objects, the strings `"first"` and `"last"` and increments and decrements can be used.

The element `<crop>` allows to crop protruding parts of a way that are not

connected to anything specific and often occur after connecting different parts. Every node at the beginning and the end of the way is cropped until the first node that is part of another way is encountered. The attribute `way` identifies the way that should be cropped. The optional `max_length` attribute defines a maximum cropping length. Protruding parts that are longer than this length are not cropped.

Apart from connecting ways, disconnecting them is also necessary. For this, the element `<disconnect>` is used. The attribute `way` determines the way that is to be split, while `at` determines the node at which it is split. The way is replaced by two ways that end (or start, respectively) at this node, which is still shared by both ways.

Additionally, a shortcut for a topological pattern that is often needed at intersections is provided by the element `<connect_sidewalks>`. The attributes for this element need a more in-depth explanation. The attributes assume that when the sidewalks were created (in an earlier rule), relations between a road and its sidewalks were created, as it would otherwise be difficult to relate a road to its sidewalks in later identifications. Therefore, the element is given a relation and a role for each sidewalk and then uses those to extract the sidewalk itself. Table A.6 on page 151 explains the different attributes. After the connection, the protruding parts of the sidewalk are usually cropped with the `<crop>` element.

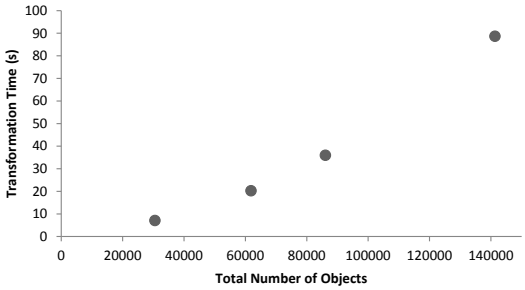
4.7 Performance

The performance of applying transformations to a map cannot be determined as a universally valid term, because it depends on how the transformation is written, similar to how the formulation of an SQL query determines its performance.

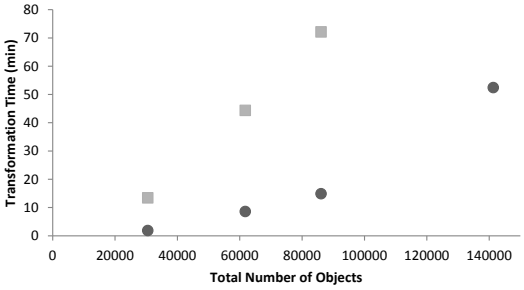
The identification parts of the rules have the most significant impact on performance. The construction is always in $O(n)$, with n being the number of identified objects, which is negligible compared to the identification.

As explained in Section 4.5, the complexity of the identification depends on whether a relation between the objects is coded into the rule. This relation can either be a direct relation, i.e. a parent-child relation, or a geographic correlation, a nearness. While the parent-child relation is represented directly in the data, the performance gain of the nearness is achieved by an R-tree, as first presented by Guttman [34].

To give a more realistic impression, three transformations were performed on several maps of a city area, with between approximately 30000 and 141000 map objects (nodes, ways and relations). The largest of those maps encompasses approximately thirty square kilometer of the city of



(a) Transformation time with structurally connected objects in identification.



(b) Transformation time with unconnected objects in identification. Light gray dots show results without R-Tree, dark gray with R-Tree.

Figure 4.4: Transformation Performance. Transformation time on maps with 30k, 61k, 86k and 141k objects.

Stuttgart, a densely populated area with many map objects. The other maps were correspondingly smaller parts of the same area. The first transformation created two-dimensional representations of streets, specifically named and two-dimensional representations of intersections and pedestrian traffic lights and was also used in the evaluation in Section 8.2 as T2. The identification rules of this transformation use only combinations of map objects that are connected by the data structure, thus minimizing the number of possible combinations. Figure 4.4a shows the time that was needed to perform the transformations on a desktop pc with a Core2 with 2.39GHz and 4GB of RAM.

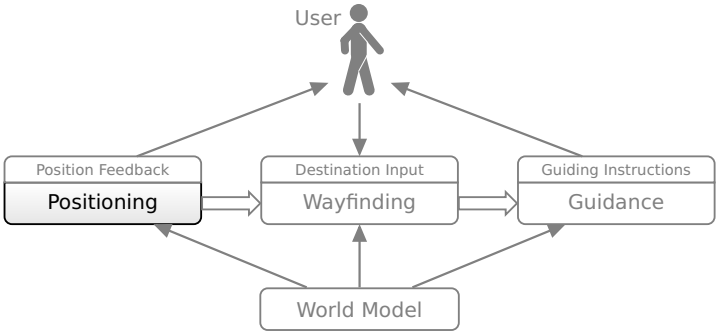
The second and third transformation are variations of the same transformation and contain identification of object combinations that do not have any connection in the data structure itself. The result is the route graph described in Section 9.2. One variation uses a naive writing of the rule, the other variation heavily uses the `near` attribute and thus the R-tree to reduce identification time. Figure 4.4b shows the transformation time for both variations.

4.8 Discussion

The first goal of the development of Map Content Transformations was to prove that it is possible to create a system that can produce adapted maps based on rules that encode the requirements and preferences of a specific user group, while not restricting this adaptation purely to graphical appearance. This goal was achieved, as the fully functional implementation of the system proves. Later chapters will explain in detail, how all the

stages of navigation can profit from the adapted maps that can be created with the system.

The performance of applying the transformations, as represented by Figure 4.4a, currently makes it unfeasible to apply complex transformations to large areas in an amount of time that a user can reasonably be expected to wait. For smaller areas, such as used for the testing of the prototype, this does not pose a problem. For real-world deployment, a possible solution could be to cache the result of common transformations on the server side.



5

Determination of the Position

In every navigation system, determining the current position of the user is a vital step. Car navigation systems rely on GPS in combination with dead reckoning for places without satellite connection, such as tunnels. Additionally, some knowledge from the map is used (e.g. it is likely that a car drives on a road), and sometimes other assumptions (such as that the driver follows the directions of the system) play a role as well.

However, simply adapting these mechanisms to pedestrian navigation systems is not a very good solution, as the requirements differ between these two types of navigation systems.

The main differences regarding the requirement are accuracy and availability: The accuracy needs to be higher than for a car navigation system, as it does make a difference whether the user is positioned on a sidewalk or in the middle of the street, whereas for car navigation systems, street level accuracy is good enough. Navigation systems for blind people have especially high requirements regarding accuracy, as the ability of the user to mentally correct small errors by visual comparison of the reality and the representation given by the system is severely limited. Positioning for pedestrians should ideally be available both in- and outdoors for longer periods of time, and additional infrastructure for indoor should not be necessary as a world-wide roll-out of such an infrastructure is not to be expected.

It should be noted that those requirements are not specific for navigation

---

Parts of this chapter have previously been published in:

Bernhard Schmitz, Attila Györkös, and Thomas Ertl. Combination of map-supported particle filters with activity recognition for blind navigation. In *Proceedings of the 13th International Conference on Computers Helping People with Special Needs (ICCHP 2012)*, 2012. DOI:10.1007/978-3-642-31534-3_78

systems for disabled people—accuracy and availability of positioning is sought in many applications that require pedestrian positioning. However, while not being specific, the requirements are still of higher importance than in many other applications: Where high accuracy may be of benefit in other applications, it is a requirement when guiding blind users.

For these reasons, the problem of exact in- and outdoor positioning is an area of research in and of itself, and in its many aspects beyond the scope of this thesis. This thesis therefore concentrates on one aspect of position determination, namely particle filters. The reason for this decision is that particle filters can be used to significantly improve indoor positioning (see [118]), if accurate maps of the indoor environment are available, such as those mentioned in Section 3.3.1.1 on page 34. Additionally, particle filters can be used in conjunction with Map Content Transformations to achieve some improvement in position determination in outdoor areas, as will be shown in Chapter 6. Thus, particle filters are a good example of the versatility that is required in a system that aims to be available in a wide range of situations and places.

5.1 Background

Particle filters are a subset of Monte Carlo algorithms, which have first been used in positioning systems for robot navigation by [29] under the name of Monte Carlo Localization. Later, they have also been successfully employed for pedestrian navigation, mostly with a foot-mounted inertial sensor by [9], [59], and [118].

In the context of map-based positioning, each particle represents a possible position, annotated with a certain probability, depending on the measurement that has been carried out and the uncertainties present in that measurement. In the simplest case, the probabilities are evenly distributed across all particles, except for those that have passed through walls and are therefore set to zero. The concrete position is then computed from these particles.

The particle filter that was implemented for the system in this thesis operates on the data by the Xsens Mtx measurement unit worn on a strap around the neck (see Section 2.4). Compared to attaching it to a foot, this has the disadvantage of not providing zero crossings in forward acceleration. However, it is easier to wear, the sensor data is not influenced by additional foot movements and (excepting pitch) the device coordinate system corresponds to the body coordinate system.

Activity recognition recognizes several movement types of the user, e.g. ascending a stair. It can thus be used to enhance the particle filter, as

these movement types can be correlated to a position on the map. Activity recognition with inertial sensors has been a widely researched topic in recent years. Many systems have been built that use a single sensor [56, 73], or several sensors placed on different parts of the body [7, 102]. The former show that a single sensor can be used to detect simple activities like walking or ascending stairs, and an activity detection can therefore be used in our navigation system, which is fitted with a single sensor.

At first, a step detection, step width estimation and activity recognition is performed on the accelerometer data. The results of the step width estimation together with the direction given by the sensor's combined gyroscope and magnetic field sensor is used as an input to the particle filter.

5.1.1 Step Detection and Width Estimation

A lowpass filter is applied to the accelerometer data, and a list of zero crossings and extrema is generated from the mean adjusted resulting data. Steps are then detected by applying a deterministic finite automaton accepting the regular expression $zc\ min * \ zc * \ max * \ zc$ to this list, where zc stands for a zero crossing and min and max for minimum and maximum respectively.

According to the method developed by Kourogi and Kurata [58], the walking velocity v is estimated for each step based on the simple formula

$$v = (max - min) * s_{factor} + s_{offset}$$

where max and min are the maximum and minimum values for the observed step. The parameters s_{factor} and s_{offset} are different for each person and are negative, if the activity *walking backwards* is recognized (see Section 5.1.2). As the sampling rate of the sensor data is given, the duration of one step is also known and therefore the step width can be calculated.

5.1.2 Activity Recognition

To recognize activities, a plain decision tree algorithm was implemented. The algorithm currently detects four different activity types: *walking*, *walking backwards*, *ascending stairs* and *descending stairs*. In order to train the decision tree, a total of 1.5h of acceleration data was collected.

Staircases in buildings do not consist of continuous steps. Normally, there exists a level area on each floor (and often in between floors). Therefore, the data for ascending and descending stairs had to be manually edited to remove the steps on those areas as well as the first and the last

Table 5.1: Recognition Rates of Activities. Rows indicate activities as they were recorded, columns show the recognized activities.

	Recognized as (in % of Cases)			
Activity	Walking	Backwards	Ascending	Descending
Walking	**94.69**	0.21	5.30	0.00
Backwards	18.52	**81.48**	0.00	0.00
Ascending	0.00	0.00	**90.91**	9.09
Descending	0.00	0.00	9.52	**90.48**

step of each staircase, which have distinct acceleration patterns. Because of this, only about 45% of the data originally collected could be used to train the decision tree for ascending and descending stairs. Furthermore, stairs with different slope angles must be taken into consideration.

As feature points, the mean value and the standard deviation are extracted for each axis from the data in addition to the minimum and maximum values of the vertical acceleration. With this data, the decision tree was trained, using a CART (Classification and Regression Tree) algorithm [20]. Ten-fold cross validation was used to evaluate the decision tree, i.e. the data is divided into ten disjoint subsets, nine of which are used for training, while the tenth is used as a test data set. This is repeated ten times, so that each data set is used as a test set once. The recognition rates of the resulting decision tree obtained by computing the mean rates of all ten test data runs can be found in Table 5.1.

5.2 Particle Filter

Particle Filters are an intuitive probabilistic approach to handling inertial positioning: Instead of tracing a single location, a number of possible locations, each with an associated probability, are being traced. A possible position together with its probability is called a particle. All particles represent a subset of all possible locations and are used to compute the most probable location. The details of how particles are distributed and probabilities are assigned can vary. For particle distribution, two different models were implemented, a direct and an indirect one.

In the direct model, two normally distributed random variables are used. One is added to the direction, the other to the step length. This is the more intuitive implementation, as errors made in the detection and computation of direction and step length are independent. The indirect model uses three

normally distributed random variables. One is added to the direction, the other two to the x and y position respectively, after the position has been computed.

In both models, probabilities are directly affected by two factors: Walls and activity type. It is not possible for a person to move through a wall, and the maps are considered accurate. Therefore all particles that passed through a wall are assigned a probability of 0, i.e. there is no uncertainty involved regarding collisions with walls.

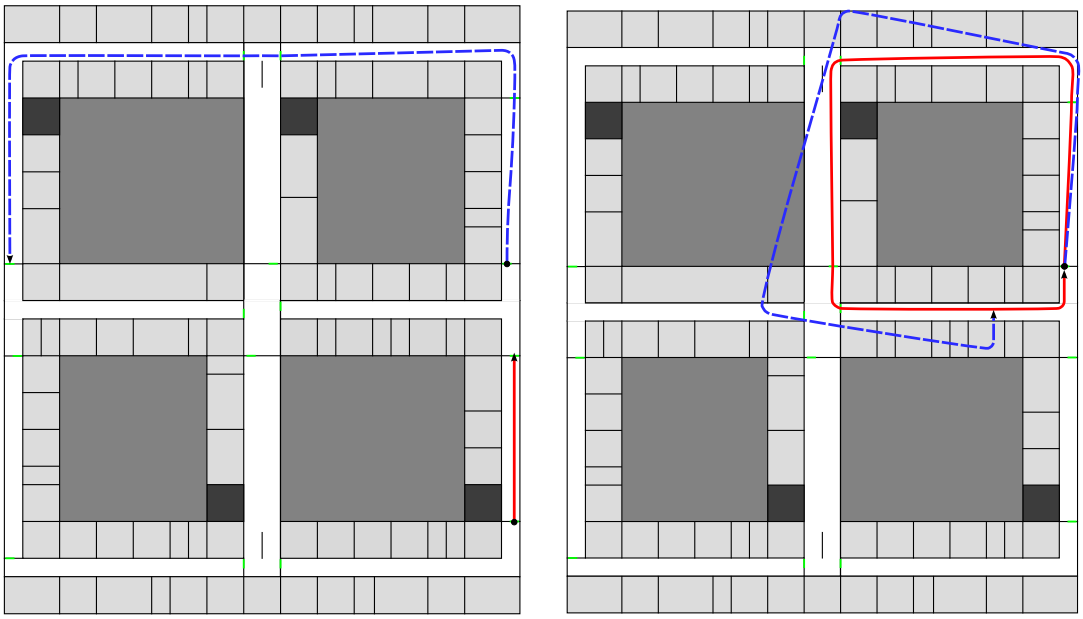
If no collisions are found, the activity type is taken into account. The activity types that are used for the correction are *walking*, *ascending stairs* and *descending stairs*. The recognized activity is compared with the expected activity based on the particle position. If they match, the probability for this particle should theoretically be 1.0, otherwise 0.0. However, as shown in Table 5.1 the activity recognition does not work perfectly. The recognition rates are known for each possible combination of activity and detection, and these rates can be used to determine the uncertainty for the measurement. Before being used to compute the approximated position of the user, the probabilities of the particles have to be normalized.

The use of elevators is currently not a supported activity. This decision was based on two reasons: Previous work differs widely in the reported recognition rates for elevator usage: While Kourogi and Kurata [58] report perfect recognition rates for elevator detection, Bao and Intille [7] report that elevator detection does not work very well. It can be speculated that the type of the elevator plays a role, or that the diversity of activities recognized by Bao and Intille [7] has a negative influence on recognition rates. The decisive reason is that even if an elevator usage would be recognized, it would not give the number of the floor on which the elevator stops.

5.3 Results

Data from three different routes were collected in order to test the developed algorithm (Figure 5.1). One route was collected twice, once with a sensor that had a drift of up to 40° (Route 3b).

All routes have been evaluated with the direct and indirect model, using normally distributed random variables with a standard deviation ranging from 5° to 30° (in 5° steps) for directional variables and from 0.2m to 1.0m (in 0.2m steps) for position and length variables. The best results have been achieved with a standard deviation of 5° or 10° and 0.2m or 0.4m, respectively.



(a) Routes 1 (red) and 2 (dashed blue)

(b) Routes 3a (red), 3b (dashed blue)

Figure 5.1: Plan of the University Building with Recorded Routes. Route 3b does not show the real route (which is the same as 3a), but the route that would result without collision detection and particle filter.

Table 5.2 shows the accuracy obtained with the direct and the indirect model. For this table, the best results from the experiments with differing random variables have been selected, in order not to discriminate against a model. If no distance is given, the estimated location did not end up anywhere near the real location, due to getting stuck in a room somewhere along the way. As can be seen, the direct model achieves better results throughout, but is not able to cope with the highly drifting sensor, where the indirect model is still able to compute a result.

Figure 5.2 shows the average and maximum runtime of the computation of one step using between 50 and 1000 particles. The performance measurements have been executed on a laptop with an Intel Core i5 2410M processor with 6 GB of RAM. Unsurprisingly, the performance deteriorates with each increase in the number of particles. In contrast to that, the accuracy is not directly related to the number of particles, as can be seen in Table 5.2. Therefore, a direct tradeoff between performance and accuracy does not exist. The maximum runtime can be more than twice as high as the average runtime. High runtimes happen, if a resampling occurs, i.e. new particles have to be generated.

Table 5.2: Accuracy of the Particle Filter. The best achievable accuracy (in meter) selected from a variety of tested standard deviations for the random variables, using both direct and indirect model. The last line, given as a reference, shows the result using only step estimation, without a particle filter. The direct model has better accuracy, the indirect model is able to cope with a drifting sensor (route 2b).

	Deviation from Real Position for Routes (in m)							
	Direct Model				Indirect Model			
Particles	1	2	3a	3b	1	2	3a	3b
50	0.3	1.3	0.5	-	0.5	11.7	0.8	11.7
100	0.1	1.3	0.3	-	0.5	10.8	0.8	9.8
250	0.3	1.2	0.3	-	0.5	3.2	0.7	9.2
500	0.2	1.3	0.3	-	0.6	2.0	0.7	9.1
none	0.2	-	-	-	0.2	-	-	-

5.4 Discussion

The implementation of a particle filter did significantly improve the positioning inside buildings in comparison to a simple step-tracking algorithm. However, the effectiveness of combining activity detection with the particle filter could not be reasonably determined. In the test building, staircases

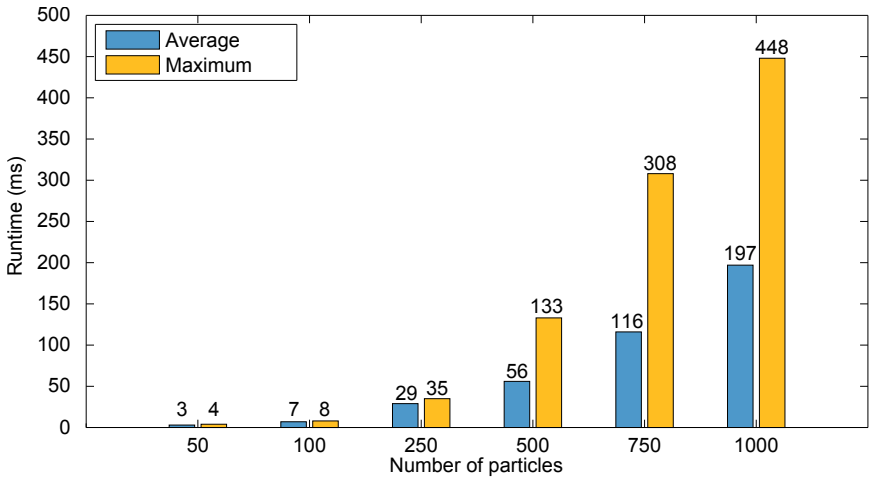
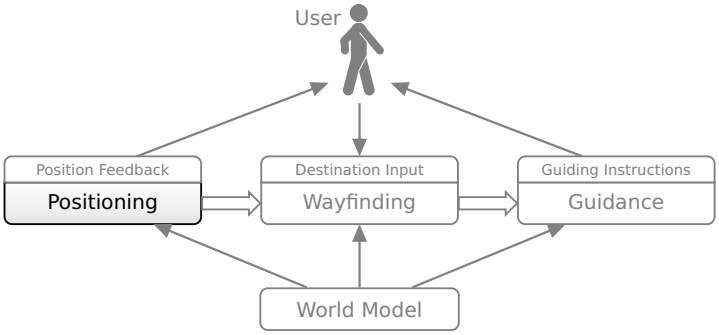


Figure 5.2: Average and Maximum Runtime. The increase in runtime with more than 250 particles cannot be justified by an increase in accuracy (see Table 5.2). The high maximum runtimes occur, when resampling takes place.

are entered through doors, whereby particles not entering the staircase are automatically assigned a probability of 0, as they would pass through a wall. The recognition for walking backwards is used to correctly determine the direction of a step. The results confirm the finding of previous work in this area, and show that it is possible to use particle filters with an amount of particles that is small enough for use in a real-time application. Due to the limitation of adjusting particle probabilities at walls, the approach presented in this chapter is limited to well-mapped areas, mostly indoors. The next chapter discusses a novel approach to applying particle filters in outdoor areas.



6

Map Content Transformations in Positioning

The previous chapter described positioning with particle filters in the context of an indoor environment. Tests were conducted with a highly accurate map, which clearly indicated impenetrable walls.

Nevertheless particle filters are not strictly a technology restricted to indoor use. Even though the many walls and obstacles present in an indoor scenario are advantageous for particle filters, outdoor areas have specific properties that can be used to influence the probabilities of the particles. Furthermore, the influence of these properties on the probabilities depend on the abilities and preferences of the user. The most obvious example is possibly that the probability of a wheelchair user being on a staircase approximates zero. Other examples, such as a person being unwilling to walk on a non-paved surface, such as grass, can be easily imagined.

At first glance, the dependence of positioning on the user might be rather surprising—and indeed a reliable, precise and widely available positioning system would render any user-specific algorithm for positioning obsolete. However, for the time being, such an algorithm seems like a sensible approach, but requires the possibility for individualization. As Map Content Transformations are specifically aimed at individualization, they can be used as a background tool for this individualized positioning with particle filters.

Using Map Content Transformations as a preprocessing step for a particle filter can serve two distinct purposes. The first is to create explicit representations of implicit OpenStreetMap information. A primary example are tags of nodes and ways, containing specific information, for example about the surface of a way or about the composition of its shoulder. This information is potentially useful for the particle filter, but not necessarily

usable in its original form: For impenetrable objects like walls a polyline representation suffices, because a particle that crosses it can be said to have a probability of zero. If, however, the probability is to reflect some user preference, e.g. a user who will likely not walk on grass, but still might do so in some circumstances, a two dimensional representation is needed.

The second purpose of Map Content Transformations in the context of particle filters is the individualization of the map and the probabilities that influence the particle filter. While the transformations for the first purpose could theoretically be the same for all users—even if some of them might be unnecessary depending on the those of the second purpose—the transformations that are employed for the second purpose need to differ for individual users or user groups. They are used to transform the diverse notation of OpenStreetMap tags into a unified representation of the user's abilities and preferences. In the simplest case this unified representation is merely the probability of the user walking on the specified area.

6.1 Map Content Transformation for Particle Filters

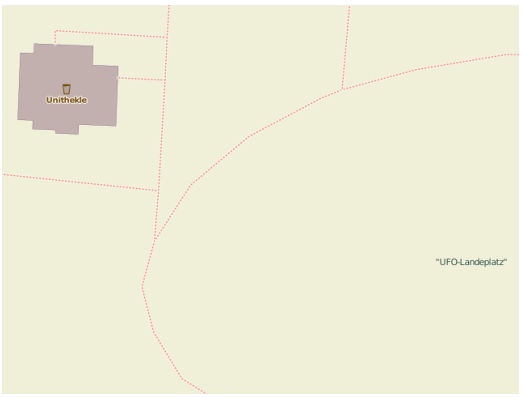
While sections 4.4–4.6.6 explained how Map Content Transformations work, particle filters are the first example of their real practical use. This section therefore gives an example of how concrete Map Content Transformations can be used in this context. This includes the transformation itself and how its result can be used by a particle filter.

Particle filters work best when there are some restrictions to the movement of the user. Consequently, particle filters are of little use in large open areas with a uniform surface and no obstacles. OpenStreetMap data does, with some exceptions like fences, not by itself represent movement restrictions, and it is therefore the primary goal of the Map Content Transformations for particle filters to create these.

