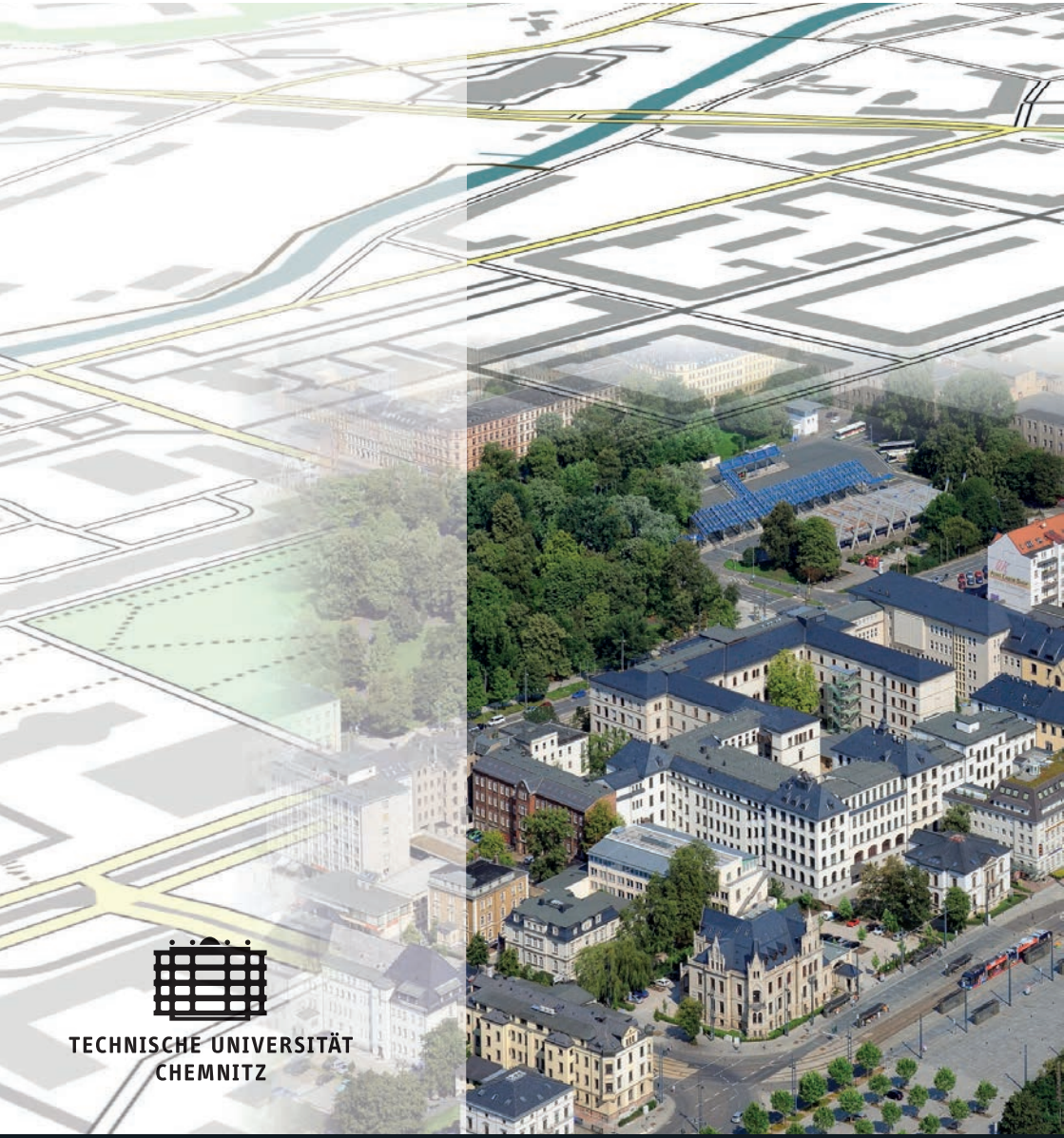
Matthias Werner, Mario Haustein (Hrsg.)

Ortsbezogene Anwendungen und Dienste

9. Fachgespräch der GI/ITG-Fachgruppe Kommunikation und Verteilte Systeme



TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik · www.tu-chemnitz.de/informatik

Prof. Dr. Matthias Werner, Mario Haustein (Hrsg.)

Ortsbezogene Anwendungen und Dienste

- 9. Fachgespräch der GI/ITG-Fachgruppe Kommunikation und Verteilte Systeme -

Prof. Dr. Matthias Werner, Mario Haustein (Hrsg.)

Ortsbezogene Anwendungen und Dienste

9. Fachgespräch der GI/ITG-Fachgruppe Kommunikation
und Verteilte Systeme
13. & 14. September 2012

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Universitätsverlag Chemnitz
2013

Impressum

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Cover
Luftbild: TU Chemnitz/Wolfgang Thieme
Karte: Jens Pönisch, Geodaten (c) OpenStreetMap-Mitwirkende
http://www.openstreetmap.org/copyright
Montage: Mario Haustein, Michael Kunz

Technische Universität Chemnitz/Universitätsbibliothek
Universitätsverlag Chemnitz
09107 Chemnitz
http://www.bibliothek.tu-chemnitz.de/UniVerlag/

Herstellung und Auslieferung
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
http://www.mv-verlag.de

ISBN 978-3-941003-77-4

http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-104609

Tagungsbeiträge

[1] Der Vollartikel zum Vortrag konnte wegen des Todes von Prof. Hagen Höpfner im Oktober 2012 leider nicht fertiggestellt werden.
[2] Zu diesem Vortrag liegt kein Vollartikel vor.

[3] Zu diesem Poster liegt kein Extended Abstract vor.

From single device localization towards mobile network-based route calculation

[1] Gerald Eichler, [2] Roland Schwaiger

Deutsche Telekom AG, Telekom Innovation Laboratories
Internet & Services: Information Relevance
[1] Deutsche-Telekom-Allee 7, D-64295 Darmstadt, Germany
gerald.eichler@telekom.de

[2] Ernst-Reuter-Platz 7, D-10587 Berlin, Germany
roland.schwaiger@telekom.de

Abstract: The contribution analyses and compares several methods for device- and network-centric localization. The paper introduces several approaches for route calculation by schedule-based context enrichment and an efficient solution for ticketing in public transport. Route accuracy is verified by experimental results from a field trial with several smartphone localization techniques. Aggregated, anonymized and vectorized data is a pre-condition for enhanced track and trace solutions and traffic measurements.

Keywords: location based services, metadata enrichment, vector data mapping, track & trace, public transport, mobile ticketing.

1 Localization for mobile ticketing

Modern smartphones support multiple methods for device-centric localization. However, some available options have drawbacks, especially high battery power consumption but also limited indoor access for Global Positioning (GPS)-based localization, the need for provider specific databases for cell-based localization or the need for subscription at a third party for Wireless Local Area Network (WLAN)-based solutions. For the last, the precision and reliability varies within different urban and rural areas dramatically, although co-localization tries to reduce such effects.

Network-centric route calculations (tracing) mostly imply problems with efficiency and privacy, when extending punctual localization to track and trace solutions [EiBo06]. Additionally, it implies to comply with much more regulations as other solutions [WeSo07]. Device-specific localization features are sometimes difficult to access, depending on the vendor and operating system (OS), although *get location* is a simple OS system call. Both approaches offer great advantages, when combining proactive track and trace and reactive geo-fencing.

Route, infrastructure data and schedule mapping are important input factors for traffic measurements and prediction in public transport. With pseudonymization and anonymization, data can be aggregated to traffic vectors. Cell change is an efficient trigger point for device-driven self-localization. A future option for enhanced statistics is provider-controlled signaling-based localization (SBL).

The “Ring&Ride” project is a good example how a user can profit from a Location Based Service (LBS) ticketing solution with a minimum of effort [LME+09]. In its simplest way, tracking is initiated and stopped by freecall without any application installation. Within the project, different localization techniques were compared with the goal of a reliable route calculation for correct tariffing of tickets. To verify the route and identify the vehicle of transport e.g., train (ICE vs. S-Bahn), tram, bus or walking passages, the collected data is aligned with integrated timetables of public transport. Any useful context information increases the data quality [ELR09] and can support the estimation of missing single positions of a track vector.

2 Mobile ticketing approach

Electronic tickets for the use of public transport become more and more important. The huge number of mobile smartphones is accompanied by easy-to-install apps and reliable payment solutions. However, most solutions for the application of electronic tickets are limited either locally or to a dedicated single transport provider e.g., Deutsche Bahn. Offers like “city plus” support local traffic at source and destination, but countrywide unique solutions are still missing. The reason is a lack of interoperable fare management. Public transport vendors are often not fond of interwork because of a fear that their earnings could be too low while others are the winners. In other words, there is still a lot of intransparency for real traffic flows.

When thinking of common solutions, three parts of an interactive process chain need to be unified, in order to add comfort for the traveler (customer), see Figure 1:

1. **Point of sales:** the front-office, respectively the way, where customer get their contracts and tickets
2. **Check-in and check-out:** the method how customers indicate the staring point and end of their travel
3. **Post processing:** the back-office, where fare calculation, billing, statistics and cost sharing is carried out

All three parts contain critical points which need flexible solutions to increase the acceptance rates for both, customers and public transport providers. Customers do not like long term or complex contracts, want to be very flexible, prefer different ways to get a ticket, and aim for a certain amount of privacy. Public transport providers look for efficiency in use of their vehicle capacities, a minimum of technical infrastructure for ticketing to be maintained and transparent statistics. Last but not least, a lean backoffice is required.

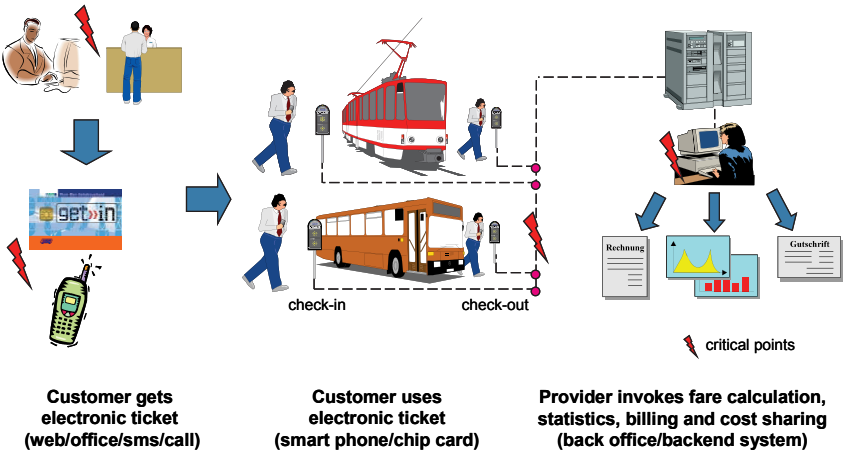


Figure 1: Integrated mobile ticketing solution for interoperable fair management

2.1 Use case and role model

The general **use case** is fairly simple:

A customer C wants to travel ad-hoc from A to B on Route R with a single ticket T using vehicles $V_1, V_2, \ldots, V_n$ provided by public transport providers $P_1, P_2, \ldots, P_n$.