When implementing a Map Content Transformations for particle filters, one should keep in mind that in all likelihood a large part of the map will be unaffected by the transformation. There needs to be a default probability for all areas with no set probability, which can also be either 1.0 or 0.0. The latter only makes sense if one knows that the base data and the transformation are very accurate, which is rarely the case. For the other areas, one should set probabilities as well as possible.

The simplest case is adding a probability to an area that is already defined in OpenStreetMap, e.g. a grass area that is explicitly marked as such. Figure 6.1a shows several footpaths passing through a meadow with a nearby lake. The probability for the lake can be set to 0, for the meadow, a value depending on the user should be chosen. With these probabilities set,

(a) OpenStreetMap map of footpaths through a meadow with a nearby lake.



(b) OpenStreetMap map of footpaths through a grass area that is not marked.

Figure 6.1: Screenshots of Map Details. The grass area that is present in both cases is specifically marked as such in one case, but not in the other.

it becomes apparent why the one-dimensional ways need to be converted to two-dimensional polygons for this chapter: In order to model the behavior of the user, who is expected to stay on the footpaths, the footpaths need to become areas of their own, with probabilities higher than that of the meadow.

In other cases, the surface of the area surrounding the footpath is not explicitly defined. It is possible to store a hint about the areas to both sides of the footpath in a tag of the way. Using this simple tagging mechanisms would allow blind users to enter such information where it is not present. Figure 6.1b shows an example of such an area. In this case, the transformation needs to create areas to the side of the footpath containing the probabilities. The footpath itself does not strictly need to be transformed into a polygon, but it is still advisable, especially, if there is a default probability other than 1.0.

Another example for a Map Content Transformations that could be of benefit for a particle filter is the creation of a impenetrable barrier on one side of a sidewalk which is known to be bordered by a row of houses or a fence. Again, this knowledge could be stored in a simple tag of the street.

6.2 Evaluation of Individualized Positioning

In order to evaluate the benefits of map transformations for positioning in a navigation system, a relatively simple experiment was conducted: First the inertial data of three routes were recorded. All routes were recorded in

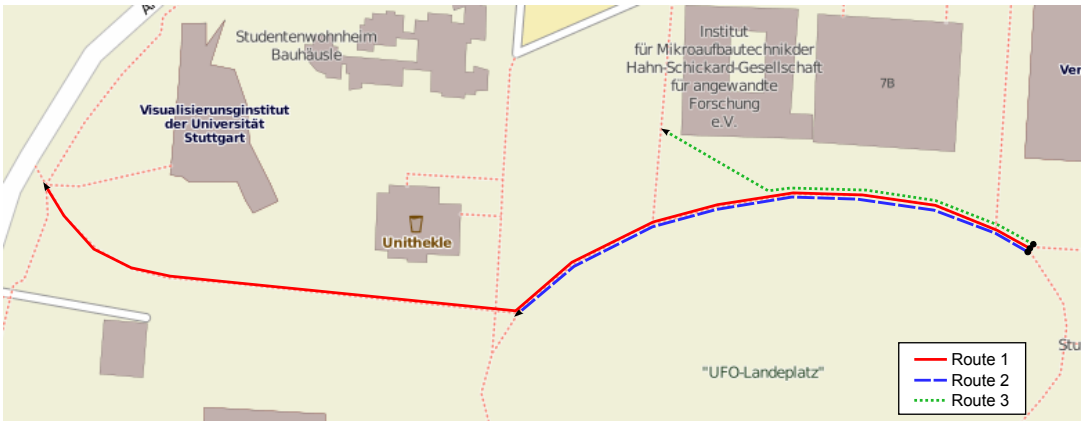


Figure 6.2: Recorded Routes for Individualized Positioning. Route 1 (red) is about 300m long. Route 2 (dashed blue) was intentionally recorded with high sensor drift. Route 3 (dotted green) was recorded with the same sensor as Route 1, and leads through a grass patch.

the same area, where several tarmac footpaths cross a large grass area. Figure 6.2 shows the layout of the different routes. The first route had a length of about 300m. The second route of about 160m was a part of the first route, but was recorded with a sensor with higher gyroscopic drift. The third route, recorded with the same sensor as the first, was intentionally leaving the tarmac walkways and taking a shortcut through the grass. The inertial data recorded on those routes was then used with the algorithm for step length estimation described in Section 5.1.1 in combination with the particle filter as described in Section 5.2. The probability of a particle is adjusted only at the moment it enters an area with a different probability, not during each step it stays inside that area, as doing so would rapidly decrease the probability of particles on the grass with each step.

In order to test both possibilities described in the previous section, two slightly edited versions of the original OpenStreetMap data in combination with two different Map Content Transformations were used: The first version added tags to the way in order to indicate that both shoulders of the path consist of grass. The transformation uses this information to construct polygons representing the way and the shoulders, with probabilities lower than 1.0. The second version added a grass polygon to the OpenStreetMap data, as it exists in many places in OpenStreetMap. The transformation simply adds a probability lower than 1.0 to that polygon. It was intentionally decided to use recorded data on edited versions of the same area, in order to rule out influence by other factors and thus improve the comparability of the two methods.

The recorded inertial data was then used to perform position tracking

(a) Detail of transformed map based on `"shoulder"` tags of the footpath (S).

(b) Detail of transformed map based on a polygonal area (A).

Figure 6.3: Results of Transformations. Both methods were used for testing, the appropriate results can be found in Table 6.1.

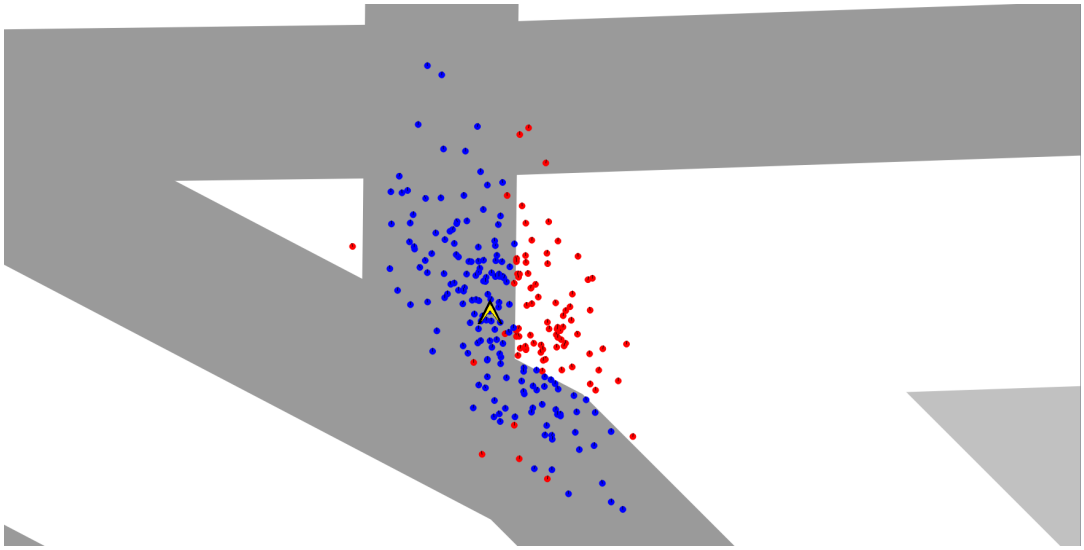


Figure 6.4: Particles on a Map Detail with Grass Shoulders. Particles with higher probability are displayed in blue, those with lower probability in red. For better particle visibility, the grass shoulder is rendered white. Screenshot was taken with 250 particles and shoulder probability set to 0.

of the pedestrian for the three routes with differing probabilities for the shoulders or grass polygon. The computed position was then compared to the real position of the user during the recording of the data. As the real position for both the start and the end point of each route was determined

Table 6.1: Accuracy of Particle Filter with Map Content Transformations. The three routes as shown in Figure 6.2 were tested with both the shoulder (S) and the area (A) transformation (see Figure 6.3). The distance between the computed and the actual position is given in m. The first row uses the position as computed without a particle filter. As expected, the positioning improves with the use of a particle filter, if the user stays on the tarmac footpaths, but deteriorates, if the user leaves those paths.

	Deviation from Real Position (in m)					
	Route 1		Route 2		Route 3	
Probability	S	A	S	A	S	A
-	9	9	14	14	4	4
0.5	5	3	11	11	8	8
0.2	6	7	10	10	7	8
0.01	4	7	7	6	7	8
0.0	3	7	4	4	-	-

by visual comparison of the real world with maps and aerial images, the results cannot considered to be highly accurate, and were thus rounded to the nearest meter. Table 6.1 shows the results of the experiment. For the experiment, the direct model (see Section 5.2) with 250 particles, and standard deviations of 0.6m for step length and 10° for direction was used.

Positioning for Routes 1 and 2 is improved by the particle filter. As expected, the determination of the position gets worse with a particle filter for the route that actually goes across the grass area. In this case, if the probability of walking on grass is set to 0, no reasonable result can be achieved, as the computed position is unable to follow the actual path.

The method of comparing computed and real position only at the destination does have some drawbacks, as it does not consider the difference of position during the walking. In some cases there are larger differences in position during the walking period, which are not reflected in the end position. Position differences that occur during mid-walk can also even out themselves over time, leading to surprising results, such as the good position results for Route 1 with the probability 0.5. In order to give an idea of the effects over time, Figure 6.5 displays the computed routes for Route 2 with and without a particle filter. For reasons of visibility it was decided to display only one route, and Route 2 has the largest difference between the real route and the collected data. For this example, the transformation creating shoulder polygons with a probability of 0.01 was used.

The difference between the transformation that depends on the

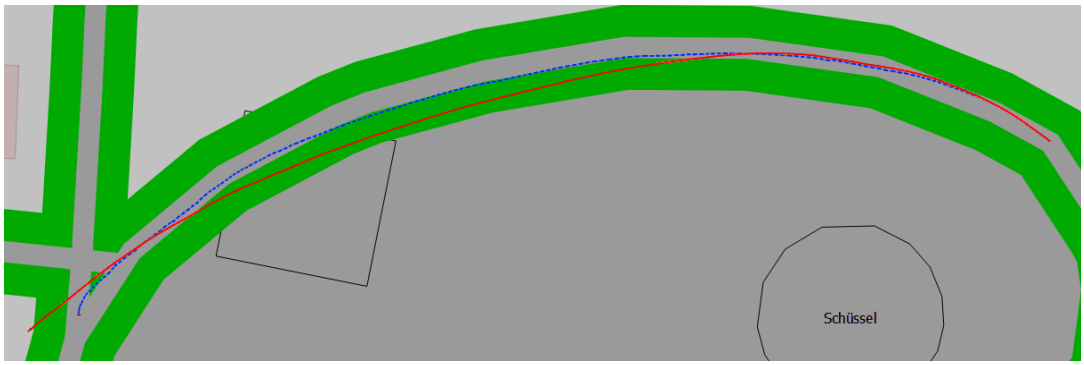


Figure 6.5: Comparison of Positioning with and without Particle Filter.. The complete computed route for Route 2 is displayed here, without particle filter (red) and with particle filter and probability of 0.01 (dashed blue). The real walking position was in the middle of the footpath, but it needs to be considered that the footpath's real course is only approximated with a polyline here.

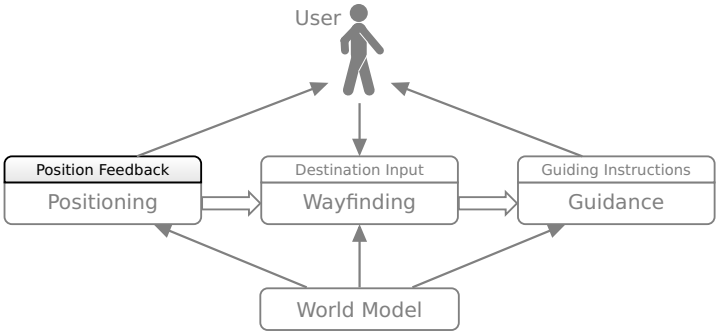
`"shoulder"` element and the one that depends on a mapped grass area, can be explained by the different extent of the area that is marked with a probability in the result of the transformation, as particles can enter this area, if the probability is higher than 0.0. If resampling occurs, particles can be generated on the other side of the shoulder polygon, thus not being influenced by its probability, whereas with the explicitly mapped grass area, particles generated outside of the generated footpath polygon are always influenced by the area's probability.

6.3 Discussion

The Map Content Transformations used in the evaluation are just one example of their application in positioning. As the evaluation has shown, Map Content Transformations can be used to improve positioning in specific circumstances. In contrast to other application areas of Map Content Transformations, the individualization is not necessary to match the abilities or requirements of the user. Instead, the individualization can be used to achieve the best possible result by taking into account the user's preferences. This is in contrast to the indoor particle filter of Chapter 5, where the particle probabilities are given by the physical realities of the building. The results show that knowledge about the user's habits encoded into the transformation can have an effect on the accuracy of positioning: People who are blind often actively avoid walking on grass in order not to get lost, and such users are best served by setting the probability to 0. This

same setting would corrupt positioning for users that do walk on grass in specific circumstances. It could also be shown that the difference between using either areas or shoulders is not large enough to warrant a definite preference for one method. It is thus best to use whatever approach is suggested by the map data at hand.

Nevertheless, this application area for Map Content Transformations can be expected to exist only temporarily: Positioning is a general problem, and as soon as it is solved satisfactorily, e.g. by an advanced GPS-like system, there will be no further need on a positioning system that relies on personal information about the user, and the techniques presented in this chapter will become obsolete.



7

Position Feedback

After the position of the user of a navigation system has been established, it becomes possible to give feedback about the current location. In its simplest form this feedback could simply be a short verbal description, output as text, via a text-to-speech engine or on a Braille display.

But just as the text output of early car navigation systems was gradually replaced with displayed maps, so can the positional feedback of pedestrian navigation systems be enhanced by maps. As visually displayed maps are a good and usable solution for most users, this chapter focuses on blind users. Blind people cannot access maps as easily as sighted people, but the information that is conveyed through a map is useful nonetheless: Information about the immediate environment can help the user to not just navigate to his goal but also create a mental map of the environment, which can be helpful for later navigation. For this reason, this chapter does not only include portable systems, but also deals with stationary systems for map feedback. Even though these systems cannot directly give positional feedback, they can nevertheless be of help for navigation in general, if used in advance of visiting a specific area, or after a visit to review routes and building locations.

Two completely different approaches to conveying maps to blind people are discussed, mainly tied to the hardware side of things. The first approach

---

Parts of this chapter have previously been published in:

Bernhard Schmitz and Thomas Ertl. Making digital maps accessible using vibrations. In *Proceedings of the 12th International Conference on Computers Helping People with Special Needs (ICCHP 2010)*, pages 100–107, 2010. DOI:10.1007/978-3-642-14097-6_18

Bernhard Schmitz and Thomas Ertl. Interactively displaying maps on a tactile graphics display. In *Proceedings of the Workshop on Spatial Knowledge Acquisition with Limited Information Displays*, pages 13–18, 2012

(Section 7.2) is based on widely available and very cheap consumer hardware, whereas the second approach (Section 7.3) can profit from highly specific and expensive hardware.

7.1 Background

With the advance of widespread availability of digital maps, making them accessible for all is a logical next step. A sighted person observing any standard map can get an immediate two-dimensional overview of the general layout. But when acoustic or tactile maps are used, this immediate overview is not as easy to get. Even if tactile maps are laid out two-dimensionally and the user employs several fingers, the overview is limited. Therefore most approaches to digital maps for blind users employ a cursor or "virtual observer" that can be moved across the (virtual) map. Feedback is only given about the area under the cursor or immediately surrounding it. Accordingly, only one point (or its immediate surroundings) can be observed at any given point in time. Getting an idea of a larger area therefore necessarily involves moving the cursor. This is equivalent to a single finger moving across an embossed tactile map. The distinction of whether a system uses a virtual observer or not, can be used as a classification. However, as most accessible maps rely on this technique, the sense that is used to convey the desired information, normally hearing or touch, is a more natural distinction.

Auditory Maps Heuten et al. [39, 40] play different sounds corresponding to different locations on the map, where the connection between the location and the sound can either be natural (e.g. a water sound for a lake) or artificially created. These sounds are then played in a 3D environment in which the user can locate their source. The approach by Cohen et al. [26] was not primarily developed with the intention to display maps, however it can be used to do so. It was developed to convey graphs to blind users by representing edges and vertices by sound. Zhao et al. [125] use auditory maps to present the geographic distribution of statistical data (e.g. number of inhabitants) but do not concentrate on exploring standard city maps.

Tactile Maps In contrast to purely auditory maps, which are normally bound to the virtual observer model, previous work on tactile or combined tactile and auditory maps varies between those using a virtual observer model and those representing a larger overview at once. Rice et al. [83] combine tactile and auditory feedback in their system that controls the virtual observer by a force feedback mouse. Wang et al. [114] print out

Figure 7.1: Rumble Gamepad. Both analog thumbsticks are used for moving the virtual observer, the buttons are used for various functions.

a certain area of the map with a thermal embosser. The printout is then placed on a touchpad, allowing audio feedback upon the touch of the user's finger. A similar approach was used by Paladugu et al. [78] with the goal of evaluating design patterns for the production of tactile maps. The system by Zeng et al. [120, 124] is most similar to the one presented in Section 7.3, also using a tactile graphics display. The system uses a GIS that features imported OpenStreetMap data and renders roads as lines and buildings and other points of interest as fixed symbols from a library. It therefore has advantages regarding readability of the maps, because specific types of map objects can be distinguished directly by their symbols, but is not as flexible, as the approach presented here.

7.2 Displaying Maps on a Rumble Gamepad

The general idea for the approach taken in this section is that the hardware that is used to give feedback about an area should be portable and as cheap as possible, and capable of being integrated into a white cane for mobile use. For this reason, the concrete implementation uses only a standard rumble gamepad (Figure 7.1) and a text-to-speech engine. The combination of these two is used to convey an idea of the map.

7.2.1 Standard Map Interaction

As discussed in section Section 7.1, many accessible maps use a virtual observer model that allows the inspection of the area at the observer's location, and the system presented in this section is no exception. The user can move the virtual observer across the map using the gamepad's controls.

A vibration is emanated from the gamepad whenever the virtual observer is on or near a street. This conveys a basic idea of the layout of the streets. The user can then inquire about the name of the current street with a press of a button. The name is output by a text-to-speech engine or a Braille device. However, due to hardware limitations, there are some additional tweaks that need to be introduced before the tactile map is usable.

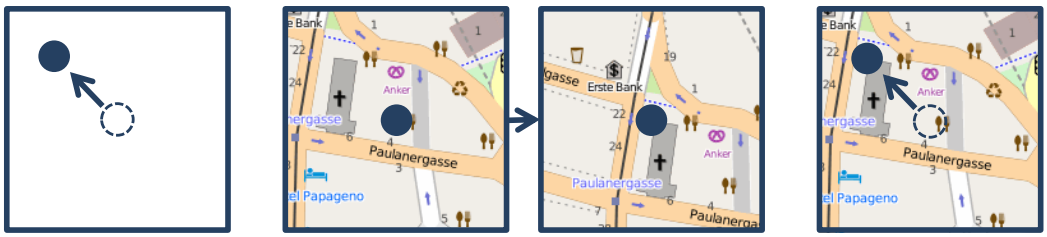
7.2.1.1 Moving the Virtual Observer

The virtual observer has to be moved by the user. Moving it with an analog stick of the gamepad is an obvious choice. However, there are two fundamentally different ways of implementing this movement, as shown in Figure 7.2.

The intuitive implementation is to allow the user to move the virtual observer freely across the map (Figure 7.2b). This is akin to scrolling, and this implementation was therefore named “scrolling movement”. As an analog stick is used, the speed of the movement can be made adjustable by linking it to the inclination of the stick. This implementation has the disadvantage of not having a fixed mapping of the stick position to a map position, which can make the examination of an area rather unintuitive.

The other possible implementation is to directly map a stick position to a map position (Figure 7.2c). In this implementation, the center position of the stick is fixed to a position on the map. Moving the stick results in a movement in a fixed area around that position. Therefore this implementation was named “fixed area movement”. As an example, pushing the stick slightly up will result in the virtual observer going to a position slightly to the north of the center position. Pushing it all the way down will result the virtual observer going to a position further to the south. The obvious disadvantage of this implementation is that it limits the area of observation. However, as each position of the stick is directly mapped to a position on the map, the examination of a certain area is much easier than when using scrolling movement. Another advantage of fixed area movement is that the center position can be fixed to the current position of the user if used in a navigation system. This allows a quick and easy exploration of the area immediately surrounding the user.

As both implementations have their pros and cons, both versions were included in the prototype. Each version was mapped onto one of the two analog sticks. This enables the user to scroll quickly across the map and then examine a certain area in detail. The one that allows free movement is called the scrolling stick and the one that allows for the detailed examination of a certain area is called the observer stick for the remainder of this chapter.



(a) Stick Movement *(b)* Scrolling *(c)* Fixed Area

Figure 7.2: Moving the Virtual Observer. Displayed are the effects of moving the thumbstick to the top left (a). With scrolling movement (b) the view continuously scrolls in northwest direction. Once the thumbstick is recentered, the virtual observer remains in its new position. With fixed area movement (c) the virtual observer is moved to a position to the northwest of its original place. Once the thumbstick is recentered, the observer returns to its initial position.

7.2.1.2 Vibration

As vibration is used to announce a street, the different possibilities of implementing vibrations need to be discussed. It might seem obvious, that vibration should occur when the virtual observer is on or near a street. However, as the test implementation showed, this mode of operation has serious drawbacks. Because of the latency inherent to the hardware, it is possible to move the observer from one side of a street to the other without receiving any feedback from the device. The resulting effect is similar to streets appearing and disappearing at random. This seriously hampers the ability to build a mental map of a certain area. Fortunately, this problem can be solved easily: Check whether a street has been crossed and in that case emanate a short vibration regardless of the current position of the stick. The disadvantage of this solution is that it slightly reduces the connection between stick position and map position in the fixed area movement implementation of the virtual observer, because a vibration might occur when no street is beneath the virtual observer. However, compared to the disappearing streets problem mentioned above, this is a minor inconvenience.

7.2.1.3 Street Announcement

The virtual observer combined with the vibration gives the user an idea of the general layout of the streets. However, the names of the streets are not conveyed by the vibration and need to be announced by a text-to-speech engine or a Braille device. In the concrete system, the street name

is announced upon the user requesting it by pressing a button. Again, there are different ways of implementing this.

The intuitive implementation of street announcement always outputs the name of the street that is currently closest to the virtual observer. This works best together with the “Vibrating Near a Street” implementation. However, the user will make a connection between the vibration and the street name, which does not always reference the same street in this implementation. This problem can be overcome by announcing the street that was last crossed by the virtual observer. This street is not necessarily the one closest to the current position of the stick. However, this allows the user to build a direct connection between the vibration and the announced street name, improving the mental map built by the user.

7.2.1.4 Zooming

Exploring a map does not only involve scrolling around but also zooming. Zooming out has different effects on scrolling and fixed area movement. When zoomed out, the scroller stick will scroll over a larger area in the same time and with the same inclination of the stick. The observer stick will cover a large area when zoomed out. This has a negative effect on the facility of inspection of the area, as more and more streets are accumulated in the area covered by the stick.

Therefore fewer details are displayed when zooming out. All streets are displayed when zoomed. Residential roads are removed when zooming out. Upon zooming out even more, all roads are removed and only towns or suburbs are displayed, with their borders activating the vibration. This loss of details is equivalent to the reduced details in small scale maps as compared to large scale maps. Zooming with reduced detail allows the user to both gain an overview and get more detailed information about certain areas.

7.2.2 Advanced Map Features

The techniques presented so far can convey an idea of the layout of a map. However there remain several problems which need to be overcome.

7.2.2.1 Intelligent Zoom

First there is the problem of cluttering the output with too many streets. Often there are still more roads than can be reasonably handled with a small analog stick on many zoom levels, even if roads are masked when zooming out. A minimal movement of the stick will result in the virtual observer

crossing several streets, making a clear distinction between these hard, if not impossible. However, on certain zoom levels there is an appropriate number of streets within reach of the observer stick. Therefore, presenting the user only with those views that deliver a satisfying result is a reasonable solution to the problem of “street cluttering”.

Unfortunately there are no fixed zoom levels that will always yield satisfying results. The optimal zoom levels depend on the density of the streets in the observed region. For this reason an intelligent zoom was developed, that will always zoom to a level that displays a reasonable number of streets, main streets or suburbs.

7.2.2.2 Single Street Observation Mode

The methods described above are well suited for getting an overview of the general layout of streets in an area, and for finding out which streets lie next to each other. However, a user inspecting a map will often also be interested in the exact layout of a specific street. This can become problematic when streets are bent or winding. Following the street with the observer stick becomes hard or impossible, especially when it is connected to many side streets. For this reason, the “Single Street Observation Mode” was developed.