To reach this goal, a clear role distribution of the involved parties is required. At least one party needs to be the service owner, being in charge as legal unit and subcontracting all required sub-services, see Figure 2. There is no need to establish a new company. It is preferred to select an established "Verkehrsverbund", which is best qualified for this role.

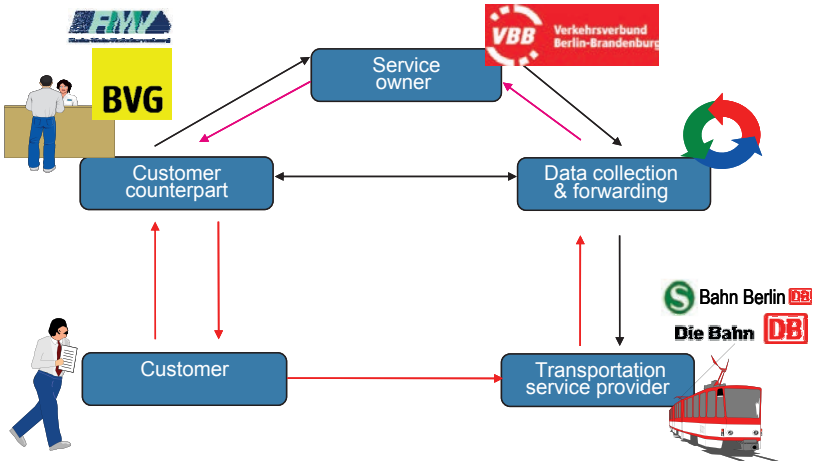


Figure 2: Role model for interoperable fair management

2.2 Tracing and additional information sources

Smartphones offer multiple options for localization. Mainly known are GPS tracking, mobile network cell ID tracking or WLAN-based tracking, relying on SSID data bases. A trace is defined as a list of single locations, described by longitude and latitude. In practice, all these methods result into incomplete or imprecize traces between check- and check-out. GPS coverage is limited indoors, like in big stations, while cell density and WLAN is low in rural areas.

However, there is no need for exact traces as there are additional sources for reliable route calculation.

1. **Spatial correlation:** for all public transport systems, the regular routes are predefined as trains, trams or buses have dedicated routes.

2. **Chronological correlation:** transport systems are based on known schedules. Furthermore, real-time data can significant increase this type of correlation.

By combining customer location data (recorded traces), public network infrastructure data (vehicle stops and predefined tracks) and timetable data (schedule and delay), a high degree of correct route calculation can be reached. As a result, origin and destination of the travel as well as the used vehicles per section can be determined, see Figure 3.

Timetable data e.g., were imported from the HaCon Fahrplan-Auskunfts-System (Hafas)[1] database provided by Deutsche Bahn (DB). The database contains nearly all timetables of public transport in Germany, Austria and Switzerland as well as real-time information.

[1] Hannover Consulting (HaCon), URL: http://www.hacon.de/hafas/

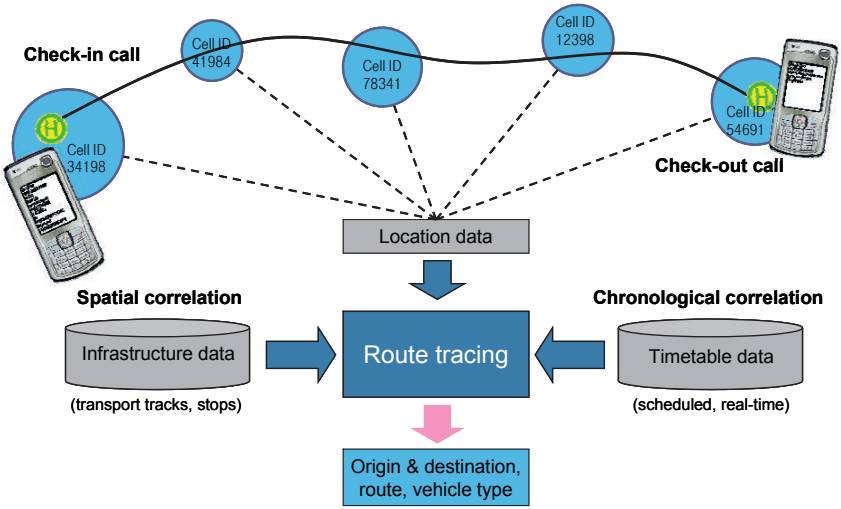


Figure 3: Additional data sources for reliable route calculation and vehicle determination

3 The Ring&Ride solution

The Ring&Ride (R&R)[2] consortium proposes a complete architecture which was evaluated within two field trial phases. A special focus was set on a comparison of different localization technologies.

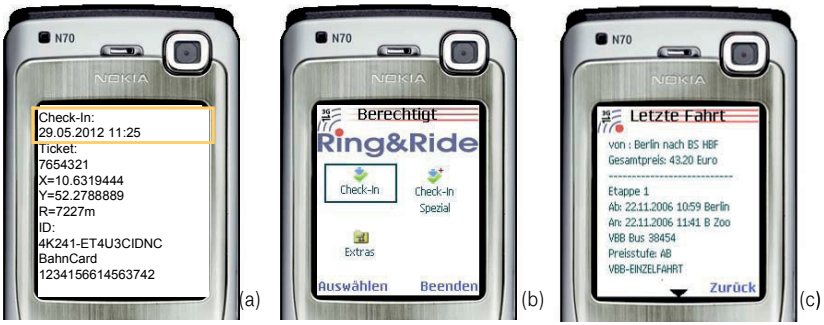


Figure 4: (a) R&R check-in SMS for trial purposes, (b) R&R mobile phone application, (c) R&R application confirmation with prize, route calculation and tariff optimization

[2] (R&R) was funded by the German Federal Ministry for Education and Research (BMBF) and supervised by German Federal Ministry for Economy and Technology (BMWT), Förderkennzeichen 19 P 5026.

In its simplest way, check-in and check-out can be proceeded by a 0-800-freecall number, without any additional client software, which makes the use case applicable for any type of mobile phone. Responses are sent as SMS or MMS (QR-code). A native phone application offers more usability, see Figure 4(b) and (c).

A trip is characterized by five interaction steps:

1. **Start of trip:** 0800-RINGRIDE freecall for check-in
2. **Electronic ticket:** receive ticket by SMS or MMS. The ticket information includes date and time, unique ticket ID and secure signature, location and cell radius (test phase), optional: discounts e.g., BahnCard.
3. **Travel:** with optinal chance of ticket verification by a conducter
4. **End of trip:** 0800-RINGRIDE freecall for check-out
5. **Billing:** receive confirmation SMS or MMS: duration and route, tariff and price

The minimum solution is based on localization information at step 1 and 4, which can be provided by the mobile network operator (MNO) by identifying the cell coordinates of the SMS or call origin. This information is delivered as longitude (x), latitude (y) and a predicted cell radius (r), see Figure 4(a). In addition, each localization event gets a timestamp t. For normal operation, a data connection is required to trigger or transfer location information to collect intermediate location tuples (x, y, r, t).

A route R is defined as a vector of locations: $Rt := \{(x_1, y_1, r_1, t_1), (x_2, y_2, r_2, t_2), \ldots, (x_n, y_n, r_n, t_n)\}$

3.1 Architecture

The R&R architecture is based on a requirement analyses for public transport solutions [MLE07], described in a use case model, using UML2[3].

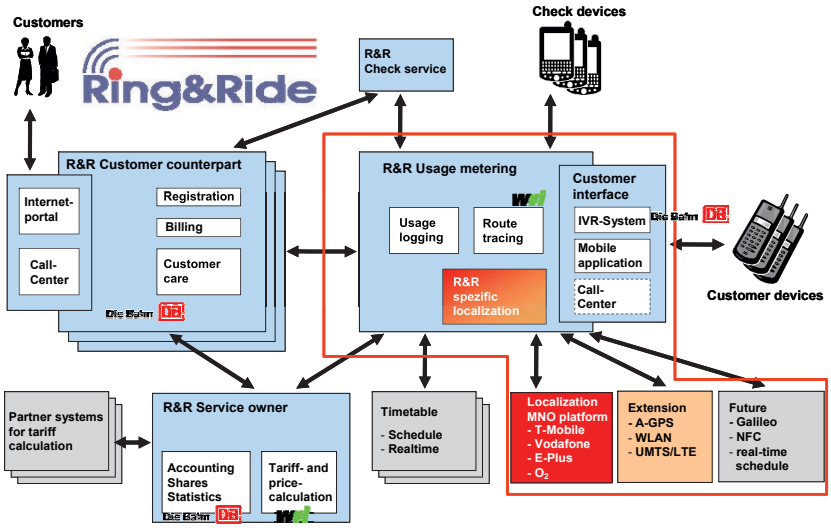


Figure 5: Ring&Ride architecture, including localization and privacy support

[3] Object Management Group (OMG): URL: http://www.omg.org/spec/UML/

In Ring&Ride a modular structure was chosen (Figure 5.) The interaction of distributed components was based on SOAP web services, exchanging XML messages. The backend is programmed in Java 1.5.0. Main logical components are:

- Registration/customer care/billig (Anmeldung/Kundenbetreuung/Rechnungserstellung)
- Customer interface (Kundenschnittstelle)
- Usage logging (Leistungserfassung)
- Localization (Ortung)
- Persistence and permissions (KOSE)
- Check (Kontrolle)
- Tracing (Routenermittlung)
- Tariff and pricing (Tarif- und Preisermittlung)
- Accounting and statistics (Abrechnung und Erlösmanagement)

The data model follows the European standard EN 1545[4] [LST08+] It defines all data elements and the coding used in the application layer, e.g. stops, vehicle types, route descriptions, payment methods, etc. R&R complies with this standard, although it turned out that Ring & Ride uses some data elements that the standard does not contain, e.g., mobile telephone numbers (MSISDN) or geo-coordinates.

3.2 Privacy and localization

The application of LBS for track and trace of individuals is a severe privacy issue [EiBo06] where data protection and telecommunication laws are touched. Therefore, potential customers need to accept personal tracing and the service owner has clearly to declare, how and what for collected data are used.

All four primary German MNOs – T-Mobile, Vodafone, E-Plus and O_2 – were involved for the test setting. To gain a unified interface for cell-ID-based localization, the Permission and Privacy Gateway (PPGW) as trusted localization party service was used [EP11].