When the user presses a button, all but the last crossed street are removed from the display. Upon the button press the name of the street is announced so that the user is aware which street is currently being observed. At the same time, the vibration mode is switched to “Vibrating Near a Street” (see Section 7.2.1.2). The street can be displayed wider, i.e. the distance to the street necessary for vibration is increased. As it is now the only street being displayed, this cannot lead to problems related to overlapping streets but leads to an easier observation of the street.

7.2.3 Evaluation

As a way of evaluating both concept and implementation, a small study with four participants was conducted. All participants were sighted but could not see any output on the screen. Two of the four participants had never used a gamepad with analog sticks before, one participant had used one once several years before, and one was a regular user of such a gamepad. Two tasks were designed with the goal of testing the intelligent zoom and Single Street Observation Mode.

The first task was to name the southern parallel street of a given street. A rough location description for these streets was given which consisted of the suburb and a triangle of major streets in which they were located.

Table 7.1: Results of Task 1. Results for the task designed to test Intelligent Zoom.

Mode	Intelligent Zoom		Fixed Zoom	
Participant	1	3	2	4
Gamepad Use	Never	Once	Never	Regularly
Time (min:sec)	11:06	10:55	20:00+	8:32
Correct Answer	Yes	Yes	No	Yes

At the beginning of the task, the virtual observer was located in another suburb.

The second task was to roughly draw the course of a certain street (which has one notable turn). For this task the virtual observer was already located on that street.

7.2.3.1 Results for Intelligent Zoom

Table 7.1 shows the time that was needed by each participant to finish the first task and whether the result was correct. Participants 1 and 3 (using Intelligent Zoom) were able to complete the task correctly in about 11 minutes each. Of those without fixed zoom participant 2 canceled the task after 20 minutes without a result; participant 4 was able to complete the task in 8:32 minutes. Thus, there is no clear conclusion about the effects of Intelligent Zoom. It can be suspected that participant 4 profited from his regular use of a gamepad with two analog sticks and would have been even faster with Intelligent Zoom.

7.2.3.2 Results for Single Street Observation

The results for Single Street Observation are shown in Table 7.2. Participants 1 and 2 (both using Single Street Observation) were able to complete the second task, taking 4:10 and 9:10 minutes. Of those with no Single Street Observation, participant 3 stopped after 5:41 minutes without a correct result and declared the task to be “very difficult”. Participant 4 took 10:43 minutes for the correct result. Again it has to be kept in mind that participant 4 was the only proficient dual analog stick user. Participant 4 suggested a mode similar to Single Street Observation for this task on his own. These results can be considered as a hint that Single Street Observation mode is indeed an improvement that makes it possible to inspect the layout of a street.

Table 7.2: Results of Task 2. Results for the task designed to test Single Street Observation.

Mode	Single Street Observation		No Single Street Observation	
Participant	1	2	3	4
Gamepad Use	Never	Never	Once	Regularly
Time (min:sec)	4:10	9:10	5:41	10:43
Correct Answer	Yes	Yes	No	Yes

7.2.3.3 General Results

The study has shown that it is possible to inspect maps using a standard rumble gamepad with the concrete implementation. Furthermore, both the results and the observations during the study seem to suggest that routine in using analog thumbsticks is an important factor regarding the efficiency of the system's use. One user suggested a bookmark function that would allow the user to save the current location of the virtual observer and quickly jump back to that location. However, the time it took participants to complete the tasks is prohibitively long for mobile use of this system. For this reason, the idea to integrate this approach into a white cane was not pursued further. After this approach was originally developed, touchscreens have become widely available in smartphones, including the possibility of vibrating feedback. Thus, there are now better, if not yet cheaper, alternatives for conveying maps.

7.3 Versatile Displaying of Maps on Touch-Sensitive Displays

Touchscreens have become widely available in the last few years, and are a cheap, non-specialized hardware. At the same time, specialized hardware, such as tactile graphics displays, have also integrated touch-sensitivity. The commonality between the two, is that—in contrast to other solutions, such as a mouse-keyboard combination—there is a direct spatial relationship between the input they receive and the output they produce. This spatial relationship make them a good choice for displaying maps for blind people, allowing a direct link between an exploring finger and the shape and position of an underlying object.

This section presents a versatile system for displaying maps on touch-

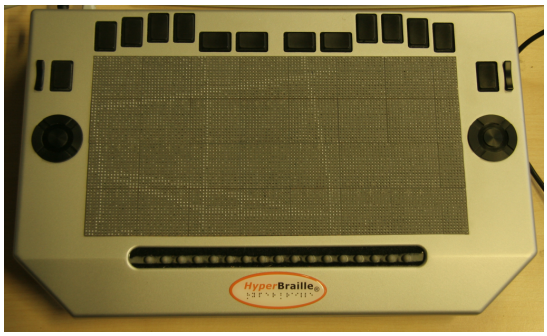


Figure 7.3: The Tactile Graphics Display. It consists of a pin-matrix display, a braille keyboard on top and two four-way digital crosses.

sensitive displays. It describes the basic implementation that is independent of the device, and explains differences that depend on specificities of the devices, and how those influence the interaction of the user with the system. The basic idea of the system is based on the “tactile-acoustical map” by Hub et al. [51].

7.3.1 Display Devices

The system was tested with two distinct display devices. The more commonly used one was a 12.1" touchscreen with a native resolution of 1280x800. For blind users, this device was used in fullscreen mode, so that no windows controls interfere with the exploration of the map. The specialized device, a pin-matrix display, also called tactile graphics display, has a resolution of 120x60 pins and touch-sensitive sensors for feedback about the position of the user’s fingers. The device is shown in Figure 7.3. For the display of the maps, the driver of the pin-matrix display converts the normal graphic output by interpolating it for display on the specific low resolution. With this approach, the system is not limited to a specific tactile graphics display, instead device any that can convert on-screen graphics can be used. Because of this approach some graphical details might be lost during the conversion. This can mostly be avoided by choosing appropriate styles for the graphics display (Section 7.3.4.1), or by setting the display window of the system to the resolution of the tactile graphics display, so that each pixel on screen is represented by exactly one pin on the tactile graphics display. Similarly, the position of the user’s finger is (upon pressing of a button) converted into a normal mouse click event, which is then handled by the application.

7.3.2 Maps

The systems displays two different kinds of maps: Highly detailed maps of buildings and small outdoor areas and OpenStreetMap data for an overview of large outdoor environments. The detailed maps are the same maps described in Section 3.3.1.1 on page 34. For large outdoor areas where no detailed maps are available, OpenStreetMap data is used. A certain area around the current viewport is downloaded. If the user scrolls out of that viewport, new data is downloaded automatically.

7.3.3 Interactions

As in any common map application, the user may zoom and scroll the map by using the shortcut keys on the keyboard or the four-way digital cross of the tactile graphics display. The specific feature of the system is the interaction of the exploring finger with the map objects. This interaction is implemented slightly differently for the touchscreen and the tactile graphics display.

On the tactile graphics display, a user can notice immediately an object under his finger, as the pins are raised (or lowered, depending on style) at the specific position. Therefore it is sufficient that the system react to user input: if the user presses a button, the name, function or address of the object is read out by a text-to-speech engine. As streets, represented by polylines, can be very thin, and the resolution of the touch-sensitivity of the pin-matrix display is lower than its pin resolution, performing this action a configurable distance away from a street is still considered as a click on this street.

On a standard touchscreen, however, the user gets no immediate feedback by the hardware on whether there is an object under his finger. Therefore a sound feedback was integrated, which emits a clicking noise, if a user enters the area of a map object, or changes the current object, while moving across the screen with his finger. In addition to that, the name, address, or description of the object is immediately output as well, as this was preferred by a blind user that gave immediate feedback about the system. Another distinct sound is played, if the user leaves a map object. At any time the user can lift his finger for a short time in order to repeat the textual information about the object.

Determining the text that needs to be output is straightforward, if one of the purpose-built maps is used, as all displayed objects are named. However, with OpenStreetMap, it is not always clear, what information should be output. Therefore, in order to be helpful to the user, the text that is read out has to be chosen more diligently: If the object is tagged with a

name tag, the name is read out directly. If no name is given, a combination of the type of the object and (if available) the address is read out. The type of the object is determined by the OpenStreetMap tags in combination with strings from the styles (Section 7.3.4.1). This setup also allows a translation into other languages, by storing the translation of map object types in the style. If several objects are stacked on top of each other in OpenStreetMap, the objects are read out one by one after each consecutive click, beginning with the innermost. For detailed maps, consecutive clicks are equivalent to going up one step in the GML hierarchy, so that e.g. a building's name will be announced after a room number.

Some features of the system can be accessed by a menu. The menu has been designed to work on both the tactile graphics display and a touchscreen. Four actions are used to control the menu: *Up* and *down* scroll through a menu, *right* selects an item or opens a submenu, and *left* leaves a submenu. These actions are mapped onto the cursor keys of the keyboard, one of the four-way crosses of the tactile graphics display, and onto performing directional swipe gestures on the touchscreen. The individual menu items are read out by the text-to-speech engine. The menu can be opened with a keystroke, or by an optional button in the corner of the touchscreen, if no hardware keys are available. It allows access to the various maps (Section 7.3.2), the different styles (Section 7.3.4.1), and to the place search (Section 7.3.4.2).

7.3.4 Additional Features

While the system as described above is functional, some additional features were implemented in close collaboration with a blind colleague, who used the system most frequently. Those features can greatly enhance the usability of the system.

7.3.4.1 Display Styles

The display component of the system is completely configurable. This means that for all tags in OpenStreetMap the color and thickness of the lines that are drawn, as well as the color and hatching of polygonal objects can be freely chosen. Objects can also be completely hidden, depending on their tags. Details like color and hatching cannot be reproduced directly on the tactile graphics display, and are mainly useful for collaboration with sighted users that use a monitor. Line thickness or the hiding of objects can be used to avoid clogging the tactile graphics display with too much information. These style settings can be stored and loaded and also added to a quick styles menu, allowing easy selection of different styles, such

as “only buildings” or “only streets”. Figure 7.4 shows a detail of an area near the campus of the University in Stuttgart-Vaihingen, showing both buildings and streets (a, b), only buildings (c, d), and only streets (e, f).

7.3.4.2 Place Search

The place search allows entering the name of cities, streets or well-known entities such as landmarks. The search string is forwarded to both GeoNames[1] and OpenStreetMap Nominatim[2]. The places found by both services are presented to the user in an accessible list, that can again be accessed with the cursor keys. Upon selection of an entity, the map is changed to OpenStreetMap (if it is not already the case), and centered on the geographic position of the selected entity. This allows the user to quickly switch between different areas of interest.

7.3.5 Results

First tests have shown that the system can greatly enhance the spatial understanding of its users. A blind colleague, who used the system, says that she has had to correct her understanding of the street layout even of areas that she has lived in for a long time. The possibility to quickly jump to a desired location was regarded positively, as it enabled the user to go from exploring one area, such as the workplace, to another, e.g. the place of residence. Furthermore, the different styles were regarded as very helpful to reduce information overload depending on specific tasks, e.g. the colleague chose to hide all buildings when exploring the layout of streets. The use of OpenStreetMap has the advantage of providing a worldwide data set. Especially in conjunction with the flexibility achieved by the different styles, this provides a main difference from many of the previous works in this area. The system presented in this chapter provides the background for the next chapter, where it will be shown, how Map Content Transformations can have a further positive influence on position feedback.

7.4 Discussion

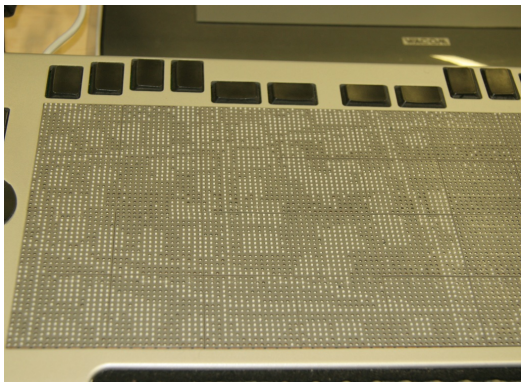
In this chapter, several possibilities of relaying map information to the users have been introduced. In all of those approaches, the information that is being presented to the user is taken directly from the underlying

[1] www.geonames.org (last accessed 2014-10-03)
[2] http://nominatim.openstreetmap.org/ (last accessed 2014-10-03)

map, and thus can only integrate data that is directly present in that map, even though other information, that might only be present implicitly, might also be beneficial for the user. Thus, the next chapter will concentrate on integrating Map Content Transformations with the approaches presented here, in order to further enhance position feedback.

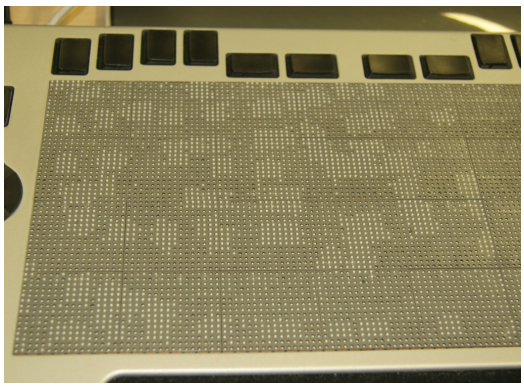
The use of a single rumble gamepad, originally considered for mobile use, proved to be too time consuming, compared to using a touchscreen or a tactile graphics display. For this reason, the following chapter will employ only the second alternative.

(a) Buildings and streets (screen)

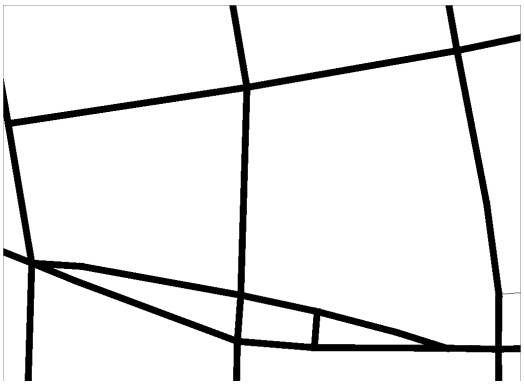


(b) Buildings and streets (tactile graphics display)

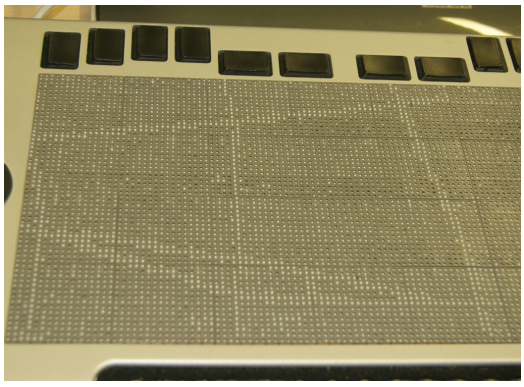
(c) Buildings only (screen)



(d) Buildings only (tactile graphics display)

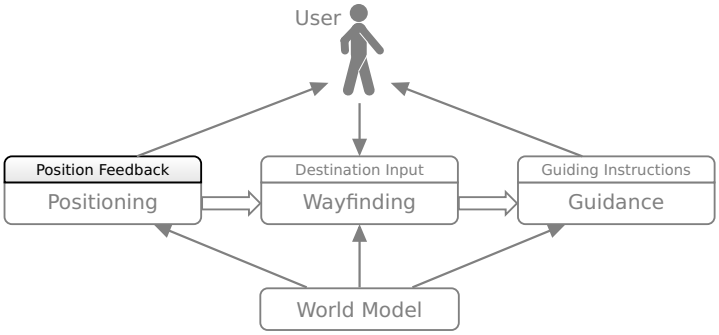


(e) Streets only (screen)



(f) Streets only (tactile graphics display)

Figure 7.4: Map with Three Different Styles. Each style is shown on both the computer screen and the tactile graphics display.



8 Map Content Transformations in Position Feedback

In sections 7.2 and 7.3, two distinct methods of displaying maps to blind users were presented. The second approach is already customizable by applying different styles, but these customizations are always restricted to the style of displaying an object or the decision whether to display it at all. As shown in Chapter 4, Map Content Transformations can be used to create individualized maps. Accordingly, they are a suitable tool to further individualize the presentation of maps. Thus, it becomes possible to convey information that is specifically relevant to the user group, helpful for understanding the environment, or implicitly understandable when looking at a visual map, but not apparent when using some other mode of map exploration. Often, this kind of useful information is not explicitly present in the original data, but can be derived from it, if combined with the information coded into the transformation rules set. Most concrete use cases will combine those different aspects.

8.1 Implementation

As the choice of transformation is entirely dependent on the specific situation, there are no general rules on what constitutes a good transformation. Therefore in the following some examples of transformations that satisfy different aspects are given. These were developed specifically with blind

Parts of this chapter have previously been published in:

Bernhard Schmitz and Thomas Ertl. Creating task-specific maps with map content transformations. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on MapInteraction (MapInteract '13)*, pages 84–90, 2013. DOI:10.1145/2534931.2534945

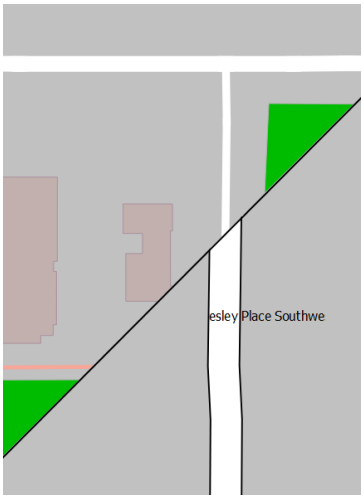
users in mind, and illustrate a possible benefit that this specific user group can gain through Map Content Transformations.

8.1.1 Creation of Polygonal Representation

As explained in Section 7.2.1, many digital maps use a “virtual observer” that gives feedback about the area at its position. Concrete implementations of this may vary, but it is very simple to implement a system that gives feedback only about polygons that enclose the position of the virtual observer. Supporting polylines in conjunction with a virtual observer often imposes certain restrictions or workarounds, such as those explained in Section 7.2.1.3. Even if the system itself is capable of a very good coupling between the position and the feedback, supporting polylines introduces problems: Feedback needs to be given within a specific distance to the polyline. If this distance is set globally, differences in width, e.g. between streets and sidewalks cannot be represented. If distances can differ between types, this either needs to be hardcoded into the system that outputs the information, which is contrary to the goal of individualization by transforming only the map, or the distances need to be stored in the polylines themselves, introducing problems with efficiently storing and retrieving them, as spatial access methods would need to be informed about this difference in

```
<rule order="10">
  <identify>
    <way id="residential_way">
      <tag k="highway" v="residential"/>
    </way>
  </identify>
  <construct>
    <hull_polygon on="residential_way"
     from="first" to="last"
     type="way_rectangle" width="8">
      <tags is="residential_way"/>
    </hull_polygon>
  </construct>
</rule>
```

(a) The rule identifies residential ways and creates polygonal representations. Each polygon spans the entire length of the original way and receives its tags.



(b) The original map (top) and the transformed map (bottom), which consists only of the polygon created with the rule in (a).

Rule 8.1: Creating Polygonal Representation of an OpenStreetMap Way. In this example, only residential ways are transformed.

types, e.g. by considering the difference when computing bounding boxes. Allowing only polygons therefore simplifies matters considerably.

Regarding the system created for this thesis, the implementation for the tactile graphics display uses a simple, globally set distance (see Section 7.3.3). The direct sound feedback for touchscreen devices does not work with untransformed streets, as it requires polygonal objects, in order to determine entering and leaving.

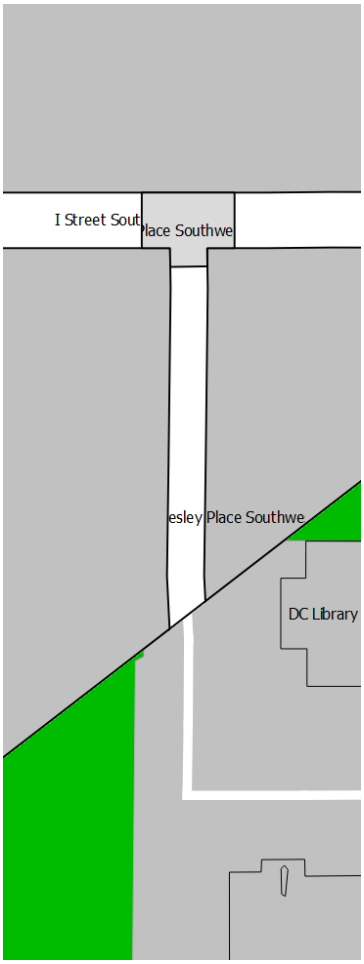
Many relevant entities are represented in OpenStreetMap as polylines (ways). Converting those polylines into polygons is one of the important aspects of Map Content Transformations for position feedback. Rule 8.1 shows an example of a rule that converts a residential way into a polygonal representation. In this example, the width is not stored in a tag and therefore an approximation is encoded into the rule.

8.1.2 Designation of Intersections

One of the disadvantages of having to use a virtual observer for a map is that it cannot immediately convey the layout of a map in the way of a two-dimensional visual map. While inspecting the map with a virtual observer, the user learns about the objects on the map and their rough spatial relationship to each other, but much information needs to be deduced slowly during the exploration process. Some of that kind of information can be integrated more directly into the map, such as information about intersections of streets. Intersections can be seen immediately on a visual map. However, when exploring a map with a virtual observer, they have to be deduced from the layout of the streets. By adding specifically designated zones at the intersections, the user can immediately perceive the intersections when inspecting a street, without having to analyze the layout of nearby streets. Rule 8.2 shows a rule for creating a designated intersection zone at the intersection of a residential and a tertiary highway. Again, it is assumed that the width of the street is not given in the tags. The different types of the two intersecting streets are reflected in their different widths. The rule constructs a polygon, the name of which indicates the intersection and the names of the two streets that intersect here.

```
<rule order="10">
<identify>
  <way id="residential_way1">
    <tag k="highway" v="residential"/>
    <tag k="name" v="*" id="rw_name1"/>
    <node id="intersec"/>
  </way>
</identify>
<identify>
  <way id="tertiary_way">
    <tag k="highway" v="tertiary"/>
    <tag k="name" v="*" id="tn"/>
    <node is="intersec"/>
  </way>
</identify>
<construct>
  <intersection_polygon on="intersec"
   along1="residential_way1"
   along2="tertiary_way"
   type="intersection_rectangle"
   width1="8" width2="12" length="10">
    <tag k="highway" v="tertiary"/>
    <tag k="name"
     v="Intersection {rw_name1} {tn}"/>
  </intersection_polygon>
</construct>
</rule>
```

(a) The rule identifies intersections of tertiary and residential roads and creates intersection areas.



(b) The transformed map (top) contains street polygons and the intersection area created in (a).

Rule 8.2: Creation of Specially Designated Intersection Areas. Intersection areas can help orientation in a virtual observer based environment, where only a specific point of the map is conveyed to the user.

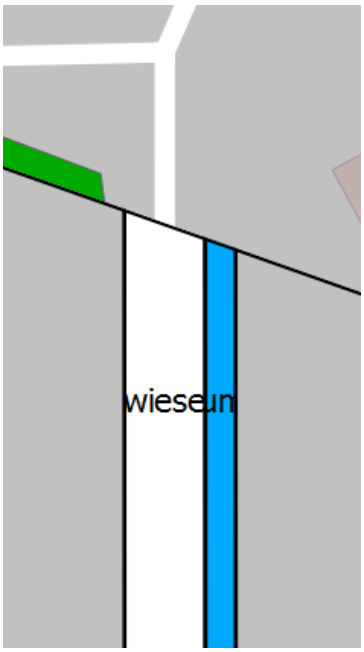
8.1.3 Addition of Sidewalks

Knowledge about the existence and location of sidewalks can be helpful for blind pedestrians. Explicitly representing the sidewalks is therefore a good example of how Map Content Transformations can help to build maps for specific purposes. In the original OpenStreetMap data, sidewalks are only represented as tags of ways. Rule 8.3 presents a rule for residential ways with a sidewalk on the left side, and how a polygonal representation of the sidewalk is created. Left and right sides of a street are determined

by the inherent direction of ways in OpenStreetMap (see Section 4.1.1 on page 45), which is why the sidewalk is on the right side from the position of view in the result.

```
<rule order="10">
<identify>
  <way id="left_sidewalked_way">
    <tag k="highway" v="residential"/>
    <tag k="sidewalk" v="left"/>
    <tag k="name" v="*" id="swname"/>
  </way>
</identify>
<construct>
  <hull_polygon on="left_sidewalked_way"
   from="first" to="last"
   type="way_rectangle"
   width="3" distance="-8/2-3/2">
   <tag k="name" v="Sidewalk {swname}"/>
   <tag k="sidewalk" v="yes"/>
  </hull_polygon>
</construct>
</rule>
```

(a) The rule identifies streets that are explicitly tagged as having a sidewalk on the left side and creates aptly named polygonal representations.



(b) The transformed map (bottom) contains the sidewalk polygon created in (a).

Rule 8.3: Adding Sidewalks to Streets. Knowing the positions of sidewalks beforehand can ease navigation in a certain area.

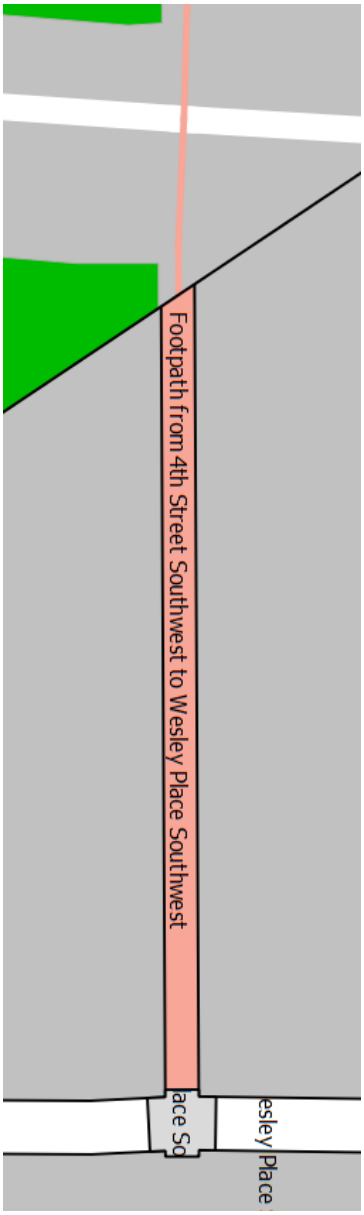
8.1.4 Addition of Descriptive Names

A large number of objects that do not have names is present on maps, and OpenStreetMap is no exception to that. The benefit of their existence on the map is a combination of their denoted function (often represented by the rendering in the map style) and their spatial relationship to other objects, which is immediately apparent on visual maps. For example, there are many unnamed footpaths, but a person looking at a visual map can immediately see which streets they connect and therefore whether they would make an appropriate shortcut. However, this spatial relationship is often lost when using a virtual observer to explore the map. If the virtual observer is above the footpath, the user will only get this functional description (“footpath”), which is lacking the contextual information. Rule 8.4 identifies footpaths that do not have a name and connect two different roads. A polygonal

representation of the footpaths is then created and a descriptive name that includes both streets is given to it.

```
<rule order="10">
<identify>
  <way id="footway">
    <tag k="highway" v="footway"/>
    <not> <tag k="name" v="*"/>
    </not>
    <node id="node1"/>
    <node id="node2"/>
  </way>
</identify>
<identify>
  <way>
    <tag k="highway" v="*"/>
    <tag k="name" v="*" id="name1"/>
    <node is="node1"/>
  </way>
</identify>
<identify>
  <way>
    <tag k="highway" v="*"/>
    <tag k="name" v="*" id="name2"/>
    <node is="node2"/>
  </way>
</identify>
<construct>
  <hull_polygon on="footway"
   from="first" to="last"
   type="way_rectangle" width="5">
    <tags is="footway"/>
    <tag k="name"
 v="Footpath from {name1} to {name2}"/>
  </hull_polygon>
</construct>
</rule>
```

(a) The rule identifies unnamed footpaths and creates polygonal representations with an appropriate name.



(b) The transformed map (bottom) contains a named footpath polygon created in (a).

Rule 8.4: Creation of Named Footpath Polygons. Naming Footpaths in this way makes previously implicit information directly accessible.The implicit information is easy to recognize on a visual map, but difficult to handle for blind people.

8.1.5 Designation of City Blocks

The space between streets in OpenStreetMap is sometimes populated with building polygons, sometimes left empty. However, even if building polygons do exist, it can be a better choice to not display them, depending on the preferences of the user. In many cases, these polygons do not include any additional information, and whether to display them or not might depend on the task at hand, as explained in Section 7.3.4.1. When buildings and similar entities are not being displayed, there are large empty spaces between streets. Those empty spaces pose no problem in a visual representation of a map, but they do not provide feedback to the user when using a virtual observer. They are therefore a good example of the different requirements of different interaction modes that can be fulfilled with Map Content Transformations: Rule 8.5 creates designated city blocks from a quadrilateral defined by four streets. The names of the streets are reflected in the designation of the city block. While the newly created map entity does not have any counterpart in the real world, it describes the area and thus can give an immediate feedback about the current position of the virtual observer that would otherwise be missing from the map.

8.2 Evaluation

Evaluating Map Content Transformations for position feedback is difficult for several reasons. One reason is that maps in general—and therefore transformed maps as well—are the backend of applications. However, when evaluating the position feedback, it is primarily the front-end that is being evaluated, i.e. it is impossible to evaluate the use of Map Content Transformations without inadvertently evaluating the entire system.

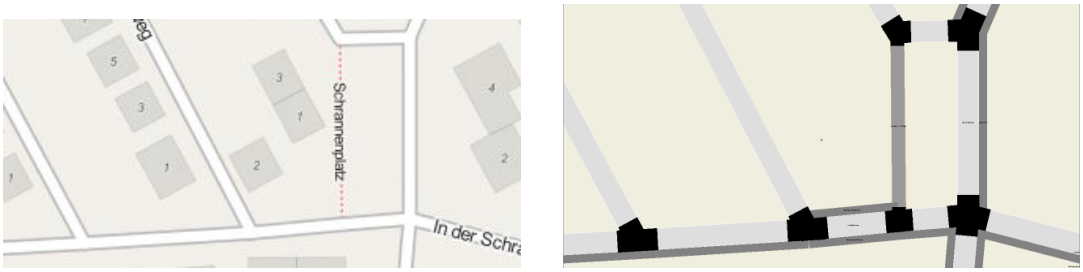
An additional problem of the evaluation is that the system is freely scriptable. Again, this means that not only the system itself, but also the scripts that are being used will be evaluated. In consequence, if an evaluation shows that something that was presented to the users is not helpful, it could mean that only the specific transformations chosen during the evaluation are not useful, and that other transformations, more adapted to the specific user, could still be useful.

Keeping in mind all of the above, an evaluation can at best show that transformed maps can be helpful to some users. Any stronger claims would not be sound, while a result that does not show any helpfulness would not prove the uselessness of the map transformations as well.

The evaluation is based on the transformations from Section 8.1. The participants were asked to perform several tasks on the transformed maps,

```
<rule order="10">
<identify>
  <way id="way1">
   <node id="n1"/>
   <node id="n2"/>
   <tag k="name" v="*" id="way1_name"/>
   <or>
     <tag k="highway" v="residential"/>
     <tag k="highway" v="secondary"/>
     <tag k="highway" v="tertiary"/>
   </or>
  </way>
</identify>
<identify>
  <way id="way2">
   <node is="n2"/>
   <node id="n3"/>
   <tag k="name" v="*" id="way2_name"/>
   <or> [...] </or>
  </way>
</identify>
[...]
<construct>
  <way polygon="closed">
    <for type="node" id="cn1" on="way1" from="n1" to="n2">
      <node is="cn1"/>
    </for>
    <for type="node" id="cn2" on="way2" from="n2" to="n3">
      <node is="cn2"/>
    </for>
    <for type="node" id="cn3" on="way3" from="n3" to="n4">
      <node is="cn3"/>
    </for>
    <for type="node" id="cn4" on="way4" from="n4" to="n1">
      <node is="cn4"/>
    </for>
    <tag k="name" v="Area between {way1_name} and {way3_name},
     as well as {way2_name} and {way4_name}"/>
  </way>
</construct>
</rule>
```

Rule 8.5: Creation of Designated City Blocks. The city blocks serve as an orientation aid for virtual observer based map displays. Identification has been shortened by three similar rules that identify streets containing one node of the previous street and the last street containing node n1 of the first street.



(a) The original OpenStreetMap representation.

(b) The transformed version of the map in (a).

Figure 8.1: Detail of a Map used in the Study. Transformation 3 in the user study displayed maps with intersections and sidewalks. In order to make the areas distinguishable in the figure, different colors were assigned to them (streets: light gray, sidewalks: dark gray, intersections: black). The sidewalks were created from the “sidewalks” tag.

and to give their opinion on the map features that were introduced with the transformation.

8.2.1 Participants

The user study was conducted with five participants. These are not enough participants for a statistically sound analysis regarding task performance. However, it is possible to determine whether certain tasks are achievable by at least some blind users. For the reasons described in section 8.2 this can be considered as sufficient, as the experiment is influenced by so many factors that conclusions beyond this would be questionable anyway. However, this does not influence the results regarding the map transformations. At most this means that the first participant could have had a better experience of interacting with the system.

The participants were between 40 and 70 years old, two were female, three male. All were legally blind, four since birth, one became blind at 14 years of age. Two subjects had minimal eyesight left, allowing for bright/dark distinction. The participants worked between 15 and 40 hours per week on a computer. Two participants said they had much experience with standard tactile maps, one participant had medium, one a little experience, and one participant had no prior experience at all. Regarding computer-assisted tactile maps, one participant had much experience, and all others had no experience at all. All participants were fluent in German, which is why the maps, transformations and questions were in German.

8.2.2 Procedure

For the user experiment, the touchscreen device described in Section 7.3.1 in conjunction with the interaction techniques from Section 7.3.3 was used. First, all participants were told that they could take a break or leave the study at any time. After they had answered the questions regarding age, blindness and prior experience with tactile maps, the system was explained to them. After that they were given as much time as they wanted to acquaint themselves with the system using a test map that displayed only streets. No participant took longer than five minutes to try out the system. After that, the tasks described in Table 8.1 were given to them, one after the other. Participants were encouraged to speak their mind during the entire time of the experiment. After each task, opinion questions about the specific transformation used in the task were asked. After all tasks were finished, general questions were asked and additional comments welcomed.

8.2.3 Results

Although the time that participants took to finish a task was taken, it quickly became apparent that the time was more influenced by random factors like “does a user find a specific street quickly by chance?” than by the difference between the transformations. As a result, those times are not meaningful for evaluation and not mentioned any further here. The focus of this evaluation is therefore on the effectiveness rather than the efficiency of the transformed maps, as well as on the user opinions. To give a general idea, participants took between 30 seconds and 5 minutes for each task.

8.2.3.1 Task Performance

All participants were able to name the streets connected by a specific street both with transformation 1 and transformation 2. Three participants were able to find the traffic lights with any of the transformations, however all agreed that the area of the traffic lights was too small. Three of the five participants could complete both tasks regarding sidewalks. Three participants named all four city blocks meeting at the intersection with transformation 4. The two other participants named one or two of the blocks.

8.2.3.2 Questionnaire Results

With five participants in the user study, a statistically relevant analysis of the data is not possible. Therefore, the complete result set is shown in Table 8.2.

Table 8.1: Transformations, Tasks and Questions. The Transformations (Tx) that were used are described in the section referenced in the leftmost column. The tasks that the users were asked to perform with each transformation and the questions they were asked are designated TxTx and TxQx, respectively. General questions asked after all tasks were finished are designated GQx.

		Tasks and Questions
8.1.1	T1T1	Find specific street and tell which two streets are connected by the found street.
	T1T2	Find pedestrian traffic light on specific street and tell near which intersection it is located.
	T1Q1	I think that this way of displaying a map is helpful.
	T1Q2	I think it is possible to recognize the layout of the streets.
8.1.2	T2T1	Find specific street and tell which two streets are connected by the found street.
	T2T2	Find pedestrian traffic light on specific street and tell near which intersection it is located.
	T2Q1	I think that the specially designated intersections are helpful to recognize the layout of the streets.
	T2Q2	I think that the specially designated intersections help to save time.
8.1.3	T3T1	Tell on which side of a specific street the sidewalk spans the entire street.
	T3T2	Tell between which crossing streets a sidewalk also exists on the other side of the street.
	T3Q1	I think it is possible to recognize where the sidewalks are located.
	T3Q2	I think that displaying the sidewalks on the map is helpful.
8.1.5	T4T1	Name all four streets meeting at a specific intersection
	T4Q1	I think that the special designation of street blocks is helpful.
	T4Q2	I think that the special designation of street blocks help to recognize the layout of the streets.
	T4Q3	I think that the special designation of street blocks helps to save time.
	GQ1	I think that displaying the maps on the touchscreen is helpful.
	GQ2	I had problems with the maps being displayed on the touchscreen.

After the first transformation four participants tended to agree, or agreed completely that the layout of the streets could be determined. At this point two people were unsure whether this way of displaying maps was helpful, while the rest tended to agree or agreed completely. Four participants agreed completely that specially designated intersections were helpful, one tended to agree. One user was unsure whether the intersections could save time, the others tended to agree, or agreed completely. Four participants agreed completely that it is possible to determine the locations of sidewalks, the fifth did not complete the task and answer the question. Three agreed completely, that displaying the sidewalks was helpful, one tended to agree. All participants agreed completely or tended to agree that the designation of city blocks is helpful and makes it easier to understand the layout of the streets. One participant was unsure whether the designation of city blocks would help to save time, while the others agreed completely or tended to agree.

Regarding the general questions, three participants agreed completely that displaying the maps on the touchscreen was helpful, two tended to agree. The only question that resulted in completely different answers from the participants was the one regarding problems with the maps displayed on a touchscreen. One participant disagreed completely, two tended to disagree, and two agreed completely. The two participants who agreed were the first participant, who used the earlier version, and one participant who had little experience with maps of any kind.

8.2.3.3 Additional User Comments

The feedback from the participants was generally positive and encouraging. One participant said “I did not expect that this would work so well. Personally, I like the quality of the whole thing.” Another participant said “If you integrate all the things you have shown to me, and add additional information like bus stops, then you have really built a system that can help blind people.” Another one said “it would be outstanding if you can make that available”—almost all participants asked about the availability of the system. The difficulty of interacting with the touchscreen-based system was assessed differently among participants. One user said “I could follow the streets and the sidewalks”, while another user said “it is difficult to find out the direction of a street”. One participant said that “sidewalks are difficult, you have to take your time”. The users made many suggestions for improving the system. Among those were the ideas to indicate the house number at every intersection and to indicate whether a street was a dead end. Similarly, one user proposed to announce the information about traffic lights and zebra crossings together with the intersection. Another

Table 8.2: Questionnaire Answers. The number of participants who stated the respective agreement with each statement from table 8.1 is noted in each field.

Question	Agree completely	Tend to agree	Unsure	Tend to disagree	Disagree completely
T1Q1	1	2	2		
T1Q2	1	3	1		
T2Q1	4	1			
T2Q2	3	1	1		
T3Q1	4				
T3Q2	3	1			
T4Q1	2	3			
T4Q2	2	3			
T4Q3	2	2	1		
GQ1	3	2			
GQ2	2			2	1

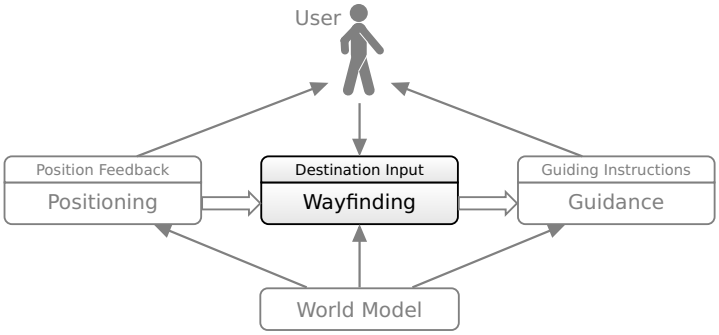
important suggestion was to increase the size of certain areas so as to not be true to scale, but improve the readability. The interface itself was also addressed, with the suggestion that the speech output should stop as soon as the finger leaves the area.

8.3 Discussion

One of the main reasons for conducting this user study was to find out whether rule based map transformations are helpful to blind people. As all map details that were created using transformations (intersections, sidewalks, city blocks) were regarded positively by all participants, it can be concluded that map transformations are indeed capable of producing maps that are more helpful to blind users than maps that are displayed without those transformations. It is interesting to note that the general question regarding the display of maps on touchscreens was answered more positively than the similar question after transformation 1. This could be either because the participants had in the meantime evaluated the different transformations that display additional data, or because the users

have generally become more acquainted with the system. All elements were displayed true to scale, which was not a good idea. Instead, it is more important to display elements in a size that makes them easy to recognize.

Many of the suggestions can be implemented by simply adjusting the rules of the transformations that are being used. This is true both for the additional information that was requested (bus stops, house numbers) and for the request to adjust the scale of objects that were displayed too small. Other suggestions, such as the stopping of the speech output, cannot be solved by additional rules, and require adjustments of the system used for map display.



9 Route Planning

After the position of a person has been established, the second stage of navigation becomes involved. Planning the route to the destination consists of two components: The destination must be input by the user, and a wayfinding algorithm needs to determine the best way to that destination, requiring a world model which supports route planning.

Algorithms for route planning are an entire field of research themselves, and beyond the scope of this thesis. Many of those algorithms are optimized for large graphs, e.g. the street network of all of Europe [8]. For pedestrian navigation systems, such heavily optimized algorithms are often not strictly necessary, due to the reduced scope of the route planning. The route planning described in this chapter is all based on a simple A* algorithm, as first described by Hart et al. [36], but could easily be replaced with a more advanced algorithm.

Destination input is of interest from two different perspectives: On the one hand, the modality of the interaction needs to be adapted to the specific requirements of the user. On the other hand, the possible destinations themselves, as provided by the world model, can be individualized according to specific criteria.

The remaining sections of this chapter will therefore concentrate on the individualization of route planning through Map Content Transformations, considering both destination input and wayfinding. There are two levels of the individualization of route planning: The first level are hard restrictions,

---

Parts of this chapter have previously been published in:

Bernhard Schmitz and Thomas Ertl. Individualized route planning and guidance based on map content transformations. In *Proceedings of the 14th International Conference on Computers Helping People with Special Needs (ICCHP 2014)*, pages 120–127, 2014. DOI:10.1007/978-3-319-08599-9_19

where certain conditions make it impossible for a person to use a specific path, for example the fact that a wheelchair user cannot use stairs. The second level are preferences, where the user is willing to take certain detours, if it helps to avoid inconvenience, but might still be able to take the inconvenient way, if the detour is too long. As an example, some users might prefer a traffic light to cross a larger street, if the detour is not too long. As shown in Chapter 2, there are already numerous projects that provide navigation for specific user groups. Especially related to this chapter is an OpenStreetMap-based routing interface for wheelchair users by Müller et al. [72]. However, those projects use algorithms that are tailored for their specific user group. Furthermore, they do not use all the data that is available in OpenStreetMap, and the routing that is available works only on street level, i.e. it does not take into account the existence or absence of sidewalks, the locations of pedestrian crossings, and similar data.

9.1 Creation of Route Graphs with Map Content Transformations

When talking about the data model of the world that is used by the routing algorithm, two separate types can be distinguished. The classic approach consists of a graph represented by explicit edges, denoting the walkable paths, and vertices at positions where those paths meet or are bent. When the world model consists of nothing but two-dimensional polygons, though, this approach cannot be used directly. Instead, there is need to implement ways of coping with that data model, such as proposed by Hong and Murray [44].

Route planning algorithms that work directly on the untransformed data model use the former approach, as OpenStreetMap's data format can be mapped onto a graph. On the other hand, the polygonal data created for the map feedback in Section 8.1 would require an implementation of the latter approach.

As compatibility of the route planning with the untransformed data was considered important, the implementation presented in this chapter uses the classic approach with vertices and edges. As a result, the original data can be used as a fallback solution in areas where the transformation rules are not applicable. As has been explained in Section 4.1.1, OpenStreetMap's data format consists of three kinds of entities: nodes, ways and relations. However, it is important to note that nodes and ways are not equivalent to the vertices and edges of a graph, as ways can span several nodes, but

graph edges are by definition the connection of two vertices. Therefore, in order to map OSM data onto a graph, the ways need to be divided at each node.

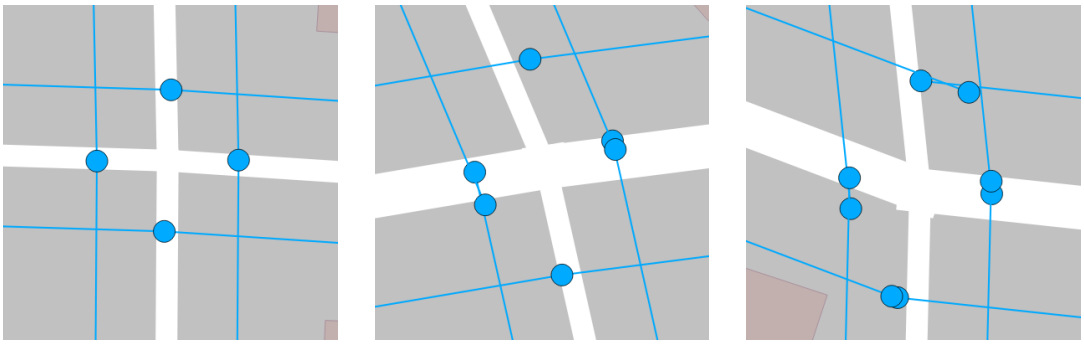
9.1.1 General Route Graph Improvements

One of the main advantages of a route graph created with Map Content Transformations is that it can utilize information about sidewalks, pedestrian crossings and similar information that is hidden in the OpenStreetMap data. The transformed map is therefore a better fit for a pedestrian navigation system, especially for disabled users, for whom such information can be important in route planning. But even for non-disabled users it might be helpful if a navigation system knows that further down the road a sidewalk might only exist on one side, and thus can guide the user on the correct side from the start.

The transformations that are needed to create such a route graph are at first glance not that much different from those created for the map feedback in Section 8.1, with polylines instead of polygons as sidewalks. However, there is one important distinction: Route graphs require topological consistency. For a presentation to the user it is not important, whether two polygons are connected or not, and therefore the transformations used for this purpose do not take topological consistency into account (see Figure 4.2 on page 57). For a route graph there must be a connection between all nodes in order to find the best path to the desired destination. This is in contrast to the required design of Map Content Transformations, whereby specific situations in the map are identified independently of each other, and construction rules are applied to the results separately. Therefore, a set of objects created by one rule is connected to objects created by other rules only in very specific circumstances, namely if both rules use references to the same objects. If the desired topology cannot be expressed in terms of already existing objects, specific topological rules are required.

9.1.2 Individualization of the Route Graph

In addition to those general improvements, which are similar across many user groups, the route graph can be used to express the specific needs and preferences of the user. The requirement to take those needs and preferences into account has been widely recognized, and many purpose-built routing algorithms take into account this sort of information. However, in many cases the routing algorithm itself is optimized to observe specific criteria, for example the slope in routing for wheelchair users [72]. This approach, even if customizable, still presupposes that the developer of



(a) The intersection node is in the middle of two street ways, resulting in continuous parallels.

(b) One street is represented by two distinct ways. The parallels to this way are disconnected.

(c) The intersection is formed by four distinct ways. All parallels are disconnected.

Figure 9.1: Representation of Intersections in OpenStreetMap. Depending on the representation of an intersection, the result of creating parallels differs.

the algorithm knows something about those criteria and allows for their inclusion and customization in the algorithm.

By employing Map Content Transformations, it becomes possible to use any shortest path algorithm based on a specific data format and map the criteria onto that data format. Any criteria that the author of the algorithm does not anticipate can later be transformed into that data format by the transformation.

The current implementation of the navigation system uses a simple shortest path algorithm working on edge weights. Those edge weights can be influenced by the tags of the ways. If no specific tag is given, the distance between the vertices is calculated. This calculated distance can be influenced with the tag with the key `distance_multiplier`. The calculated distance is multiplied with the value of the tag. Alternatively, the weight can be given directly with the key `weight`. It is thus upon the writer of the Map Content Transformations to ensure that the route graph reflects the needs and preferences of the user.

9.1.3 Creation of an Individualized Route Graph

In the following sections, an example of the creation of a route graph with Map Content Transformations will be given. The example includes both the general improvements and an individualization by edge weights.

Rule 9.1 shows a rule that copies all highways in the original map to the

```
<rule order="70" source="original" permutations="false">
<identify>
  <way id="ftw">
    <tag k="highway" v="*"/>
  </way>
</identify>
<construct>
  <way is="ftw" as="reference"/>
</construct>
</rule>
```

Rule 9.1: Copying Ways while Preserving Topological Consistency. By constructing the ways as references, nodes that are shared in the original map are also shared in the newly created map.

destination map. As all the ways in the construction part of the first rule are used as references, the topological consistency of the original map is retained, i.e. intersections are represented by a single node used by both ways. The reason for the creation of these copies is that it becomes easier to split all those ways at intersections, as shown in Rule 9.2. Disconnecting the ways at intersections creates a representation of intersections that is consistent throughout the map. In OpenStreetMap, intersections that look completely equivalent when rendered, can be represented differently in the data. With some intersections, the intersection node is the start or the beginning of the ways that build the intersection. With others, the intersection node is in the middle of the ways. When creating parallels to those ways for sidewalks, the result at the intersection is quite different (Figure 9.1). One way to counter this problem is to first use the element `<disconnect>` at each intersection, thus ensuring that all intersections are represented in the same manner, i.e. such as in Figure 9.1c, making them easier to deal with in later transformations.