Since 2006, PPGW is used for various localization platforms, either SMS-services (T-Info or Gelbe Seiten) or mobile applications e.g., Qiro[5]. PPGW works MSISDN-based and returns the current position at the time of request, based on MNOs' cell information.

A single localization is resolved by the PPGW: (x, y, r) = f (MSISDN, t)

4 Localization methods

R&R focuses on testing multiple localization methods for route tracing in the given context. As external references, GPS, WLAN base stations and base stations of the MNO were selected for evaluation

GPS was used as a basic acurate reference, whereas available. Assisted GPS (A-GPS) is taken into consideration for futher studies to decrease the time to first (ratherly precise) log. WLAN was especially applied at large train stations. Both, GPS and WLAN are power consuming and therefore inefficient for long travels. Refer to Figure 6 for further advantages and disadvantages of the applied methods.

Base stations of MNO are characterized by their cell IDs. In the applied model a cell is characterized by a circle with a center C (x, y) and its radius r, provided by the MNO. The higher the density of the urban and therefore the transport infrastructure the lower is r. Especially pico cells can help to trace precise routes.

[4] EN 1545: Data elements for public transport
[5] URL: http://www.deutsche-startups.de/2007/08/16/qiro-lokalisiert-freunde/

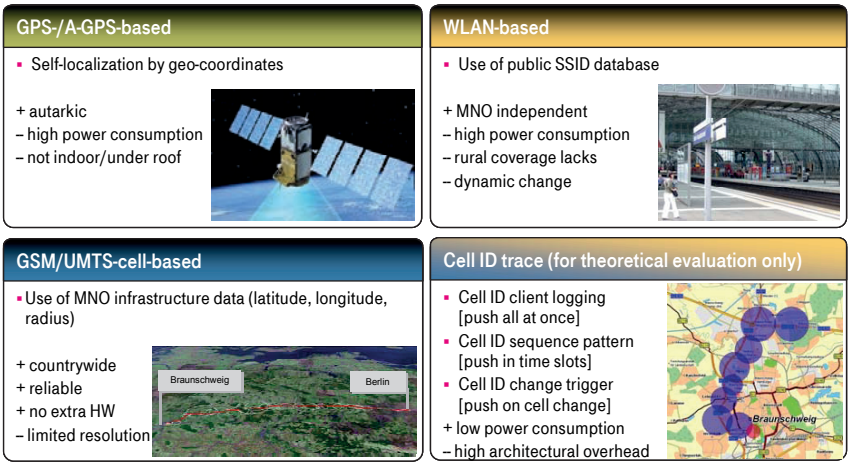


Figure 6: Applied localization methods for the Ring&Ride field trial

4.1 Device vs. network centric localization

The methods for localizations cover a wide range from autarkic device-centric localization, including movement sensors, up to plain MNO-centric localization. While auto-localization protects privacy very well, it is not save enough against user manipulation. Practical well-established solutions e.g., interaction with NFC tags (QR and RFID) or recorded GPS location data can be faked at the end device.

Cell-based methods were analyzed on different collection and trigger base. Cell IDs could be either collected on the mobile device (client side) or by the MNO (provider side). Futhermore, localization can be triggered time-driven e.g., in regular time slices or event-driven e.g., on the handover event. The last can reduce the effort of localization costs.

Most localization solutions rely on a third party, see Figure 7. It could be either

1. **Operator-centric*:*** by a MNO, using its infrastructure data base (lower left) or

2. **Web-centric:** by a service provider, offering a client-server solution

Today, most web-centric solutions work country-wide, but rely on collected data in public or private databases, like IP addresses, WLAN SSID co-localisation e.g. Google Lattitude[6] or SSID footprint models, e.g. awiloc[7] by Fraunhofer IIS. Footprint models have two advantages

1. **Preciseness:** models are more precise than single localization refernces.

2. **Robustness:** models are robust against temporal changes.

3. **Privacy:** no personal identifier need to be transferred for localization e.g., MSISDN, assigned IP address

[6] Google Lattitude: URL: www.google.de/latitude/
[7] Fraunhofer IIS: URL: http://www.iis.fraunhofer.de/de/bf/ln/technologie/rssi/tuw.html

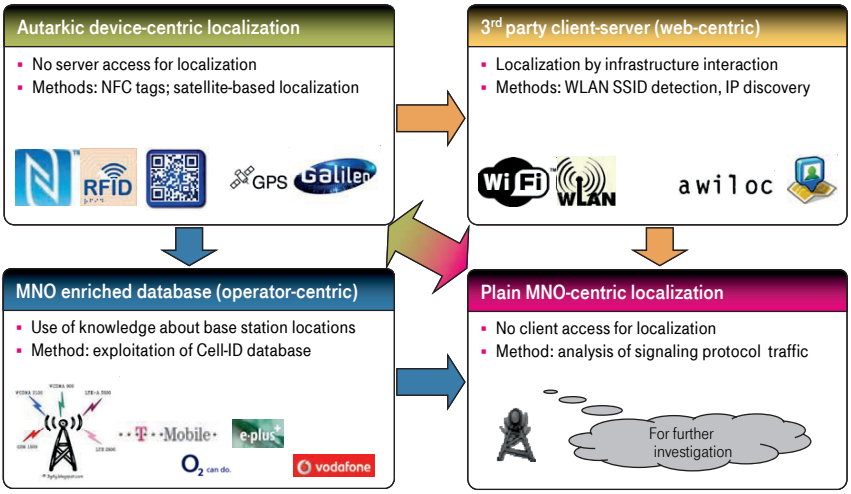


Figure 7: Device-centric vs. mobile-network-centric localization

4.2 Ring&Ride field trial focus and settings

The R&R trials were focused on two key issues:

1. Accuracy of route tracing (localization) and pricing (tariffing)
2. Acceptance and usage of the application (frontend) and system (backend)

Both, local public transport in Berlin and long distance travel between Braunschweig and Berlin were evaluated within the two trial phases for different localization methods [Wer+10]. Pre-scheduled trips were done by multiple testers, carrying multiple mobile devices each.