After the streets have been copied, the sidewalks can be constructed. Rule 9.3, which is used as an example here, creates sidewalks on the right hand side of streets that are not explicitly tagged as having only left sidewalks. The relation that is being constructed makes it easier for later rules to associate the sidewalk with the original street. The tag of the sidewalk is added, because the system considers only ways that are explicitly tagged with the key `navigation` as parts of the route graph. This convention allows the creation of geometry that acts as a temporary help during computation, but does not influence the resulting route graph.

As the construction of the sidewalks deals with each way separately, the sidewalks are not connected. They just end at each intersection, as shown in Figure 9.2a. In order to create a reasonable route graph, the sidewalks

```
<rule order="80" permutations="false" source="transformed">
<identify>
  <way id="ftw" ignore_order="true">
    <tag k="highway" v="*"/>
    <node id="thenode" ignore_order="true"/>
  </way>
</identify>
<identify>
  <way id="hw" ignore_order="true">
    <tag k="highway" v="*"/>
    <node is="thenode" ignore_order="true"/>
  </way>
</identify>
<construct>
  <disconnect way="ftw" at="thenode"/>
  <disconnect way="hw" at="thenode"/>
</construct>
</rule>
```

Rule 9.2: Splitting Ways at Intersections. By splitting all ways, a unified representation for intersections is created, simplifying their later handling.

need to be connected at the intersections. Rule 9.4 shows the identification of the sidewalks that belong to an intersection. Those sidewalks have in common that they are connected by a relation to streets that share a node that represents the intersection. The listing has been abridged, as the full version requires three more similar relations with ids `"swrLX"`, where X is between 2 and 4, sidewalks on the left are identified, and the node `"cn"` is referenced with the `is` attribute. Likewise, three more similar relations for the sidewalks on the right are needed. These do not explicitly reference the intersection node, because they reference the way that was identified together with the sidewalks on the left, and the connection is therefore implicitly given.

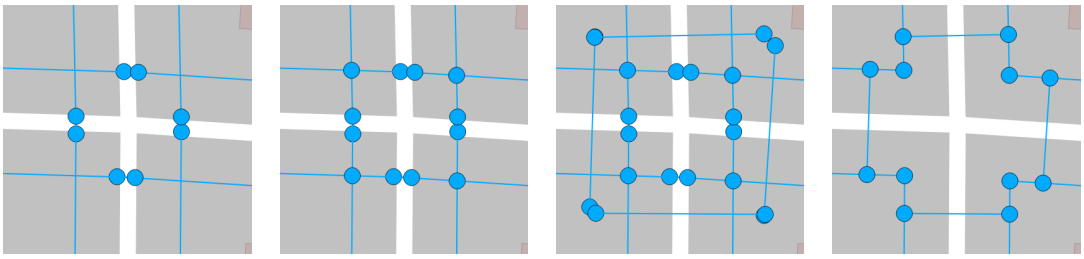
These identified sidewalks can then be connected in a reasonable way. A shortcut construction rule can be used for this (Rule 9.5). The resulting connected version of the route graph is shown in Figure 9.2b.

Connecting the sidewalks is a direct representation of the situation at the intersection, reflecting the connection that exists between the real sidewalks. However, for a route graph, this is not enough, as people can cross a street to the sidewalk on the opposite side. This possibility is not explicitly represented in the OpenStreetMap data, but can still be considered during the construction. Rule 9.6 shows how these implicit crossings are created: A perpendicular to the identified street is created, which is located near

```
<rule order="100" permutations="false" source="transformed">
  <identify>
    <way id="tertiary_way" ignore_order="true">
      <or>
        <tag k="highway" v="tertiary"/>
        <tag k="highway" v="secondary"/>
        <tag k="highway" v="primary"/>
      </or>
      <not>
        <tag k="sidewalk" v="left"/>
      </not>
    </way>
  </identify>
  <construct>
    <relation>
      <parallel role="sidewalkr" on="tertiary_way" distance="10.0"
       from="first" to="last" lengthen_by="1">
        <tag k="navigation" v="sidewalk"/>
      </parallel>
      <way role="mainway" is="tertiary_way" as="reference"/>
      <tag k="connection" v="sidewalk"/>
    </relation>
  </construct>
</rule>
```

Rule 9.3: Constructing Sidewalks. As an example, sidewalks are constructed on the right side of all larger motorized ways that are not explicitly tagged for having a sidewalk on the left side only. If the width of the street was tagged, the distance could be computed from the tagged width, here it is simply assumed.

the identified intersection node. As it is not determined whether this intersection node is the first or the last of the nodes of a street, there need to be two versions of this construction, both enclosed by a `<try>` element. Again, the second version has been replaced by an ellipsis, but it only differs in the offset of the for loop in line 8 and the reference to that offset node in line 11, both of which become positive. As the geometric construction will inevitably throw an exception for the version that does not apply, only the correct version will be executed and thus construct the crossing. For a person, who is not completely comfortable with crossing a street without the presence of a pedestrian crossing, a distance multiplier is added to the implicit crossing. As a result, the route planning algorithm will prefer an explicit crossing, such as pedestrian traffic lights or a zebra crossing, if it requires only a small detour. Three appropriately adjusted versions of the



(a) Parallels to the road represent sidewalks.

(b) Sidewalks are connected.

(c) Creation of implicit crossings.

(d) Complete intersection representation after cropping.

Figure 9.2: Creation of a Route Graph at an Intersection. The sidewalks are created as parallels to the road (a), then connected (b). As the pedestrian is able to cross streets, crossing segments are added perpendicular to the road (c) and connected to the sidewalk. Protruding parts are cropped, resulting in a connected graph representing sidewalks and crossings (d).

```
<rule order="200" source="transformed" permutations="false">
<identify>
 <relation id="swrL1" ignore_order="true">
  <way id="tertiary_way" role="mainway" ignore_order="true">
    <node id="cn" ignore_order="true"/>
  </identify>
  <way id="swl1" role="sidewalkl" ignore_order="true"/>
 </relation>
</identify>
[...]
<identify>
 <relation id="swrR1" ignore_order="true">
    <way is="tertiary_way" ignore_order="true"/>
    <way id="swr1" role="sidewalkr" ignore_order="true"/>
 </relation>
</identify>
[...]
```

Rule 9.4: Identification of Sidewalks. Sidewalks constructed in earlier rules are identified on the left and right side of one way. For a four-way intersection, three similar additional rules for each side are required.

entire relation are needed for the other ways of the intersection, resulting in the graph depicted in Figure 9.2c.

The implicit crossings are not yet connected to the sidewalks. How this connection is implemented is shown in Rule 9.7. Additionally, the crossing is cropped, i.e. the protruding part are removed. Figure 9.2d shows the

```
<connect_sidewalks on="cn" role="connection"
 along1="tertiary_way" along2="tw2" along3="tw3" along4="tw4"
 relationA1="swrL1" relationA2="swrL2"
 relationA3="swrL3" relationA4="swrL4"
 relationB1="swrR1"  relationB2="swrR2"
 relationB3="swrR3" relationB4="swrR4"
 roleA1="sidewalkl" roleA2="sidewalkl"
 roleA3="sidewalkl" roleA4="sidewalkl"
 roleB1="sidewalkr" roleB2="sidewalkr"
 roleB3="sidewalkr" roleB4="sidewalkr">
  <tag k="navigation" v="connection"/>
</connect_sidewalks>
```

Rule 9.5: Connecting Sidewalks. The sidewalks identified in Rule 9.4 are connected in order to create a topologically reasonable representation.

final graph at the intersection. In addition to the crossings, the sidewalks have been appropriately cropped here as well.

The rules presented in this chapter are not enough to create a fully fledged route graph for an area in OpenStreetMap, but it explains all the principles that are needed. Additional rules are required to handle other situations, such as footpaths and how they are connected to sidewalks, explicit crossings and roundabouts.

Figure 9.3a shows a route calculated based on a transformation that includes all the required rules. The transformation used for this route does not include the distance multiplier from line 7 in Rule 9.6, i.e. it can be used for people who prefer the shortest path and are comfortable with crossing a street without a zebra crossing or a pedestrian light. For other users, who prefer using explicit crossings, the version including the distance multiplier as given in Rule 9.6 can be used. Figure 9.3b shows the resulting route, which is slightly longer, but guides the user across a pedestrian traffic light to cross the street.

9.2 Destination Data

Car navigation systems offer route planning to addresses and points of interest. These are specifically stored as destinations in the underlying data. OpenStreetMap does not offer such data directly. The only information that can be used by a navigation system to provide destinations are the tags of the map objects.[1] As not all tags are useful as destination designators, there needs to be a convention by which the navigation system

[1] Provided that no further, external sources for destinations are used

```
<relation>
  <tag k="connection" v="implicit"/>
  <way as="reference" is="swr1" role="sidewalk_right"/>
  <way as="reference" is="swl1" role="sidewalk_left"/>
  <way role="crossing_way">
    <tag k="navigation" v="crossing"/>
    <tag k="distance_multiplier" v="3.0"/>
    <try>
      <for type="node" id="curNode" on="tertiary_way"
       from="cn" to="cn" offset="-1">
        <gtl>
          <![CDATA[ POINT t1, t2, r1, r2;
           [t1, t2] = [curNode, curNode-1] *
                      (11/DISTANCE([curNode, curNode-1]));
           [r1, r2] = [t2, t1] L 10.5;
           [r1, r2] = [r2, r1] * 2;
          ]]>
        </gtl>
        <node is="r1" type="copy">
          <tag k="navigation" v="crossingnode"/>
        </node>
        <node is="r2" type="copy">
          <tag k="navigation" v="crossingnode"/>
        </node>
      </for>
    </try>
    [...]
  </way>
</relation>
```

Rule 9.6: Creating Implicit Crossings. The crossings are created near the intersection and perpendicular to the street that belongs to the sidewalks identified in Rule 9.4. Here, just one crossing is created as an example.

decides what to consider as a possible destination. This “destination by convention” approach is suitable for both untransformed and transformed OpenStreetMap data.

However, there is another problem with using OpenStreetMap data: Many tags that are interesting as destinations are associated with map entities that are not directly connected to the street network. As an example, the addresses of buildings are stored in tags of the building outline. There is no representation in the data of the connection between a building and the street it belongs to; the only association between the two is the name of the street and its counterpart in the building address. Therefore, a standard route finding algorithm that only uses OpenStreetMap’s street network as

```
<rule order="230" source="transformed" permutations="false">
<identify>
  <relation>
    <tag k="connection" v="implicit"/>
    <way id="swr" role="sidewalk_right" ignore_order="true"/>
    <way id="swl" role="sidewalk_left" ignore_order="true"/>
    <way id="crossing" role="crossing_way"/>
  </relation>
</identify>
<construct>
  <connect way1="crossing" way2="swr" from1="first" to1="last"
   from2="first" to2="last">
    <tag k="navigation" v="connection"/>
  </connect>
  <connect way1="crossing" way2="swl" from1="first" to1="last"
   from2="first" to2="last">
    <tag k="navigation" v="connection"/>
  </connect>
</construct>
<construct>
  <crop way="crossing"/>
</construct>
</identify>
```

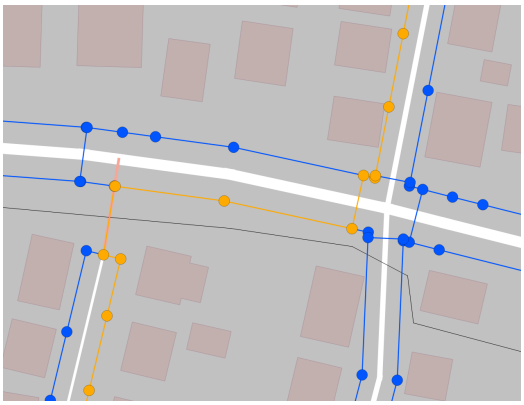
Rule 9.7: Connecting the Implicit Crossing. In order to maintain topological consistency, the crossings created in Rule 9.6 are connected with the sidewalks.

a route graph cannot guide a user to a specific house.

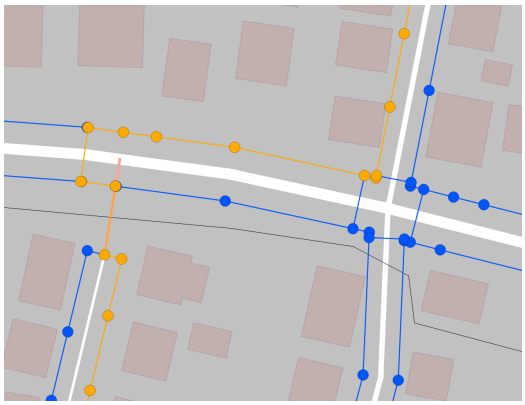
Map Content Transformations make it possible to create a route graph that includes destinations which are not part of the original street network, by either connecting the destination locations to the graph, or projecting them onto the graph. Looking at the example of building addresses, these addresses can be used as a destination, if they are mapped to the nearest sidewalk of the street that has the same name as the street part of the building's address.

The first step towards such a mapping is a simple rule, not shown here, that creates a node in the transformed map containing the address of the building in the original map. This node can be at the position of the marked entrance to the building, if available, or at some other position within the building.

Rule 9.8 shows how this node can then be mapped onto a route graph created with the rules from Section 9.1. The rule identifies nodes that are near a street and have the street's name as part of its address. While



(a) The shortest way includes crossing a street at an intersection without pedestrian crossings.



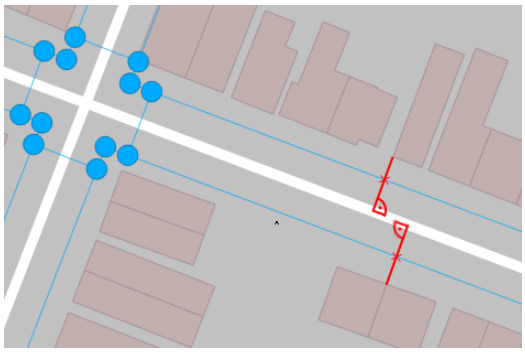
(b) Setting appropriate multipliers will make the route planning select a way that is a little longer, but uses a pedestrian crossing.

Figure 9.3: Comparison of two routes with the same start and destination. Both routes were calculated by a simple A*-Algorithm. The route graph for (a) is based on nothing but distance between nodes. The route graph for (b) was created by a transformation with a penalty factor on crossings that are not explicitly represented in OpenStreetMap.

iterating over the street, a perpendicular to the street's current section through the identified house node is created and intersected with the left sidewalk of the street. As the perpendicular through the node is only created, if the perpendicular intersects the segment, and otherwise an exception is thrown, only one new node on the sidewalk is created for each house node. Those are automatically inserted after the current node on the sidewalk, due to the `insert` attribute. Again, a similar rule is required for the sidewalk to the right of the street.

The effects of applying these rules can be seen in Figure 9.4. The route graph shown in Figure 9.4a was constructed with the rules from Section 9.1. After the application of the rule, every house is represented by a node on the route graph (Figure 9.4b) and can be selected as a destination by choosing the address of that house.

House numbers mapped onto sidewalks and similar points of interest assume that the destination is one node in the route graph. However, in reality sometimes the desired destination cannot be represented as such a single node, especially in route graphs that are as detailed as those used here. If, for example, the user simply wants to reach a specific intersection, this intersection is represented by several nodes in the route graph (see Figure 9.2 on page 116). Therefore, the it was chosen to implement meta-nodes in the developed navigation system. These are sets of nodes



(a) The sidewalks have already been constructed. The (manually added) red lines demonstrate the construction of the house number nodes on the sidewalks.

(b) All houses are now represented with nodes on the sidewalk and can be selected as destinations.

Figure 9.4: Construction of a Route Graph with House Numbers. Perpendiculars to the street that go through a node on the building's outline (preferrably the tagged entrance) are intersected with the sidewalk (a). The intersection points are added as new nodes and thus become possible destinations in the route graph (b).

connected through a relation that can have a specific name. Meta-nodes can serve as a destination, and the user will always be guided to the nearest node of the meta-node. Meta-nodes can easily be created by the transformation that creates the route graph. As an example, to create meta nodes for intersections, one can simply enclose the connection nodes created by Rule 9.5 on page 117 with a relation containing a tag with the key `"navigation"` and the value `"metanode"`. The name can then be created with the names of the streets, assuming that those have been identified beforehand.

The routing algorithm will automatically look for the nearest node of such a meta-node, and stop as soon as this destination is reached, thus preventing a possible unnecessary guidance across a street, which would be possible if just any of the intersection's nodes had been selected.

9.3 Input Modalities

In order to select the desired destination, a user must somehow be able to enter that destination into the system. Selecting from a list of available locations is theoretically possible, and might even be practicable, if the number of selectable destinations is relatively small, but in a real-world

application with many destinations, such a list would quickly grow too large.

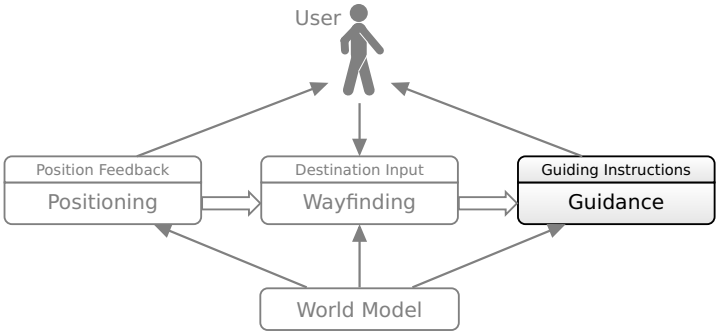
The implementation of the navigation system therefore uses a mixture of entering and selecting a destination, similar to those used in most commercial navigation systems. A name, or a part of a name can be entered. After that, a list of all destinations that partially match the entered string is created, and can be scrolled through. In that list, a destination can be selected. If the selected name is the unique name for a destination, the route guidance to that destination is started. If the name is part of a an address, another list with possible options, e.g. house numbers, is created. Upon selection of the exact destination, the route guidance is started.

9.4 Discussion

In this chapter it was shown that the specifically developed Map Content Transformations can be used to individualize route planning. This individualization can affect both the destination input by the user and the wayfinding algorithm itself. The examples given in this chapter are in no way a complete representation of the possibilities of Map Content Transformations. They can also be used to create route graphs that consider the slope of a street or the sidewalk's surface for wheelchair users, try to avoid busy roads, include any kind of map object as destinations, or contain a preference for walks through a park, to name just a few. With additional, not yet existing OpenStreetMap tags, further refinements of the individualized route graphs are possible, which could, for example, help to avoid particularly confusing areas or sidewalks that are too narrow, or even include a preference for a particularly sheltered path on a rainy day.

```
<rule order="190" permutations="false" source="transformed">
<identify>
  <relation ignore_order="true">
    <way role="sidewalkl" id="sidewalk" ignore_order="true"/>
    <way role="mainway" id="mainway" ignore_order="true">
      <tag k="name" v="*" id="streetname"/>
    </way>
  </relation>
</identify>
<identify>
  <node near="sidewalk" id="housenode" distance="40"
   ignore_order="true">
    <tag k="addr:street" v="{streetname}"/>
    <tag k="addr:housenumber" v="*" id="housenumber"/>
  </node>
</identify>
<construct>
  <way is="sidewalk" as="reference">
    <for type="node" id="cnode" on="mainway"
     from="first" to="last-1" offset="1">
      <for type="node" id="swnode" on="sidewalk"
       from="first" to="last-1" offset="1" insert="after">
        <try>
          <gtl>
            <![CDATA[ POINT t1, t2, p1, p2;
            [t1, t2] = [cnode, cnode+1] t [cnode, housenode];
            [t1, t2] = [t2, housenode] x [swnode, swnode+1];
            ]]>
          </gtl>
          <node is="t2" as="copy">
            <tag k="navigation" v="goalnode"/>
            <tag k="name" v="{streetname}"/>
            <tag k="housenumber" v="{housenumber}"/>
          </node>
        </try>
      </for>
    </for>
  </way>
</construct>
</rule>
```

Rule 9.8: Inserting House Addresses on Sidewalks. Perpendiculars to the street through a node of the house are intersected with the sidewalks. The intersection point receives the house address as a possible destination.



10 Route Guidance

After the route has been calculated, the user needs to be guided to the selected destination. How exactly this is done again depends on the specific needs of the user. These needs influence both the modality of the guidance and the concrete implementation of how the information is transmitted by the chosen modality. In this context, modality refers to the channel through which the information is conveyed. This can be the sensory channel, i.e. visual, haptic, auditory, etc., or a reasonably distinct subdivision of the sensory channel, so that purely auditory cues can be differentiated from spoken text output.

This chapter discusses two exemplary implementations for two distinct sensory channels. While the tactile guidance is only influenced by the route graph itself, it will further be shown how text output for guidance can be individually generated through Map Content Transformations.

Parts of this chapter have previously been published in:

Bernhard Schmitz. The ViibraCane – a white cane for tactile navigation guidance. In *Proceedings of the California State University, Northridge Center on Disabilities' 25th Annual International Technology and Persons with Disabilities Conference (CSUN 2010)*, 2010. Extended Abstract.

Bernhard Schmitz, Susanne Becker, André Blessing, and Matthias Großmann. Acquisition and presentation of diverse spatial context data for blind navigation. In *12th IEEE International Conference on Mobile Data Management (MDM 2011)*, pages 276–284, 2011. DOI:10.1109/MDM.2011.66

Bernhard Schmitz and Thomas Ertl. Individualized route planning and guidance based on map content transformations. In *Proceedings of the 14th International Conference on Computers Helping People with Special Needs (ICCHP 2014)*, pages 120–127, 2014. DOI:10.1007/978-3-319-08599-9_19

Figure 10.1: The ViibraCane. In addition to direction guidance by vibration, the cane can be used as an input device to the navigation system.

10.1 The ViibraCane

As was discussed in Section 2.3, there has been much research regarding different modalities for guiding blind users. All papers referenced there have unequivocally shown that it is possible to guide blind users to a destination with various forms of haptic and auditory cues. While some concrete implementations fared slightly better than others, there has been no clear determination that a specific implementation is generally preferable to other ones. In particular, it was shown by Marston et al. [70] that a simple, binary indication of being on-track or off-track (whether haptic or auditory) suffices for accurate route following and therefore more complex devices, such as haptic belts, like the one presented by Heuten et al. [41], are not strictly necessary in order to guide a user to a destination.

For the navigation system presented in this thesis, a tactile device called ViibraCane was developed that can be used for route guidance. The ViibraCane consists of a Wii remote mounted on a cane (see Figure 10.1). The Wii remote is normally used as the standard input device for the Nintendo Wii console, and features, in addition to seven buttons and an analog directional pad, a loudspeaker, an inertial sensor, and an inbuilt vibration device. The design of the device was influenced by three considerations: Firstly, it was mentioned by several blind people, that a cane is standard equipment for a blind pedestrian and they would prefer not to be encumbered with another device. Secondly, the Wii remote is a widely available consumer device, and therefore much cheaper than specifically built hardware. The third consideration was the already proven sufficiency of the single-actuator approach. Additionally, the directional pad can be used to control the menus of the system, offering quick access to almost all functionality.

10.1.1 Guiding Functionality of the ViibraCane

As the vibration of the cane is meant to be the primary means of guidance and the user will walk in the correct direction most of the time, it is reasonable to choose non-vibration as the signal of correct heading and correspondingly vibration as the signal that the user is currently off-course. [70] have shown that a preference for this configuration (albeit small) exists. As has been mentioned, these two bits of information are enough to guide a user to a destination. However, giving the user more information will ease the use of the system. This information could include the distance to the next waypoint, the direction in which the user needs to turn, or how much he needs to turn. Not all of this can be included in the vibrating pattern, especially due to the technical limitations of the Wii remote. Therefore, some decision had to be made regarding which information to include and which to disregard.

The distance to the next waypoint does not need to be conveyed permanently and is therefore not included in the information. In contrast to that, knowing the direction, in which one needs to turn in order to face the next waypoint, is quite helpful: Not including this information could result in the user performing a turn of almost 360° in order to find the "no vibration" signal. Therefore, it was decided to encode the direction in which the user needs to turn with the length of the vibration pulse: A longer pulse indicates that the user needs to turn left and a shorter one indicates the need for a right turn.

There are two different situations in which the cane will switch from non-vibrating to vibrating status. Firstly, the user might just have veered slightly off the desired path. In this case, only a small correction of the direction is necessary. Secondly, the user might have reached a waypoint, meaning that he often needs to change his direction completely. It is desirable that the user can differentiate between those two situations. Therefore the vibration patterns differentiate between small and large correction angles: A strong vibration indicates that a large turn is necessary, whereas a lighter vibration signals that a small turn is sufficient. As the Wii remote natively allows only two states (vibrating and non-vibrating), the lighter vibration is accomplished by turning the vibration device on and off in short succession. So all in all there are five different states of vibration that are used to give directional information. One additional signal, a very long vibration is used to indicate that the user has arrived at the desired destination. It is possible to distinguish this signal from the other ones due to its length. However, for the same reason it is not well suited to indicate a direction, as it could not change quickly enough. These signals are enough to successfully guide a user to a desired destination.

10.2 Text Output

Text output, whether spoken with a Text-To-Speech engine or displayed on a Braille display, is a widely used way of conveying navigational information. Earlier car navigation systems used simple descriptions of the direction (e.g. “turn right”), while newer versions include information about the environment, such as street names. These two types of information can be described with a simple classification: Static text and dynamic text. Static text is stored in the map. For dynamic text, there are two different possibilities. It can be created from information that is stored in the map, i.e. the layout of the street, or be created on-the-fly, depending on the direction or position of the user. In car navigation systems, where a car is normally aligned with the street’s direction, the former is an option that can be considered. However, for a pedestrian navigation system, an alignment of the user to the street or the route graph cannot be presupposed, and therefore an on-the-fly creation is the preferred solution.

The normal approach used nowadays is to combine those two kinds of texts, e.g. the sentence “Turn right into St. Denys Road” contains a directional hint that can be created by any of the two mentioned methods, and the name of a street that is statically stored in the map. The feedback about the environment and how it ties together with the dynamic text is normally hardcoded into the navigation system, and can therefore not be individualized according to the user requirements.

The system described in this thesis uses text output, generated with a text-to-speech engine as well. In contrast to other systems, this output can be individualized, again be employing Map Content Transformations. The following section describes how the data model can influence text output, how this static text is combined with dynamic text, and how Map Content Transformations can be used to create an appropriate data model.

As has already been mentioned, the static text is mainly used to refer to the environment. These descriptions of the environment can have different purposes:

- For blind people, descriptions are required, that convey information that could be seen by a sighted user. As an example, if a blind user has to cross a street at a traffic light, the information whether the crossing includes a pedestrian refuge can be of vital importance, as otherwise he has no way of telling whether he has arrived at the opposite sidewalk or just at a pedestrian refuge in the middle of the street, when registering the curbstone.
- For sighted people, some descriptions of the environment can help the navigation process, by allowing a comparison of the description with the existing world. As an example, the description “turn right

and take the stairs” is more helpful than a simple “turn right”, and might prevent navigational errors.
- The third purpose of the descriptions are that they can convey additional information about the environment and thus help to enhance the knowledge of the user. If a blind user crosses a street, the simple mentioning of the street’s name can help to broaden the mental image of the environment and, for example, be a point of reference in conversations with other people.

Static text for specific situations can be stored in the route graph and therefore be created by Map Content Transformations. However, such an preprocessing step is not possible for the dynamic text, i.e. anything that is directly related to the direction or the position of the user. There are two ways to circumvent this problem:

- The first possibility is to create text that is relative to the direction of the route graph. If it is assumed that the user will follow the directions of the system, it can be predetermined what will be to his left or right depending on the path through the (directed) graph.
- The other possibility is to create static text with placeholders that are by convention replaced with dynamically created text during runtime.

The approach described in this thesis allows for both possibilities, as well as their combination. Text output is being created by reading descriptive texts stored in the route graph in conjunction with directional hints computed on the fly, according to specific conventions, as explained in the following section. The conventions described therein should not be considered as the only way to handle this, but the serve as a demonstration of the usefulness of in route guidance.

10.3 Conventions for Text Output

For OpenStreetMap data, the obvious choice is to store the descriptive texts in tags as values to specific keys recognized by the navigation system. One approach would be to store the texts in the nodes. However, the problem with this approach is that it is completely direction agnostic, i.e. a text that is stored in a node cannot differ depending on where the user should go after he reaches that node. Another possibility is storing the descriptive text in the ways. Unfortunately, as was explained in Section 9.1, OpenStreetMap’s data format does not match exactly on a graph structure.

As a consequence, when storing texts in a way, it is not clear when the text should be output to the user.

Because of these difficulties, the current implementation leaves as much possibilities to the writer of the transformation as possible. It allows for direction agnostic texts in nodes as well as texts in ways. Text that is stored in ways is used at every node contained by the way. This can be used, if there is a need to repeatedly output the same text at each node. If such a repeated output is not desired, the writer of the transformation must ensure that the result is sensible, e.g. by making sure that each way consists of only two nodes. Additionally, texts can be stored in relations that connect a way with a specific node, thus avoiding the problem of the ambiguity when storing the text directly in a way. Text stored in such a relation is treated equally to text stored directly in the way, but is only used at the node that is connected to the way by the relation. For ways and relations, there need to be different texts when a user starts at the beginning of the edge and when he arrives at its end. Additionally, both of these should also be available as reversed versions, as OpenStreetMap's ways are inherently directed (see Section 4.1.1 on page 45), and the text that is output should differ according to the direction of the user.

Taken together, the implementation outputs all texts to a user once he arrives at a node as follows:

1. Output value of key `"arrivetext"` of the way that that was active before the user reached the node or of the relation that connects that way with the current node.
2. Output value of key `"arrivetext"` of node.
3. Output dynamically created directional hint, depending on current direction of user and next node.
4. Output value of key `"starttext"` of node.
5. Output value of key `"starttext"` of way connecting current node with next node, or of the relation that connects that way with current node.

If the direction of the user in the route graph is the opposite of the inherent direction of the OpenStreetMap way, the value of the key `"starttext_reverse"` or `"arrivetext_reverse"` is read out respectively for ways and relations.

With these conventions in mind, it becomes easy to adapt the route graph created in Section 9.1, so that the result of the transformation include individualized text output, as described in the following section.

10.4 Map Content Transformations for Route Guidance

As the texts that should be stored in the route graph are highly dependent upon the user's situation, only a few representative examples for the possibilities will be given in this section.

Rule 9.6 on page 118 showed the creation of creates implicit crossings near intersections, Rule 9.7 on page 119 connects these crossings to the sidewalk. This crossing can be enhanced with some simple additions. When the tags given in Rule 10.1 are added to the way created in Line 5 of Rule 9.6, the appropriate text is output as soon as the user has to cross the street. Similarly, descriptive texts can be added for other pedestrian crossings, describing the situation according to the tags present in the OpenStreetMap data, such as zebra crossings, pedestrian traffic lights, the existence of pedestrian refuges, etc. The crossing is represented by a way consisting of only two nodes. Therefore the descriptive texts can be added directly as tags of the way. However, when the user has finished crossing the street, and a special text should be output referencing the sidewalk that should be used to continue the trip, this approach cannot be used, as the sidewalks usually span across many nodes. It is therefore necessary to create a relation that contains both the node connecting the sidewalk with the crossing, and the sidewalk itself. Rule 9.7 on page 119, which

```
<tag k="starttext" v="You will then have to cross {name1}
 without a pedestrian crossing"/>
<tag k="starttext_reverse" v="You will then have to cross {name1}
 without a pedestrian crossing"/>
```

Rule 10.1: Guidance at an Implicit Pedestrian Crossing. The tags, which should be added to the creation of the implicit crossing in Rule 9.6, include the street's name and the information that there is no pedestrian crossing.

```
<identify>
  <relation  ignore_order="true">
    <way is="swr" ignore_order="true"/>
    <way role="mainway" ignore_order="true">
      <tag k="name" v="*" id="streetname"/>
    </way>
  </relation>
</identify>
```

Rule 10.2: Identification to Make a Street Name Available. This identification is intended to be added to the Rule 9.7.

```
<relation>
  <connect way1="crossing" way2="swr" from1="first" to1="last"
   from2="first" to2="last">
    <tag k="navigation" v="connection"/>
  </connect>
  <way is="swr" as="reference"/>
  <tag k="starttext" v="Then follow sidewalk
   to the right of {streetname}"/>
  <tag k="starttext_reverse" v="Then follow the sidewalk
   to the left of {streetname}"/>
</relation>
```

Rule 10.3: Creation of Descriptive Texts for Sidewalks. Replacement for the creation of the connection in Rule 9.7, creating a text that describes the sidewalk where it is connected to an implicit crossing.

created the connecting nodes, does not reference the name of the street. Adding a second identification, as shown in Rule 10.2, makes the name of the street available. Rule 10.3 shows the relation that needs to replace the creation of the connection in Line 11 of Rule 9.7, in order to create a sensible descriptive text.

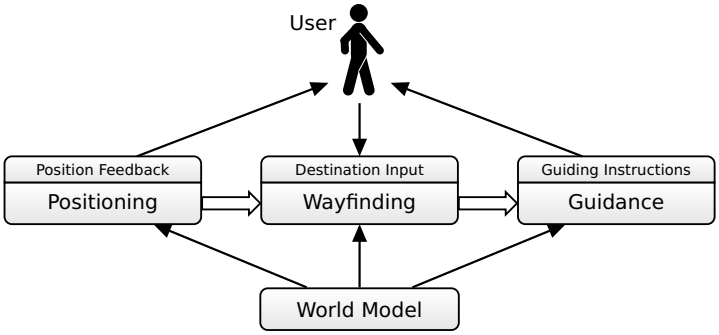
Thus, if a user is to be guided across a street without a pedestrian crossing, the system will output sentences like “Turn right. You will then have to cross St. Deny’s road without a pedestrian crossing.” upon arriving at the position where the road needs to be crossed. Arriving at the other side of the road, the output might be “Turn left. Then follow the sidewalk to the right of St. Deny’s Road.”

10.5 Discussion

The two different approaches to guidance covered in this chapter are not to be seen as alternatives, but as complementary to each other. The speech output, as currently implemented, only gives feedback at nodes in the route graph. This might not be enough to guide a blind user, who might veer off-track between those nodes. This can be corrected by other means, such as a haptic feedback device like the ViibraCane. On the other hand, such simplistic feedback alone is severely limiting in what it can accomplish: There is no information about the environment included in this purely directional haptic output, which could provide helpful contextual information for the navigation, or simply help to form a mental idea about the environment. This contextual information is provided by the textual output. As has been shown, this textual output can be individualized to a

great degree by Map Content Transformations. The examples shown in this chapter are rather simple, providing information that can be considered as generally helpful.

However a larger degree of individualization is certainly possible. The two individual representatives of subgroups that were mentioned in Section 1.2.1 on page 4 can serve as an example here. The first individual indicated preference for concentrating on the navigational task and not wanting to give attention to the surroundings, whereas the second wanted to learn as much as possible about the environment while navigating. By individualizing the textual feedback stored in the route graph, it becomes possible that the former receives only information relevant to the navigation. Another route graph, created for the latter, can even include additional, purely informative nodes containing information about nearby attractions, such as shops or cafés.



11 Conclusions

This thesis introduced several fundamental improvements in the field of navigation for special user groups. There are two aspects to these improvements: There are technical results, as it was the philosophy of this thesis to provide concrete implementations of every concept. The implementations allowed testing and verification of those concepts. On a more abstract level, the implementations provided the basis for more general insights.

11.1 Technical Results

The most obvious technical result of this thesis is a research prototype of a complete navigation system for special user groups, especially blind people. The system covers all stages of navigation and provides adapted solutions for each stage. Concretely, the system provides positioning, adapted position feedback through the display of maps on a touchscreen coupled with acoustic feedback, wayfinding with destination input, and guidance with guiding instructions through a vibrating cane and speech output. Additionally, the system can be used as a map exploration tool, especially if coupled with specific hardware, such as a tactile graphics display. A blind colleague who used that system said that she was able to learn new information about buildings and street layouts even in areas where she had lived or worked for many years.

Other important technical results are the two ways of creating adapted world models introduced by this thesis. The first is based on the idea of aggregation of varied information sources and uses the Nexus platform, developed in a collaborating project. The second is a completely new approach in which common map data is transformed through a set of individual rules resulting in individualized maps. This approach was implemented from

scratch in the form of Map Content Transformations based on OpenStreetMap data. By integrating Map Content Transformations into the navigation system all stages of navigation can profit from maps created in this way, leading to further, non-technical results.

11.2 General Insights

The complete navigation system with integrated Map Content Transformations provides a platform that helped answering many fundamental research questions (see Section 1.3):

Large-Scale Adaptation. Since OpenStreetMap is used, map data for the entire world is available. The rules in Map Content Transformations are not tied to specific locations, and therefore the adaptation can take place anywhere.

Adaptation Flexibility. Because Map Content Transformations allow completely free scripting of the adaptations, they can be written to target any level of the user group categorization. Many of the examples provided in this thesis focused on blind people as an example of a requirements group, but it could be shown that adaptations for individual users are possible as well.

Algorithmic Benefit of Adaptation. The thesis demonstrated that all three stages of navigation can profit from Map Content Transformations. Positioning can be improved by using Map Content Transformations to create individualized maps for particle filters (Chapter 6). This is a fitting combination, as blind people need good positioning, but also often intentionally restrict their movement, e.g. by not walking on grass. This restriction can be encoded into the individualization. Wayfinding can benefit from improved route graphs including sidewalks and pedestrian crossings, and be individualized according to the user's abilities and preferences (Section 9.1). Finally, detailed and individualized guidance instructions can be created, leading to a better user experience in the last stage of navigation (Section 10.4).

Integration with User Interaction. It was shown that user interaction with the navigation system can be enhanced by Map Content Transformations. The user can get feedback about the surrounding area from maps that are adapted to his preferences and the mode of display (Chapter 8). User feedback for these adapted maps was very positive, and the test users asked for the availability of the system. Map objects, like houses with their addresses, or points of interests like monuments can be mapped onto the route graph

and thus become selectable as destinations, even if they are not connected to any street network in the original OpenStreetMap data (Section 9.2). Descriptive guiding instructions that are helpful for finding the correct path or learning about the environment can be created; again fully customizable according to user preferences (Section 10.3).

Benefit for Navigation for Blind People. The thesis addressed these challenges in navigation (see Section 1.2.2), either by creating specific solutions, or by demonstrating how Map Content Transformations can be used to address them.

Looking at each challenge, they were met as follows:

Integration of Information Sources. The Nexus platform is specifically designed to unite information from various sources (Chapter 3). OpenStreetMap as a community-based system does not restrict the integration of additional information in any way. With Map Content Transformations it becomes possible to integrate information that might only be relevant to a specific user group in such a way that others are not negatively affected by it.

Street Crossings and Construction Areas. It was demonstrated that it becomes possible to describe street crossings in the static navigational hints stored in the route graph (Section 9.1), in particular if more specialized tags were to be included in the OpenStreetMap data. It was also shown how information about construction areas can be aggregated into a navigation system (Section 3.3.2.3).

Up-Front High Level Information. Several ways of providing up-front information, mainly in the form of maps, were presented in this thesis, utilizing cheap rumble gamepads, touchscreens and a specialized tactile graphics display (Chapter 7). In addition to that it was demonstrated that Map Content Transformations can be used to improve this up-front information, by creating maps that include both additional information and areas that support orientation on the map (Chapter 8).

Sharing Information. Even if the means of sharing information were not treated in this thesis, Map Content Transformations were intentionally based on OpenStreetMap. This guarantees that the entire system is designed from the ground up to accommodate the principles of community-based information sharing.

11.3 Outlook

The possibility of sharing information and adding information to the common world model is certainly a main area of future work. This will not only help to improve navigation, but also enable people with disabilities to fully integrate into the culture of participation that is at the heart of efforts such as OpenStreetMap.

Many of the techniques presented in this thesis enable a very fine-grained navigation. The largest problem that still needs to be overcome is still adequate positioning both in- and outdoors. However, it can be regarded as a more general problem that is not confined to the area of navigation systems for specific user groups, and can therefore be considered a separate area of research.

In many areas of navigation, especially for blind people, computer vision will play an important role, as both the navigation itself, and the addition of information to the common world model can benefit from it. But even then, the work presented here will continue to play an important role, as navigation systems that include computer vision require a world model as well.

Regarding Map Content Transformations, this thesis could only provide a prototype that demonstrates the possibilities and usefulness of the concepts, but is far from exhausting the full capacities of adaptable world models. Due to the flexibility of the approach, other use cases and even completely new areas of application for the rule-based individualization of maps could open up, ensuring the continued relevance of the contributions presented in thesis.

A

Technical Details

XML schema for Map Content Transformations

The following XML schema describes the XML part of the language that was developed for Map Content Transformations. For the sake of brevity and readability of the document, the schema actually defines a superset of the accepted language and thus does not include all restrictions. Notably, the schema allows inclusions of elements through indirections, e.g. of objects with disallowed attributes under logical operators or inclusion of ways in nodes through the <try> element.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="transformation">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="rule" type="ruleType"
        maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="new_map" type="xs:boolean"
      use="required"/>
    </xs:complexType>
  </xs:element>



  <xs:complexType name="ruleType">
```

```
    <xs:sequence>
      <xs:element name="identify" type="identifyType"
      maxOccurs="unbounded"/>
      <xs:element name="construct" type="constructType"
      maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="order" type="xs:integer" use="required"/>
    <xs:attribute name="source" type="xs:string"/>
    <xs:attribute name="permutations" type="xs:boolean"/>
  </xs:complexType>

  <xs:complexType name="identifyType">
    <xs:choice>
      <xs:element name="node" type="identifyNodeType"/>
      <xs:element name="way" type="identifyWayType"/>
      <xs:element name="relation" type="identifyRelationType"/>
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="identifyTagType">
    <xs:attribute name="k" type="xs:string" use="required" />
    <xs:attribute name="v" type="xs:string" use="required" />
    <xs:attribute name="id" type="xs:string" use="optional" />
  </xs:complexType>

  <xs:complexType name="operatorNodeType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="tag" type="identifyTagType"/>
      <xs:element name="not" type="operatorNodeType"/>
      <xs:element name="and" type="operatorNodeType"/>
      <xs:element name="or"  type="operatorNodeType"/>
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="operatorWayType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="tag"  type="identifyTagType"/>
      <xs:element name="node" type="identifyNodeType"/>
      <xs:element name="not"  type="operatorWayType"/>
      <xs:element name="and"  type="operatorWayType"/>
      <xs:element name="or"   type="operatorWayType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="operatorRelationType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="tag"      type="identifyTagType"/>
      <xs:element name="node"     type="identifyNodeType"/>
```

```
      <xs:element name="way"      type="identifyWayType"/>
      <xs:element name="relation" type="identifyRelationType"/>
      <xs:element name="not"      type="operatorRelationType"/>
      <xs:element name="and"      type="operatorRelationType"/>
      <xs:element name="or"       type="operatorRelationType"/>
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="identifyNodeType">
    <xs:complexContent>
      <xs:extension base="operatorNodeType">
        <xs:attribute name="id"           type="xs:string"/>
        <xs:attribute name="is"           type="xs:string"/>
        <xs:attribute name="near"         type="xs:string"/>
        <xs:attribute name="distance"     type="xs:float"/>
        <xs:attribute name="ignore_order" type="xs:boolean"/>
        <xs:attribute name="role"         type="xs:string"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="identifyWayType">
    <xs:complexContent>
      <xs:extension base="operatorWayType">
        <xs:attribute name="id"           type="xs:string"/>
        <xs:attribute name="is"           type="xs:string"/>
        <xs:attribute name="near"         type="xs:string"/>
        <xs:attribute name="distance"     type="xs:float"/>
        <xs:attribute name="ignore_order" type="xs:boolean"/>
        <xs:attribute name="role"         type="xs:string"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

```
  <xs:complexType name="identifyRelationType">
    <xs:complexContent>
      <xs:extension base="operatorRelationType">
        <xs:attribute name="id"           type="xs:string"/>
        <xs:attribute name="is"           type="xs:string"/>
        <xs:attribute name="ignore_order" type="xs:boolean"/>
        <xs:attribute name="role"         type="xs:string"/>
```

```
114         </xs:extension>
          </xs:complexContent>
116     </xs:complexType>

118     <xs:complexType name="constructType">
          <xs:choice minOccurs="0" maxOccurs="unbounded">
120         <xs:element name="node"       type="constructNodeType"/>
            <xs:element name="way"        type="constructWayType"/>
122         <xs:element name="relation"   type="constructRelationType"/>
            <xs:element name="crop"       type="cropType"/>
124         <xs:element name="connect"    type="connectType"/>
            <xs:element name="connect_sidewalks"
126         type="connectSidewalkType"/>
            <xs:element name="disconnect" type="disconnectType"/>
128         <xs:element name="for"        type="forType"/>
            <xs:element name="try"        type="tryType"/>
130         <xs:element name="gtl"        type="xs:string"/>
            <xs:element name="hull_polygon" type="hullPolygonType"/>
132         <xs:element name="intersection_polygon"
            type="intersectionPolygonType"/>
134         <xs:element name="parallel"    type="parallelType"/>
          </xs:choice>
136     </xs:complexType>

138     <xs:complexType name="constructNodeType">
          <xs:choice  minOccurs="0" maxOccurs="unbounded">
140         <xs:element name="tag"  type="constructTagType"/>
            <xs:element name="tags" type="constructTagsType"/>
142       </xs:choice>
          <xs:attribute name="is"   type="xs:string" use="required"/>
144       <xs:attribute name="as"   type="xs:string"/>
          <xs:attribute name="role" type="xs:string"/>
146     </xs:complexType>

148


150
        <xs:complexType name="constructWayType">
152       <xs:choice  minOccurs="0" maxOccurs="unbounded">
            <xs:element name="tag"     type="constructTagType"/>
154         <xs:element name="tags"    type="constructTagsType"/>
            <xs:element name="for"     type="forType"/>
156         <xs:element name="node"    type="constructNodeType"/>
            <xs:element name="connect" type="connectType"/>
158         <xs:element name="connect_sidewalks"
            type="connectSidewalkType"/>
160         <xs:element name="try"     type="tryType"/>
```

```
      <xs:element name="gtl"     type="xs:string"/>
    </xs:choice>
    <xs:attribute name="is"      type="xs:string"/>
    <xs:attribute name="as"      type="xs:string"/>
    <xs:attribute name="polygon" type="xs:string"/>
    <xs:attribute name="role"    type="xs:string"/>
  </xs:complexType>


  <xs:complexType name="constructRelationType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="tag"        type="constructTagType"/>
      <xs:element name="tags"       type="constructTagsType"/>
      <xs:element name="for"        type="forType"/>
      <xs:element name="node"       type="constructNodeType"/>
      <xs:element name="connect"    type="connectType"/>
      <xs:element name="connect_sidewalks"
      type="connectSidewalkType"/>
      <xs:element name="way"        type="constructWayType"/>
      <xs:element name="hull_polygon"
      type="hullPolygonType"/>
      <xs:element name="intersection_polygon"
      type="intersectionPolygonType"/>
      <xs:element name="disconnect" type="disconnectType"/>
      <xs:element name="parallel"    type="parallelType"/>
      <xs:element name="relation"    type="constructRelationType"/>
      <xs:element name="try"         type="tryType"/>
      <xs:element name="gtl"         type="xs:string"/>
    </xs:choice>
    <xs:attribute name="is"   type="xs:string"/>
    <xs:attribute name="as"   type="xs:string"/>
    <xs:attribute name="role" type="xs:string"/>
  </xs:complexType>
  <xs:complexType name="constructTagType">
    <xs:attribute name="k" type="xs:string" use="required"/>
    <xs:attribute name="v" type="xs:string" use="required"/>
  </xs:complexType>

  <xs:complexType name="constructTagsType">
    <xs:attribute name="is" type="xs:string" use="required"/>
    <!-- all tags of a previously identified object will be used -->
  </xs:complexType>

  <xs:complexType name="cropType">
    <xs:attribute name="way" type="xs:string" use="required" />
    <xs:attribute name="max_length" type="xs:float"/>
```

```
208   </xs:complexType>

210   <xs:complexType name="connectType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
212       <xs:element name="tag"  type="constructTagType"/>
          <xs:element name="tags" type="constructTagsType"/>
214     </xs:choice>
        <xs:attribute name="way1"  type="xs:string" use="required"/>
216     <xs:attribute name="way2"  type="xs:string" use="required"/>
        <xs:attribute name="from1" type="xs:string" use="required"/>
218     <xs:attribute name="to1"   type="xs:string" use="required"/>
        <xs:attribute name="from2" type="xs:string" use="required"/>
220     <xs:attribute name="to2"   type="xs:string" use="required"/>
        <xs:attribute name="role"  type="xs:string"/>
222   </xs:complexType>

224   <xs:complexType name="connectSidewalkType">
        <xs:attribute name="on"      type="xs:string" use="required"/>
226     <xs:anyAttribute processContents="skip"/>
        <!-- any attribute is substituted for the following attributes,
228     where X stands for a natural number:
        <xs:attribute name="alongX"    type="xs:string" use="required"/>
230     <xs:attribute name="relationAX"
        type="xs:string" use="required"/>
232     <xs:attribute name="relationBX"
        type="xs:string" use="required"/>
234     <xs:attribute name="roleAX"  type="xs:string" use="required"/>
        <xs:attribute name="roleBX"  type="xs:string" use="required"/>
236     -->
      </xs:complexType>
238
      <xs:complexType name="forType">
240     <xs:complexContent>
          <xs:extension base="constructType">
242         <xs:attribute name="on"   type="xs:string"  use="required"/>
            <xs:attribute name="type" type="xs:string"  use="required"/>
244         <xs:attribute name="from" type="xs:string"  use="required"/>
            <xs:attribute name="to"   type="xs:string"  use="required"/>
246         <xs:attribute name="id"   type="xs:string"  use="required"/>
            <xs:attribute name="offset" type="xs:integer"/>
248         <xs:attribute name="insert" type="xs:string"/>
            <xs:attribute name="role"   type="xs:string"/>
250       </xs:extension>
        </xs:complexContent>
252   </xs:complexType>

254   <xs:complexType name="tryType">
```

```
    <xs:complexContent>
      <xs:extension base="constructType">
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="hullPolygonType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="tag"  type="constructTagType"/>
      <xs:element name="tags" type="constructTagsType"/>
    </xs:choice>
    <xs:attribute name="type"  type="xs:string" use="required"/>
    <xs:attribute name="on"    type="xs:string" use="required"/>
    <xs:attribute name="along"                  type="xs:string"/>
    <xs:attribute name="width"                  type="xs:string"/>
    <xs:attribute name="length"                 type="xs:string"/>
    <xs:attribute name="distance"               type="xs:string"/>
    <xs:attribute name="from"                   type="xs:string"/>
    <xs:attribute name="to"                     type="xs:string"/>
    <xs:attribute name="width_key"              type="xs:string"/>
    <xs:attribute name="width_left_key"         type="xs:string"/>
    <xs:attribute name="width_right_key"        type="xs:string"/>
    <xs:attribute name="distance_key"           type="xs:string"/>
    <xs:attribute name="distance_left_key"      type="xs:string"/>
    <xs:attribute name="distance_right_key"     type="xs:string"/>
    <xs:attribute name="role"                   type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="disconnectType">
      <xs:attribute name="way" type="xs:string" use="required"/>
      <xs:attribute name="at"  type="xs:string" use="required"/>
  </xs:complexType>

  <xs:complexType name="intersectionPolygonType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="tag"  type="constructTagType"/>
      <xs:element name="tags" type="constructTagsType"/>
    </xs:choice>
    <xs:attribute  name="type"    type="xs:string" use="required"/>
    <xs:attribute  name="subtype" type="xs:string" use="optional"/>
    <xs:attribute  name="on"      type="xs:string" use="required"/>
    <xs:attribute  name="length"  type="xs:string" use="optional"/>
    <xs:attribute  name="role"    type="xs:string"/>
    <xs:anyAttribute processContents="skip"/>
    <!-- any attribute is substituted for the following attributes,
    where X stands for a natural number:
    <xs:attribute name="alongX" type="xs:string" use="required"/>
```

```
302     <xs:attribute name="widthX"     type="xs:string"/>
        <xs:attribute name="distanceAX" type="xs:string"/>
304     <xs:attribute name="distanceBX" type="xs:string"/>
        <xs:attribute name="widthAX"    type="xs:string"/>
306     <xs:attribute name="widthBX"    type="xs:string"/>
        -->
308   </xs:complexType>

310   <xs:complexType name="parallelType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
312       <xs:element name="tag"  type="constructTagType"/>
          <xs:element name="tags" type="constructTagsType"/>
314     </xs:choice>
        <xs:attribute name="on"        type="xs:string" use="required"/>
316     <xs:attribute name="distance"  type="xs:string" use="required"/>
        <xs:attribute name="from"        type="xs:string"/>
318     <xs:attribute name="to"          type="xs:string"/>
        <xs:attribute name="lengthen_by" type="xs:string"/>
320     <xs:attribute name="role"        type="xs:string"/>
      </xs:complexType>
322
    </xs:schema>
```