For the local public transport 637 test rides could be analyzed. All four German MNO were involved for the cell-ID-based localization:

- 207 rides with T-Mobile localization,
- 152 rides with Vodafone localization,
- 138 rides with E-Plus localization and
- 140 rides with O_2 localization.

There was an average riding time of 20 minutes, containing one change of vehicle per ride in average. Therefore rides consist of multiple sections. The distribution of vehicles per section shows:

- 472 rides with city train (S-Bahn),
- 220 rides with bus and metro bus,
- 153 rides with underground (U-Bahn) and
- 54 rides with metro tram.

4.3 Ring&Ride field trial results

Route exactness and price correctness were the two most important parameters for evaluation of the entire framework as there are many components involved.

Localization method		Mobile-network-based	Satellite-based	Infrastructure-based
		GSM/UMTS	GPS	WLAN
Route exactness	Local travel (Berlin)	67 % (267 trips)	84 %	72 % (24 trips)
	Long distance travel	89 % (57 trips)	(mixed travel)	n/a
Price correctness	Local travel (Berlin)	99 %	100 %	100 %
	Long distance travel	91 %	(mixed travel)	n/a

Table 1: Route exactness and price correctness of test trips

Following field trial criteria 1, a high price correctness was achieved for all three applied localization methods for the local travel, see Table 1. A more detailed evaluation was done on the route correctness, see Table 1. A combination of GPS and GSM/UMTS increases this value as depicted in Figure 8.

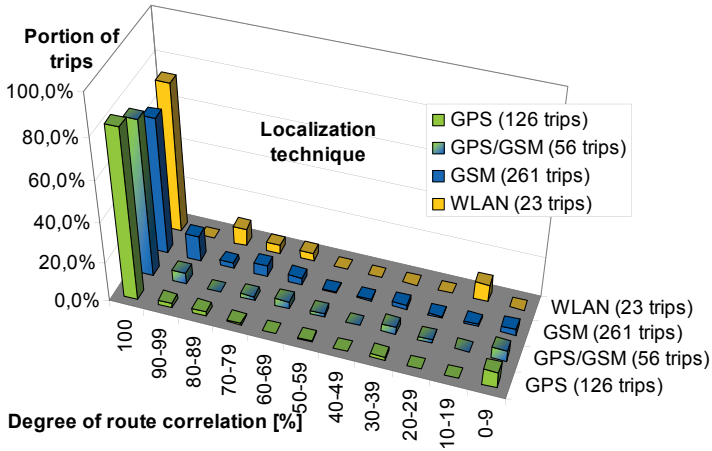


Figure 8: Ring&Ride field trial: route calculation correctness

The degree of route correlation was defined as a set of certain criteria to be fulfilled. Per section of a travel, correct starting and ending point, selected vehicle and line are double-checked against the written test protocols. Local travel is less critical for price calculation, as tickets are typical sold in zone granulation. In Berlin the basic ticket set is limited to short trip, zone AB, zone BC and zone ABC.

5 Solution evaluation

5.1 Comparison of localization approaches

There is no unique localization method which always provides the best results.

Satellite-based: as expected, GPS provides the highest accuracy, when available. But when activating GPS at the beginning of travel, it often takes too long to get the first position, especially when the mobile device is already in motion. GPS is available mostly for surface travel, however, the combination of station and vehicle roofs on rainy days results into problems. The same holds for ICE trains (copper covered windows) in combination with overhead contact installations for the traction system.

Wireless-network-referenced: although WLAN infrastructure is not a reliable reference source due to unpredictable down times, it offers a rather exact option to gain start and end points of travel in urban areas. Available databases are rich due to co-localization methods.

Mobile-network-based: for MNO-based localization for different approaches were taken into consideration. Cell Ids could be either tracked on the client side or on the provider side. Cell change trigger is an interactive solution where the provider is asked by push messages to locate the mobile device only on a cell handover event.

Localization method	Mobile-network-based (Cell ID)				Satellite-based		Infrastr.-based
	GSM-MNO-based	Cell-ID client logging	Cell-ID sequence pattern	Cell-ID change trigger	**GPS**	A-GPS	**WLAN**
Value for Ring&Ride	+	+	+	+	+	+	+
Technical critera	+ +	-	-	+	0	0	-
Economic criteria	0	0	-	+	+	- -[8]	0
Political criteria	+ +	+	-	+	+ +	+	+
Availability	+ +	+ +	+	+ +	-	0	-
Accuracy	0	0	0	0	+ +	+ +	+

Table 2: Comparison of localization technology application

Beside the technical evaluation, the R&R project analyzed several side effects by using different localization approaches. Table 2 summarizes results from the final report [Wer+10].

For the Cell ID approaches four ideas were compared. In addition to the approaches described in section 4.1 a pattern sequence matching was taken into consideration. Therefore a traced routes R_t is compared with all routes of a library of routes $\{R_l\}$. From the data protection point of view the long term storage of collected routes was seen critical.