Explanation of Transformation Attributes

The following tables give a detailed explanation of the attributes of different XML elements used in Map Content Transformations.

Table A.1: Rule Options. A list of all available options for the different attributes of the `<rule>` element. The leftmost column gives the name of the attribute.

Attribute	Option	Meaning
order	[integer]	Rules are executed in the order prescribed by this attribute. Unless explicitly requested otherwise, later rules will not consider objects that were given an identifier by previous rules.
source	original	Identification is performed only in the original map.
	transformed	Identification is performed only in the results of previous transformations of the same rule.
	both	Identification is performed in both the original map and the results of previous transformations.
permutations	true	Result can include different result sets that include the same objects as other result sets, where only the assignment of identifiers is permuted.
	false	Result will not include results sets that include the same set of objects.

Table A.2: For Attributes. A list of attributes and their options for the `<for>` element.

Attrib.	Option	Meaning
on	[string]	Specifies the object on which to iterate. Any previously identified way or relation can be used.
type	node	Iterates only on nodes (for ways and relations).
	way	Iterates on ways (only for relations).
	relation	Iterates on relations (only for relations).
from	[string]	Defines where to start the iteration. Can bei either an identifier of an object, or the strings “first” or “last”, specifying to start at the first or last object. Additionally, +X or -X (where X is an integer) can be used to specify a position X iteration steps away from the given object. Defaults to “first”.
to	[string]	Defines where to end the iteration. Same usage as “from”. If “to” comes before “from” in the ordering, it is automatically iterated backward. Defaults to “last”.
id	[string]	String that identifies the current object in each iteration step.
offset	[integer]	Specifies a positive or negative offset of objects that are available in the same iteration step. These objects are then referenceable with “Id+X”, where X is between 0 and the value of offset. If an offset is set, the “to” or “from” values must be set accordingly, so that no objects outside of the range are referenced.
insert	before	Objects created by sub-elements are inserted into the object on which is being iterated before the current object during each iteration step.
	after	Objects created by sub-elements are inserted after the current object. If “insert” is not set (default), no objects are inserted.
role	[string]	Iterates only on objects of the specified role (only for relations).

Table A.3: Hull Polygon Options. Options for different attributes of the `<hull_polygon>` element.

Attribute	Option	Meaning
type	way_rectangle	Create a polygon with a constant width that follows the layout of the way.
	way_trapezoid	Create a polygon that follows the layout of the way but has a different width at each node.
	node_rectangle	Create a rectangle around a node that is aligned with a specified way.
on	[string]	Id string of either a way or a node, depending on the type. The hull is created around the identified object.
from	[string]	Defines at which node to start the polygon. Can bei either an identifier of an object, or the string “first” or “last”, specifying to start at the first or last object. Additionally, +X or -X (where X is an integer) can be used to specify a position X iteration steps away from the given object. Defaults to “first”.
to	[string]	Defines at which node to end the polygon, with the same rules as “from”, except that it defaults to “last”.
along	[string]	Id string of a way. A node_rectangle is aligned with this way.
width	[real]	Width of the polygon. Can be either a real number or a more complex expression.
length	[real]	Length of the polygon. Can be either a real number or a more complex expression.
distance	[real]	Distance between center of polygon and street. Can be used to create sidewalks, etc.
width_key	[string]	The value of this key will be looked up in the tags of each node and determines the width of a trapezoid at that node.
width_left_key	[string]	The value of this key determines the width of the left half of a trapezoid at each node.
width_right_key	[string]	The value of this key determines the width of the right half of a trapezoid at each node.
distance_key	[string]	The value of this key determines the distance of the way_trapezoid to the original way at each node.

Table A.4: Intersection Polygon Options. Options for different attributes of the `<intersection_polygon>` element.

Attribute	Option	Meaning
type	intersection_rectangle	Currently the only available option, creating a polygonal representation of either the street or the sidewalks at the intersection.
subtype	[empty]	Create a polygonal representation of the street.
	distance	Create a polygonal representation of the sidewalks. The several resulting polygons are not connected.
on	[string]	Id string of the node that represents the intersection.
length	[real]	Length of those parts of the intersection polygon protruding into the streets.
alongX	[string]	Id strings of the ways that make up the intersection. X is a natural number, and all ways that meet at the intersection need to be enumerated.
widthX	[real]	Widths of the streets. X is a natural number, and each width corresponds to the way of the same number. Only used if subtype is empty.
distanceAX, distanceBX	[real]	Distance of sidewalk polygons to the way. X is a natural number, referring to the way. A and B are two sidewalks, left sidewalks being indicated by negative numbers. Only used if subtype is `"distance"`.
widthAX, widthBX	[real]	Widths of sidewalk polygons. The number X refers to the way. A and B refer to the two sidewalks, and correspond to A and B of the distances, which determine the left and right positions. Only used if subtype is `"distance"`.

Table A.5: Parallel Options. Options for different attributes of the `<parallel>` element.

Attribute	Option	Meaning
on	[string]	Id string of the way to which the parallel shall be created
distance	[real]	Distance of the parallel to the way
from	[string]	Defines at which node to start the parallel. Can bei either an identifier of an object, or the string “first” or “last”, specifying to start at the first or last object. Additionally, +X or -X (where X is an integer) can be used to specify a position X iteration steps away from the given object. Defaults to “first”.
to	[string]	Defines at which node to end the parallel, with the same rules as “from”, except that it defaults to “last”.
lengthen_by	[real]	Lengthen both ends of the parallel by the given amount. This can be useful for later topological constructions

Table A.6: Connect Sidewalks Attributes. The different Attributes of the `<connect_sidewalk>` element.

Attribute	Option	Meaning
on	[string]	The node that represents the intersection.
alongX	[string]	n attributes, with X being an integer between 1 and n, identifying the n ways that meet at the intersection.
relationAX	[string]	n attributes, identifying a relation that contains one of the sidewalks of alongX.
relationBX	[string]	n attributes, identifying a relation that contains the other sidewalk of alongX.
roleAX	[string]	n attributes, naming the role that one of the sidewalks has in relationAX.
roleBX	[string]	n attributes, naming the role that the other sidewalk has in relationBX.

Grammar and Effects of GTL

The following listing and table describe the EBNF of GTL and the effects of the different operators.

```
Decl     = "POINT" | "REAL" Id {"," Id} ";"                    ;
Stmnt    = (Id | LSeg) "=" (Seg | SgmConst | RCalc) ";"        ;
LSeg     = "[" Id "," Id "]"                                   ;
Seg      = "[" Point "," Point "]"                             ;
InnerSeg = Seg | ("(" SgmConst ")")                            ;
Point    = Id | SgmConst | ("(" SgmConst ")")                  ;
SgmConst = Intsec | SgmCalc                                    ;
Intsec   = InnerSeg ("X" | "x" | "T" | "t") InnerSeg           ;
SgmCalc  = InnerSeg ("|" | "L" | "*" | "+" | "-" | "V") RCalc  ;
RCalc    = DistC | AngleC | Arith | Real | Id                  ;
DistC    = "DISTANCE(" (Seg | SgmConst) ")"                    ;
AngleC   = "ANGLE(" (Seg | SgmConst) "," (Seg | SgmConst) ")"  ;
Arith    = "(" RCalc ("+" | "-" | "*" | "/") RCalc ")"         ;
```

Listing A.1: EBNF of the GTL. A complete GTL section consists of an arbitrary number of Declarations (Decl) and Statements (Stmnt). In this EBNF the specifications for Id and Real are omitted. Ids are identifiers, as used in the XML language, including those that contain offsets. Real are real numbers.

Table A.7: Geometric Construction. The different operators of the geometric construction rules and their result.

Option	Effect
sg X sg	Creates a segment from the first point of the first segment to the intersection of the two lines that are represented by the two segments.
sg x sg	Creates a segment from the first point of the first segment to the intersection of the two segments. Throws an exception if the two segments do not intersect
sg T sg	Creates a segment from the second point of the second segment to the nearest point on the line represented by the first segment (perpendicular).
sg t sg	Creates a segment from the second point of the second segment to the intersection of the perpendicular to the first segment with the first segment. Throws an exception if the perpendicular doesn't intersect the segment
sg | real	Creates a segment parallel to the given segment with the given distance.
sg L real	Creates a segment from the first point of the given segment to a point perpendicular to the given segment with the given distance.
sg V real	Creates a segment of length 1 with the given angle, beginning at the first point of the given segment.
sg * real	Creates a segment in the same direction as the given segment, with the length multiplied by multiplicator.
sg + real	Creates a segment in the same direction as the given segment, with distance added to the length.
sg - real	Creates a segment in the same direction as the given segment, with distance subtracted from the length.
DISTANCE(sg)	Computes the distance between the two points of the segment.
ANGLE(sg, sg)	Computes the angle between the two segments.

B

List of Abbreviations

AWM Augmented World Model

DOM Document Object Model

EBNF Extended Backus-Naur Form

ECS Extended Class Schema

ETAs Electronic Travel Aids

GML Geography Markup Language

GPS Global Positioning System

GTL Geometric Transformation Language

O&M Orientation and Mobility

RF Radio Frequency

RFID Radio Frequency Identification

SCS Standard Class Schema

VVS Verkehrsverbund Stuttgart

WML Wireless Markup Language

XML Extensible Markup Language

XSLT Extensible Stylesheet Language Transformations

Bibliography

[1] Tomohiro Amemiya and Hisashi Sugiyama. Orienting kinesthetically: A haptic handheld wayfinder for people with visual impairments. *ACM Transactions on Accessible Computing (TACCESS)*, 3(2):6:1–6:23, 2010. DOI:10.1145/1857920.1857923. (Cited on page 25.)

[2] Tomohiro Amemiya, Jun Yamashita, Koichi Hirota, and Michitaka Hirose. Virtual leading blocks for the deaf-blind: A real-time way-finder by verbal-nonverbal hybrid interface and high-density RFID tag space. In *Proceedings of the IEEE Virtual Reality 2004*, pages 165–287, 2004. DOI:10.1109/VR.2004.1310070. (Cited on pages 14, 18, 24, and 26.)

[3] Tomohiro Amemiya, Hideyuki Ando, and Taro Maeda. Phantom-drawn: direction guidance using rapid and asymmetric acceleration weighted by nonlinearity of perception. In *Proceedings of the 2005 international conference on Augmented tele-existence (ICAT '05)*, pages 201–208, 2005. DOI:10.1145/1152399.1152436. (Cited on pages 25 and 26.)

[4] Tomohiro Amemiya, Hideyuki Ando, and Taro Maeda. Lead-me interface for a pulling sensation from hand-held devices. *ACM Transactions on Applied Perception (TAP)*, 5(3):15:1–15:17, 2008. DOI:10.1145/1402236.1402239. (Cited on pages 25 and 26.)

[5] Andrea Ballatore, Gavin McArdle, Caitriona Kelly, and Michela Bertolotto. Recomap: an interactive and adaptive map-based recommender. In *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10)*, pages 887–891, 2010. DOI:10.1145/1774088.1774273. (Cited on page 44.)

[6] Nikola Banovic, Rachel L. Franz, Khai N. Truong, Jennifer Mankoff, and Anind K. Dey. Uncovering information needs for independent spatial learning for users who are visually impaired. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '13)*, pages 24:1–24:8, 2013. DOI:10.1145/2513383.2513445. (Cited on pages 1, 4, 5, and 6.)

[7] Ling Bao and Stephen Intille. Activity recognition from user-annotated acceleration data. In *Pervasive Computing - Second*

International Conference (PERVASIVE 2004), pages 1–17, 2004. DOI:10.1007/978-3-540-24646-6_1. (Cited on pages 65 and 67.)

[8] Holger Bast, Stefan Funke, Domagoj Matijevic, Peter Sanders, and Dominik Schultes. In transit to constant time shortest-path queries in road networks. In *9th Workshop on Algorithm Enginneering and Experiments (ALENEX '07)*, pages 46–59, 2007. DOI:10.1137/1.9781611972870.5. (Cited on page 109.)

[9] Stéphane Beauregard, Widyawan, and Martin Klepal. Indoor PDR performance enhancement using minimal map information and particle filters. In *IEEE/ION Position, Location and Navigation Symposium (PLANS 2008)*, pages 141 –147, 2008. DOI:10.1109/PLANS.2008.4570050. (Cited on page 64.)

[10] Susanne Becker. Generation and application of rules for quality dependent façade reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64:640–653, 2009. DOI:10.1016/j.isprsjprs.2009.06.002. (Cited on page 36.)

[11] Masahiro Bessho, Shinsuke Kobayashi, Noboru Koshizuka, and Ken Sakamura. A pedestrian navigation system using multiple space-identifying devices based on a unique identifier framework. In *2007 International Conference on Machine Learning and Cybernetics*, pages 2100–2105, 2007. DOI:10.1109/ICMLC.2007.4370491. (Cited on pages 15 and 18.)

[12] Masahiro Bessho, Shinsuke Kobayashi, Noboru Koshizuka, and Ken Sakamura. A space-identifying ubiquitous infrastructure and its application for tour-guiding service. In *Proceedings of the 2008 ACM symposium on Applied computing (SAC '08)*, pages 1616–1621, 2008. DOI:10.1145/1363686.1364069. (Cited on pages 15, 18, and 21.)

[13] Masahiro Bessho, Shinsuke Kobayashi, Noboru Koshizuka, and Ken Sakamura. Assisting mobility of the disabled using space-identifying ubiquitous infrastructure. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility (Assets '08)*, pages 283–284, 2008. DOI:10.1145/1414471.1414539. (Cited on pages 15 and 18.)

[14] André Blessing and Hinrich Schütze. Automatic acquisition of vernacular places. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services (iiWAS '08)*, pages 662–665, 2008. DOI:10.1145/1497308.1497437. (Cited on page 37.)

[15] Andre Blessing, Stefan Klatt, Daniela Nicklas, Steffen Volz, and Hinrich Schütze. Language-derived information and context models. In *Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops. (PerCom Workshops 2006)*, pages 24–29, 2006. DOI:10.1109/PERCOMW.2006.72. (Cited on pages 36 and 38.)

[16] Johann Borenstein and Iwan Ulrich. The guidecane – a computerized travel aid for the active guidance of blind pedestrians. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation (ICRA 97)*, pages 1283–1288, 1997. DOI:10.1109/ROBOT.1997.614314. (Cited on pages 11, 25, and 26.)

[17] Nikolaos Bourbakis and Despina Kavraki. A 2D vibration array for sensing dynamic changes and 3D space for blinds' navigation. In *Proceedings of the Fifth IEEE Symposium on Bioinformatics and Bioengineering (BIBE '05)*, pages 222–226, 2005. DOI:10.1109/BIBE.2005.1. (Cited on page 11.)

[18] Calvert L. Bowen, Timothy K. Buennemeyer, Ingrid Burbey, and Vinit Joshi. Using wireless networks to assist navigation for individuals with disabilities. In *Proceedings of the California State University, Northridge Center on Disabilities' 21st Annual International Technology and Persons with Disabilities Conference (CSUN 2006); Los Angeles, CA, USA*, 2006. (Cited on pages 15 and 18.)

[19] Nicholas A. Bradley and Mark D. Dunlop. An experimental investigation into wayfinding directions for visually impaired people. *Personal and Ubiquitous Computing*, 9(6):395–403, 2005. DOI:10.1007/s00779-005-0350-y. (Cited on pages 21 and 42.)

[20] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984. (Cited on page 66.)

[21] Volkert Buchmann, Mark Billinghurst, and Andy Cockburn. Directional interfaces for wearable augmented reality. In *Proceedings of the 9th ACM SIGCHI New Zealand Chapter's International Conference on Human-Computer Interaction (CHINZ '08)*, pages 47–54, 2008. DOI:10.1145/1496976.1496983. (Cited on pages 25 and 26.)

[22] Dirk Burghardt, Moritz Neun, and Robert Weibel. Generalization services on the web—classification and an initial prototype implementation. *Cartography and Geographic Information Science*, 32

(4):257–268, 2005. DOI:10.1559/152304005775194665. (Cited on pages 44 and 45.)

[23] Yao-Jen Chang, Shih-Kai Tsai, and Tsen-Yung Wang. A context aware handheld wayfinding system for individuals with cognitive impairments. In *Assets '08: Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, pages 27–34, 2008. DOI:10.1145/1414471.1414479. (Cited on page 20.)

[24] Adrian David Cheok and Yue Li. Ubiquitous interaction with positioning and navigation using a novel light sensor-based information transmission system. *Personal and Ubiquitous Computing*, 12(6):445–458, 2008. DOI:10.1007/s00779-007-0140-9. (Cited on pages 17, 18, and 21.)

[25] California Vehicle Code. Section 467: Pedestrian. URL https://www.dmv.ca.gov/pubs/vctop/d01/vc467.htm. (Cited on page 3.)

[26] Robert F. Cohen, Valerie Haven, Jessica A. Lanzoni, Arthur Meacham, Joelle Skaff, and Michael Wissell. Using an audio interface to assist users who are visually impaired with steering tasks. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility (Assets '06)*, pages 119–124, 2006. DOI:10.1145/1168987.1169008. (Cited on pages 23, 26, and 80.)

[27] Ahmed El-Safty, Bernhard Schmitz, and Thomas Ertl. An openstreetmap editing interface for visually impaired users based on geosemantic information. In *Proceedings of the 14th International Conference on Computers Helping People with Special Needs (ICCHP 2014)*, pages 116–119, 2014. DOI:10.1007/978-3-319-08599-9_18. (Not cited.)

[28] Sevgi Ertan, Clare Lee, Abigail Willets, Hong Tan, and Alex Pentland. A wearable haptic navigation guidance system. In *Second International Symposium on Wearable Computers (Digest of Papers)*, pages 164–165, 1998. DOI:10.1109/ISWC.1998.729547. (Cited on pages 17, 25, and 26.)

[29] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI '99)*, pages 343–349, 1999. (Cited on page 64.)

[30] Giuseppe Ghiani, Barbara Leporini, and Fabio Paternò. Vibrotactile feedback as an orientation aid for blind users of mobile guides. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services (MobileHCI '08)*, pages 431–434, 2008. DOI:10.1145/1409240.1409306. (Cited on pages 15, 18, 24, and 26.)

[31] Giuseppe Ghiani, Barbara Leporini, and Fabio Paternò. Supporting orientation for blind people using museum guides. In *CHI '08 extended abstracts on Human factors in computing systems (CHI EA '08)*, pages 3417–3422, 2008. DOI:10.1145/1358628.1358867. (Cited on pages 15 and 18.)

[32] Saurabh Godha, Gérard Lachapelle, and M. Elizabeth Cannon. Integrated GPS/INS system for pedestrian navigation in a signal degraded environment. In *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, pages 2151–2164, 2006. (Cited on pages 16 and 18.)

[33] Sumit Gulwani, Vijay Anand Korthikanti, and Ashish Tiwari. Synthesizing geometry constructions. *ACM SIGPLAN Notices*, 46:50–61, 2011. DOI:10.1145/1993316.1993505. (Cited on page 45.)

[34] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data (SIGMOD '84)*, pages 47–57, 1984. DOI:10.1145/602259.602266. (Cited on page 59.)

[35] Hele-Mai Haav, Aivi Kaljuvee, Martin Luts, and Toivo Vajakas. Ontology-based retrieval of spatially related objects for location based services. In *On the Move to Meaningful Internet Systems (OTM 2009)*, pages 1010–1024, 2009. DOI:10.1007/978-3-642-05151-7_19. (Cited on page 44.)

[36] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. DOI:10.1109/TSSC.1968.300136. (Cited on page 109.)

[37] Chris Heathcote. 35 ways to find your location. In *Emerging Technologies Conference*, 2004. Presentation. (Cited on page 12.)

[38] Abdelsalam (Sumi) Helal, Steven Edwin Moore, and Balaji Ramachandran. Drishti: An integrated navigation system for visually impaired and disabled. In *Proceedings of the 5th IEEE International*

Symposium on Wearable Computers (ISWC '01), pages 149–156, 2001. DOI:10.1109/ISWC.2001.962119. (Cited on pages 14, 18, and 19.)

[39] Wilko Heuten, Daniel Wichmann, and Susanne Boll. Interactive 3D sonification for the exploration of city maps. In *NordiCHI '06: Proceedings of the 4th Nordic conference on human-computer interaction*, pages 155–164, 2006. DOI:10.1145/1182475.1182492. (Cited on pages 23, 26, and 80.)

[40] Wilko Heuten, Niels Henze, and Susanne Boll. Interactive exploration of city maps with auditory torches. In *CHI '07 extended abstracts on human factors in computing systems (CHI EA '07)*, pages 1959–1964, 2007. DOI:10.1145/1240866.1240932. (Cited on pages 23, 26, and 80.)

[41] Wilko Heuten, Niels Henze, Susanne Boll, and Martin Pielot. Tactile wayfinder: a non-visual support system for wayfinding. In *Proceedings of the 5th Nordic conference on Human-computer interaction (NordiCHI '08)*, pages 172–181, 2008. DOI:10.1145/1463160.1463179. (Cited on pages 25, 26, and 126.)

[42] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, 2001. DOI:10.1109/2.940014. (Cited on page 12.)

[43] Michitika Hirose and Tomohiro Amemiya. Wearable finger-braille interface for navigation of deaf-blind in ubiquitous barrier-free space. In *Proceedings of the 10th International Conference on Human-Computer Interaction*, pages 1417–1421, 2003. (Cited on pages 14, 24, and 26.)

[44] Insu Hong and Alan T. Murray. Efficient measurement of continuous space shortest distance around barriers. *International Journal of Geographical Information Science*, 27(12):2302–2318, 2013. DOI:10.1080/13658816.2013.788182. (Cited on page 110.)

[45] Nicola Hönle, Uwe-Philipp Käppeler, Daniela Nicklas, Thomas Schwarz, and Matthias Großmann. Benefits of integrating meta data into a context model. In *3rd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2005 Workshops)*, pages 25–29, 2005. DOI:10.1109/PERCOMW.2005.20. (Cited on page 33.)

[46] Andreas Hub and Bernhard Schmitz. Addition of RFID-based initialization and object recognition to the navigation system TANIA. In *Proceedings of the California State University, Northridge Center on Disabilities' 24th Annual International Technology and Persons with Disabilities Conference (CSUN 2009)*, 2009. Extended Abstract. (Not cited.)

[47] Andreas Hub, Joachim Diepstraten, and Thomas Ertl. Design and development of an indoor navigation and object identification system for the blind. In *Proceedings of the 6th International ACM SIGACCESS Conference on Computers and Accessibility (Assets '04)*, pages 147–152, 2004. DOI:10.1145/1028630.1028657. (Cited on pages 17 and 18.)

[48] Andreas Hub, Joachim Diepstraten, and Thomas Ertl. Augmented indoor modeling for navigation support for the blind. In *Proceedings of the International Conference on Computers for People with Special Needs (CPSN 2005)*, pages 54–59, 2005. (Cited on pages 17 and 18.)

[49] Andreas Hub, Joachim Diepstraten, and Thomas Ertl. Design of an object identification and orientation assistant for the deafblind. In *Proceedings of the 6th DbI European Conference on Deafblindness*, 2005. (Cited on pages 17, 18, 21, and 23.)

[50] Andreas Hub, Tim Hartter, and Thomas Ertl. Interactive tracking of movable objects for the blind on the basis of environment models and perception-oriented object recognition methods. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility (Assets '06)*, pages 111–118, 2006. DOI:10.1145/1168987.1169007. (Cited on pages 15, 16, 18, 24, and 26.)

[51] Andreas Hub, Stefan Kombrink, Klaus Bosse, and Thomas Ertl. TANIA – a tactile-acoustical navigation and information assistant for the 2007 CSUN conference. In *Proceedings of the California State University, Northridge Center on Disabilities' 22nd Annual International Technology and Persons with Disabilities Conference (CSUN 2007)*, 2007. (Cited on pages 16, 18, 21, 23, 24, 26, and 88.)

[52] Andreas Hub, Stefan Kombrink, Klaus Bosse, and Thomas Ertl. Conference navigation and communication assistant for the deafblind based on tactile and acoustically amplified augmented map information for the 14th deafblind international world conference. In *Proceedings of the 14th Deafblind International World Conference (DbI 2007)*, 2007. (Cited on pages 16, 18, 21, 23, 24, and 26.)

[53] Takafumi Ienaga, Yukinobu Sugimura, Yoshihiko Kimuro, and Chikamune Wada. Pedestrian navigation system using tone gradient and robotic gis. In *Proceedings of the 12th International Conference on Computers Helping People with Special Needs (ICCHP 2010)*, pages 239–246, 2010. DOI:10.1007/978-3-642-14100-3_36. (Cited on pages 15, 18, 21, 22, and 26.)

[54] Volodymyr Ivanchenko, James Coughlan, William Gerrey, and Huiying Shen. Computer vision-based clear path guidance for blind wheelchair users. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility (Assets '08)*, pages 291–292, 2008. DOI:10.1145/1414471.1414543. (Cited on page 12.)

[55] Predrag Janičić. Geometry constructions language. *Journal of Automated Reasoning*, 44:3–24, 2010. DOI:10.1007/s10817-009-9135-8. (Cited on page 45.)

[56] Dean M. Karantonis, Michael R. Narayanan, Merryn Mathie, Nigel H. Lovell, and Branko G. Celler. Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *IEEE Transactions on Information Technology in Biomedicine*, 10(1):156 –167, 2006. DOI:10.1109/TITB.2005.856864. (Cited on page 65.)

[57] Eunjeong Ko, Jin Sun Ju, and Eun Yi Kim. Situation-based indoor wayfinding system for the visually impaired. In *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 35–42, 2011. DOI:10.1145/2049536.2049545. (Cited on pages 18, 21, and 26.)

[58] Masakatsu Kourogi and Takeshi Kurata. Personal positioning based on walking locomotion analysis with self-contained sensors and a wearable camera. In *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR '03)*, page 103, 2003. DOI:10.1109/ISMAR.2003.1240693. (Cited on pages 65 and 67.)

[59] Bernhard Krach and Patrick Robertson. Cascaded estimation architecture for integration of foot-mounted inertial sensors. In *IEEE/ION Position, Location and Navigation Symposium (PLANS 2008)*, pages 112 –119, 2008. DOI:10.1109/PLANS.2008.4570047. (Cited on page 64.)

[60] Lars Kulik, Matt Duckham, and Max Egenhofer. Ontology-driven map generalization. *Journal of Visual Languages & Computing*, 16 (3):245 – 267, 2005. DOI:10.1016/j.jvlc.2005.02.001. (Cited on page 44.)

[61] Vladimir Kulyukin, Chaitanya Gharpure, John Nicholson, and Grayson Osborne. Robot-assisted wayfinding for the visually impaired in structured indoor environments. *Autonomous Robots*, 21(1):29–41, 2006. DOI:10.1007/s10514-006-7223-8. (Cited on pages 16, 18, 19, 25, and 26.)

[62] Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, Jason Tabert, Pauline Powledge, Gaetano Borriello, and Bill Schilit. Place lab: Device positioning using radio beacons in the wild. In *Pervasive Computing – Third International Conference (PERVASIVE 2005)*, pages 116–133, 2005. DOI:10.1007/11428572_8. (Cited on page 15.)

[63] Lassi Lehto and Tiina Kilpeläinen. Generalizing xml-encoded spatial data on the web. In *Proceedings of the 20th International Cartographic Conference (ICC 2001)*, pages 2390–2396, 2001. (Cited on pages 44 and 47.)

[64] Jack M. Loomis, Reginald G. Golledge, Roberta L. Klatzky, Jon M. Speigle, and Jerome Tietz. Personal guidance system for the visually impaired. In *Proceedings of the first annual ACM conference on Assistive technologies (Assets '94)*, pages 85–91, 1994. DOI:10.1145/191028.191051. (Cited on pages 14, 18, 21, 22, and 26.)

[65] Jack M. Loomis, Reginald G. Golledge, and Roberta L. Klatzky. Navigation system for the blind: Auditory display modes and guidance. *Presence: Teleoperators and Virtual Environments*, 7(2):193–203, 1998. DOI:10.1162/105474698565677. (Cited on pages 14 and 18.)

[66] Jack M. Loomis, James R. Marston, Reginald G. Golledge, and Roberta L. Klatzky. Personal guidance system for people with visual impairment: A comparison of spatial displays for route guidance. *Journal of Visual Impairment & Blindness*, 99(4):219–232, 2005. (Cited on pages 14, 18, and 20.)

[67] Robert J. Lutz. *taux: A System For Evaluating Sound Feedback In Navigational Tasks*. PhD thesis, New Jersey Institute of Technology, 2008. (Cited on pages 22 and 26.)

[68] Eoin Mac Aoidh, Gavin McArdle, Mathieu Petit, Cyril Ray, Michela Bertolotto, Christophe Claramunt, and David Wilson. Personalization in adaptive and interactive GIS. *Annals of GIS*, 15(1):23–33, 2009. DOI:10.1080/00207720903270985. (Cited on page 44.)

[69] James R. Marston, Jack M. Loomis, Roberta L. Klatzky, Reginald G. Golledge, and Ethan L. Smith. Evaluation of spatial displays for navigation without sight. *ACM Transactions on Applied Perception (TAP)*, 3(2):110–124, 2006. DOI:10.1145/1141897.1141900. (Cited on pages 22 and 26.)

[70] James R. Marston, Jack M. Loomis, Roberta L. Klatzky, and Reginald G. Golledge. Nonvisual route following with guidance from a simple haptic or auditory display. *Journal of Visual Impairment & Blindness*, 101(4):203–211, 2007. (Cited on pages 14, 22, 24, 26, 126, and 127.)

[71] Bernhard Mitschang, Daniela Nicklas, Matthias Großmann, Thomas Schwarz, and Nicola Hönle. Federating location-based data services. In *Data Management in a Connected World, Essays Dedicated to Hartmut Wedekind on the Occasion of His 70th Birthday*, pages 17–35, 2005. (Cited on page 32.)

[72] Astrid Müller, Pascal Neis, Michael Auer, and Alexander Zipf. Ein Routenplaner für Rollstuhlfahrer auf der Basis von OpenStreetMap-Daten - Konzeption, Realisierung und Perspektiven. In *Angewandte Geoinformatik 2010 – 22. AGIT-Symposium*, 2010. (Cited on pages 110 and 111.)

[73] Bijan Najafi, Kamiar Aminian, Anisoara Paraschiv-Ionescu, François Loew, Christophe J. Büla, and Philippe Robert. Ambulatory system for human motion analysis using a kinematic sensor: monitoring of daily physical activity in the elderly. *IEEE Transactions on Biomedical Engineering*, 50(6):711–723, 2003. DOI:10.1109/TBME.2003.812189. (Cited on page 65.)

[74] Pascal Neis, Dennis Zielstra, and Alexander Zipf. The street network evolution of crowdsourced maps: OpenStreetMap in Germany 2007–2011. *Future Internet*, 4(1):1–21, 2012. DOI:10.3390/fi4010001. (Cited on page 42.)

[75] Moritz Neun, Dirk Burghardt, and Robert Weibel. Automated processing for map generalization using web services. *GeoInformatica*, 13(4):425–452, 2009. DOI:10.1007/s10707-008-0054-3. (Cited on page 44.)

[76] Daniela Nicklas and Bernhard Mitschang. On building location aware applications using an open platform based on the NEXUS augmented world model. *Software and System Modeling*, 3(4):303–313, 2004. DOI:10.1007/s10270-004-0055-0. (Cited on page 33.)

[77] Daniela Nicklas, Matthias Großmann, Thomas Schwarz, Steffen Volz, and Bernhard Mitschang. A model-based, open architecture for mobile, spatially aware applications. In *Proceedings of the 7th International Symposium on Spatial and Temporal Databases (SSTD 2001)*, pages 117–135, 2001. DOI:10.1007/3-540-47724-1_7. (Cited on page 32.)

[78] Devi Archana Paladugu, Zheshen Wang, and Baoxin Li. On presenting audio-tactile maps to visually impaired users for getting directions. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems (CHI EA '10)*, pages 3955–3960, 2010. DOI:10.1145/1753846.1754085. (Cited on page 81.)

[79] Helen Petrie, Valerie Johnson, Thomas Strothotte, Andreas Raab, Rainer Michel, Lars Reichert, and Axel Schalt. MoBIC: An aid to increase the independent mobility of blind travellers. *British Journal of Visual Impairment*, 15(2):63–66, 1997. (Cited on pages 12, 14, 21, and 26.)

[80] Lisa Ran, Sumi Helal, and Steve Moore. Drishti: An integrated indoor/outdoor blind navigation system and service. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2004)*, pages 23–30, 2004. DOI:10.1109/PERCOM.2004.1276842. (Cited on pages 15 and 18.)

[81] Tumasch Reichenbacher. Adaptive methods for mobile cartography. In *Proceedings of the 21st International Cartographic Conference (ICC 2003)*, pages 1311–1321, 2003. (Cited on page 44.)

[82] OpenStreetMap Wiki: Rendering. http://wiki.openstreetmap.org/wiki/Rendering. (Cited on page 43.)

[83] Matt Rice, R. Daniel Jacobson, Reginald G. Golledge, and David Jones. Design considerations for haptic and auditory map interfaces. *Cartography and Geographic Information Science (CaGIS)*, 32(4):381–391, 2005. (Cited on page 80.)

[84] David A. Ross and Bruce B. Blasch. Wearable interfaces for orientation and wayfinding. In *Proceedings of the fourth international*

ACM conference on Assistive technologies (Assets '00), pages 193–200, 2000. DOI:10.1145/354324.354380. (Cited on pages 22, 24, and 26.)

[85] David A. Ross and Bruce B. Blasch. Development of a wearable computer orientation system. *Personal and Ubiquitous Computing*, 6 (1):49–63, 2002. DOI:10.1007/s007790200005. (Cited on pages 22, 24, and 26.)

[86] David A. Ross and Alexander Lightman. Talking braille: a wireless ubiquitous computing network for orientation and wayfinding. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility (Assets '05)*, pages 98–105, 2005. DOI:10.1145/1090785.1090805. (Cited on pages 17 and 18.)

[87] Martin Rotard, Christiane Taras, and Thomas Ertl. Tactile web browsing for blind people. *Multimedia Tools and Applications*, 37(1): 53–69, 2008. DOI:10.1007/s11042-007-0170-3. (Cited on page 24.)

[88] Jaime Sánchez and Natalia de la Torre. Autonomous navigation through the city for the blind. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '10)*, pages 195–202, 2010. DOI:10.1145/1878803.1878838. (Cited on pages 14, 18, 21, and 26.)

[89] Bill N. Schilit, Anthony LaMarca, Gaetano Borriello, William G. Griswold, David McDonald, Edward Lazowska, Anand Balachandran, Jason Hong, and Vaughn Iverson. Challenge: Ubiquitous location-aware computing and the "place lab" initiative. In *Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots (WMASH '03)*, pages 29–35, 2003. DOI:10.1145/941326.941331. (Cited on page 15.)

[90] Bernhard Schmitz. The ViibraCane – a white cane for tactile navigation guidance. In *Proceedings of the California State University, Northridge Center on Disabilities' 25th Annual International Technology and Persons with Disabilities Conference (CSUN 2010)*, 2010. Extended Abstract. (Not cited.)

[91] Bernhard Schmitz and Thomas Ertl. Making digital maps accessible using vibrations. In *Proceedings of the 12th International Conference on Computers Helping People with Special Needs (ICCHP 2010)*, pages 100–107, 2010. DOI:10.1007/978-3-642-14097-6_18. (Not cited.)

[92] Bernhard Schmitz and Thomas Ertl. Rule-based transformation of map data. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops 2012)*, pages 584–589, 2012. DOI:10.1109/PerComW.2012.6197581. (Not cited.)

[93] Bernhard Schmitz and Thomas Ertl. Interactively displaying maps on a tactile graphics display. In *Proceedings of the Workshop on Spatial Knowledge Acquisition with Limited Information Displays*, pages 13–18, 2012. (Not cited.)

[94] Bernhard Schmitz and Thomas Ertl. Creating task-specific maps with map content transformations. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on MapInteraction (MapInteract '13)*, pages 84–90, 2013. DOI:10.1145/2534931.2534945. (Not cited.)

[95] Bernhard Schmitz and Thomas Ertl. Individualized route planning and guidance based on map content transformations. In *Proceedings of the 14th International Conference on Computers Helping People with Special Needs (ICCHP 2014)*, pages 120–127, 2014. DOI:10.1007/978-3-319-08599-9_19. (Not cited.)

[96] Bernhard Schmitz and Andreas Hub. Combination of the navigation system TANIA with RFID-based initialization and object recognition. In *Proceedings from 7th European Conference of ICEVI*, 2009. Poster Presentation. (Not cited.)

[97] Bernhard Schmitz, Susanne Becker, André Blessing, and Matthias Großmann. Acquisition and presentation of diverse spatial context data for blind navigation. In *12th IEEE International Conference on Mobile Data Management (MDM 2011)*, pages 276–284, 2011. DOI:10.1109/MDM.2011.66. (Cited on page 7.)

[98] Bernhard Schmitz, Attila Györkös, and Thomas Ertl. Combination of map-supported particle filters with activity recognition for blind navigation. In *Proceedings of the 13th International Conference on Computers Helping People with Special Needs (ICCHP 2012)*, 2012. DOI:10.1007/978-3-642-31534-3_78. (Not cited.)

[99] Shraga Shoval, Iwan Ulrich, and Johann Borenstein. Navbelt and the guide-cane – obstacle-avoidance systems for the blind and visually impaired. *IEEE Robotics & Automation Magazine*, 10(1):9–20, 2003. DOI:10.1109/MRA.2003.1191706. (Cited on pages 25 and 26.)

[100] Oliver Siemoneit, Christoph Hubig, Bernhard Schmitz, and Thomas Ertl. Mobiquitous devices and perception of reality. A philosophical enquiry into mobile and ubiquitous computing devices that alter perception using the example of TANIA – a tactile acoustical indoor and outdoor navigation and information assistant for the blind, deafblind, and visually-impaired users. In *Proceedings of the 5th Asia-Pacific Computing and Philosophy Conference*, pages 123–130, 2009. (Not cited.)

[101] Alexandra Skripko, Pavel Skripko, Arturs Gailitis, and Leo Selavo. RF-based node system for blind navigation in running activities. In *International Conference on Signals and Electronic Systems (ICSES '08)*, pages 527 – 530, 2008. DOI:10.1109/ICSES.2008.4673487. (Cited on pages 14, 18, 24, and 26.)

[102] Ronit Slyper and Jessica K. Hodgins. Action capture with accelerometers. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08)*, pages 193–199, 2008. (Cited on page 65.)

[103] Asim Smailagic and Richard Martin. Metronaut: A wearable computer with sensing and global communication capabilities. In *First International Symposium on Wearable Computers (Digest of Papers)*, page 116, 1997. DOI:10.1109/ISWC.1997.629927. (Cited on pages 17 and 18.)

[104] Yehuda Sonnenblick. An indoor navigation system for blind individuals. In *Proceedings of the 13th Annual Conference on Technology and Persons with Disabilities (CSUN 98)*, 1998. (Cited on pages 17 and 18.)

[105] Thomas Strothotte, Steffi Fritz, Rainer Michel, Andreas Raab, Helen Petrie, Valerie Johnson, Lars Reichert, and Axel Schalt. Development of dialogue systems for a mobility aid for blind people: initial design and usability testing. In *Proceedings of the second annual ACM conference on Assistive technologies (Assets '96)*, pages 139–144, 1996. DOI:10.1145/228347.228369. (Cited on pages 21 and 26.)

[106] Hong Z. Tan and Alex Pentland. Tactual displays for wearable computing. In *First International Symposium on Wearable Computers (Digest of Papers)*, pages 84–89, 1997. DOI:10.1109/ISWC.1997.629923. (Cited on page 25.)

[107] Sylvie Treuillet, Eric Royer, Thierry Chateau, Michel Dhome, and Jean-Marc Lavest. Body mounted vision system for visually impaired

outdoor and indoor wayfinding assistance. In *Proceedings of the Conference and Workshop on Assistive Technologies for People with Vision and Hearing Impairments: Assistive Technology for All Ages (CVHI-2007)*, 2007. (Cited on pages 18, 21, 22, and 26.)

[108] Koji Tsukada and Michiaki Yasumura. ActiveBelt: Belt-type wearable tactile display for directional navigation. In *Proceedings of the 6th International Conference on Ubiquitous Computing (UbiComp 2004)*, pages 384–399, 2004. DOI:10.1007/978-3-540-30119-6_23. (Cited on pages 25 and 26.)

[109] Iwan Ulrich and Johann Borenstein. The guidecane-applying mobile robot technologies to assist the visually impaired. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 31 (2):131–136, 2001. DOI:10.1109/3468.911370. (Cited on pages 25 and 26.)

[110] Jan B. F. Van Erp, Hendrik A. H. C. Van Veen, Chris Jansen, and Trevor Dobbins. Waypoint navigation with a vibrotactile waist belt. *ACM Transactions on Applied Perception (TAP)*, 2(2):106–117, 2005. DOI:10.1145/1060581.1060585. (Cited on pages 25 and 26.)

[111] Thorsten Völkel. Personalized and adaptive navigation based on multimodal annotation. *ACM SIGACCESS Accessibility and Computing*, (86):4–7, 2006. DOI:10.1145/1196148.1196149. (Cited on page 18.)

[112] Thorsten Völkel and Gerhard Weber. RouteCheckr: Personalized multicriteria routing for mobility impaired pedestrians. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility (Assets '08)*, pages 185–192, 2008. DOI:10.1145/1414471.1414506. (Cited on page 18.)

[113] Bruce N. Walker and Jeffrey Lindsay. Navigation performance with a virtual auditory display: Effects of beacon sound, capture radius, and practice. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 48(2):286–299, 2006. DOI:10.1518/001872006777724507. (Cited on pages 22 and 26.)

[114] Zheshen Wang, Baoxin Li, Terri Hedgpeth, and Teresa Haven. Instant tactile-audio map: enabling access to digital maps for people with visual impairment. In *Proceedings of the 11th international ACM SIGACCESS conference on computers and accessibility (Assets '09)*, pages 43–50, 2009. DOI:10.1145/1639642.1639652. (Cited on page 80.)

[115] Robert Weibel. Map generalization in the context of digital systems. *Cartography and Geographic Information Systems*, 22(4):259–263, 1995. DOI:10.1559/152304095782540285. (Cited on page 43.)

[116] Michele A. Williams, Amy Hurst, and Shaun K. Kane. "Pray before you step out": Describing personal and situational blind navigation behaviors. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '13)*, pages 28:1–28:8, 2013. DOI:10.1145/2513383.2513449. (Cited on pages 4, 5, and 6.)

[117] Scooter Willis and Sumi Helal. RFID information grid for blind navigation and wayfinding. In *Proceedings of the Ninth IEEE International Symposium on Wearable Computers (ISWC '05)*, pages 34–37, 2005. DOI:10.1109/ISWC.2005.46. (Cited on pages 14 and 18.)

[118] Oliver Woodman and Robert Harle. Pedestrian localisation for indoor environments. In *Proceedings of the 10th international conference on Ubiquitous computing (UbiComp '08)*, pages 114–123, 2008. DOI:10.1145/1409635.1409651. (Cited on pages 15, 16, 18, and 64.)

[119] Koji Yatani, Nikola Banovic, and Khai Truong. Spacesense: Representing geographical information to visually impaired people using spatial tactile feedback. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*, pages 415–424, 2012. DOI:10.1145/2207676.2207734. (Cited on pages 20, 21, 24, and 26.)

[120] Limin Zeng and Gerhard Weber. Audio-haptic browser for a geographical information system. In *Proceedings of the 12th International Conference on Computers Helping People with Special Needs (ICCHP 2010)*, pages 466–473, 2010. DOI:10.1007/978-3-642-14100-3_70. (Cited on page 81.)

[121] Limin Zeng and Gerhard Weber. Building augmented you-are-here maps through collaborative annotations for the visually impaired. In *Proceedings of the Workshop on Spatial Knowledge Acquisition with Limited Information Displays*, pages 7–12, 2012. (Cited on pages 24 and 26.)

[122] Limin Zeng, Denise Prescher, and Gerhard Weber. Exploration and avoidance of surrounding obstacles for the visually impaired. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '12)*, pages 111–118, 2012. DOI:10.1145/2384916.2384936. (Cited on page 12.)

[123] Limin Zeng, G. Weber, and U. Baumann. Audio-haptic you-are-here maps on a mobile touch-enabled pin-matrix display. In *Haptic Audio Visual Environments and Games (HAVE), 2012 IEEE International Workshop on*, pages 95–100, 2012. DOI:10.1109/HAVE.2012.6374428. (Cited on pages 24 and 26.)

[124] Limin Zeng, Mei Miao, and Gerhard Weber. Interactive audio-haptic map explorer on a tactile display. *Interacting with Computers*, 2014. DOI:10.1093/iwc/iwu006. (Cited on page 81.)

[125] Haixia Zhao, B. K. Smith, K. Norman, C. Plaisant, and B. Shneiderman. Interactive sonification of choropleth maps. *IEEE Multimedia*, 12(2):26–35, 2005. DOI:10.1109/MMUL.2005.28. (Cited on page 80.)