5.2 Comparison of mobile ticketing approaches

For comparing of the R&R solution with other established systems, six criteria were selected, see Figure 9. All criteria were defined that way that valuable scores are represented by the outer values in the graphical representation.

[8] A-GPS is rated that low in case of economic criteria, when taking additional infrastructure as local terrestric reference trnasmitters into consideration.

In general, there are two types of solutions: either those, which only have a check-in step and need a pre-definition of the destination and those, which support flexible traveling by asking for a dedicated check-out step. From the user point of view, it still remains unclear, which one is a real advantage, although questionnaires emphasized a slide preference for check-in only. Especially in the beginning customers tend to forget the check-out.

Solutions with physical check points require additional installation effort, either at the stations e.g., Touch & Travel or Oslo smart card or within the vehicles e.g., Helsinki smart card. Smart cards are a cheep and reliable system, but require additional effort for the payment, while startphone extensions e.g., Touch & Travel make use of mobile data connection and MNO payment support e.g., as mobile wallet or via the regular phone bill.

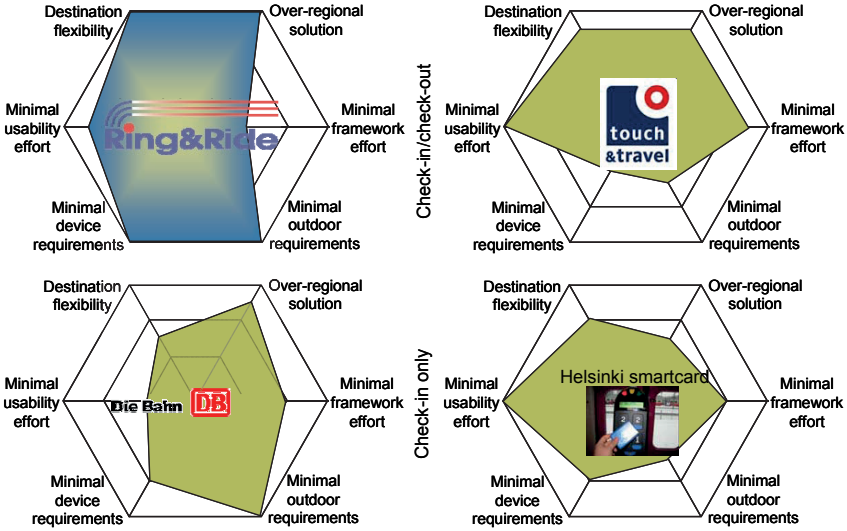


Figure 9: Mobile ticketing solution comparison: Ring&Ride, Touch & Travel NFC smart phone solution, Deutsche Bahn mobile ticketing, Helsinki RFID smart card

The complexity of the R&R implemented framework (backend) is seen as a serious drawback. However, except from this R&R gains high scores. Remaining the question: although the project is finished and nearly no decentralized infrastructural invest would be required, why is Ring&Ride not available for everyone? There are several main issues:

1. **Interworking reason:** exploitation the R&R advantages multiple regional transport providers and the main over-regional transport provider – Deutsche Bahn (DB) – are required as sharing partners. DB decided to invest in Touch & Travel. Unfortunately, the number of customers is still very low, as special NFC-enabled smartphones are required for the check-in and check-out touchpoints.

2. **Economical reason:** provider-supported tracing is still too expensive, as MNOs mostly charge on a per localization base.

3. **Technical reason:** GPS and WLAN alternatives are too power consuming for today's chipsets. Therefore, customers do not yet accept to turn on these network features during the entire time of travel.

5.3 Lessons learned

Following the project goals, there is a separate analyses of pros and cons for frontend and backend view, see Figure 10. The listing of drawbacks from the user view contains already some proposed solutions.

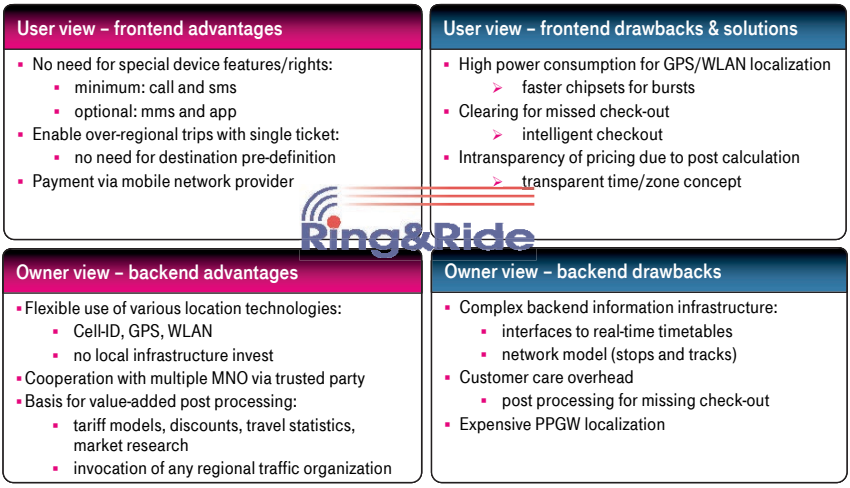


Figure 10: Ring&Ride evaluation: frontend and backend advantages and disadvantages

The most important requirement is to deal with missed check-outs by the customer. There are to criteria which might be an indication that a traveler has finished it trip:

1. **No movement anymore:** the location of the customer stays the same over a long time.

2. **Movement out of correlated, predefined tracks:** the customer leaves areas covered by public transport.

In both cases the costumer might be notified by a reminder SMS: “Did you finish your travel?”. Answering with “Yes”, the route will be closed at the last correlated location.

5.4 The future: signaling-based localization

Client interaction and single localization require a huge amount of effort. Therefore, it is a meaningful approach to move towards plain signaling-based localization (SBL) in mobile networks. Signalling contains mature information about cells. Handovers indicated by location update (LUP) messages can be monitored and taken for route calculation. In this case the user interaction is again reduced to the effort of the simple phone solution by indicating the start and end of travel, meaning start tracking me and stop tracing me. No additional data transfer is required, except delivering a valid ticket.

Furthermore, SBL offers easily more aggregated statistics, either for

1. **Dedicated locations:** as point bypass counts

2. **Dedicated routes:** as traffic statistics

The SBL approach is not limited to public transport. Flow Car Data (FCD) trajectory is known as an SBL approach in the telematic domain [FJO+08].

Furthermore, SBL offers an enabler for a wide range of new B2B applications. Refer to Figure 11 for examples. Aggregation, pseudonymization and k-anonymization[9] will successfully help to overcome arising data protection and privacy issues, often discussed in public. Customers' smartphones need not to be touched.

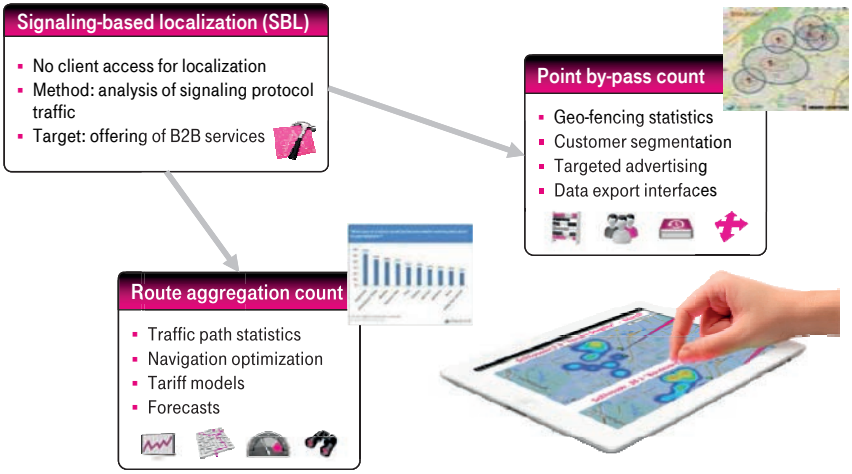


Figure 11: Signaling-based localization – a vision for future MNO-centric localization

Acknowledgement

The Ring&Ride project was a joint collaboration of Berliner Verkehrsbetriebe Anstalt des öffentlichen Rechts (BVG), Deutsche Bahn (DB), Deutsche Telekom AG (DTAG), Oecon Ingenieurgesellschaft für Industrieberatung und Projektmanagement mbH (Oecon), Rhein-Main-Verkehrsverbund GmbH (RMV), S-Bahn Berlin GmbH and WVI Prof.-Dr. Wermuth Verkehrsforschung und Infrastrukturplanung GmbH (WVI), coordinated by Technische Universität Braunschweig, Institut für Verkehr und Stadtbauwesen. Many thanks for the fruitful corporation.

List of abbreviations

Abbreviation	Full text
A-GPS	Assisted Global Positioning System (using additional references)
B2B	Business to Business (business model)
BMBF	Bundesministerium für Bildung und Forschung (Germany)
BMWT	Bundesministerium für Wirtschaft und Technologie (Germany)
DB	Deutsche Bahn
EN	European Norm
EP	European Patent
FCD	Floating Car Data

[9] aggregation to a minimum cluster size of k

GPS	Global Positioning System
GSM	Global System for Mobile Communication (mobile network)
Hafas	Hannover Consulting Fahrplan-Auskunfts-System
LBS	Location Based Service
LUP	Location Update
MNO	Mobile Network Operator
MSISDN	Mobile Subscriber Integrated Services Digital Network Number
NFC	Near Field Communication
OMG	Object Management Group
OS	Operating System
PPGW	Permission and Privacy Gateway (service)
QR	Quick Response (2D graphical code)
R&R	Ring&Ride
RFID	Radio Frequency Identifier
SBL	Signaling Based Localization
SSID	Service Set Identifier
UML	Unified Modeling Language
UMTS	Unified Mobile Telecommunication System (mobile network)
WLAN	Wireless Local Area Network

References

[EiBo06] Gerald Eichler, Andreas Böhm: Einheitliche Einzellokalisierung – Voraussetzung für Track & Trace Mehrwertdienste. In: Praxis in der Kommunikationstechnik (PIK), Nr. 29 (1/2006). K.G. Saur Verlag, München, Januar 2006.

[EP11] European Patent Office: Verfahren und Kommunikationssystem zum Bereitstellen von aufenthaltsortsbezogenen Diensten, EP 1 725 053 A3, Patentblatt 2011/13, March 2011.

[ELR09] Gerald Eichler, Karl-Heinz Lüke, Britta Reufenheuser: Context information as enhancement for mobile solutions and services. In proceedings of the 13th International Conference Intelligent Networks (ICIN 2009), Bordeaux, France, October 2009.

[FJO+08] Markus Friedrich, Prokop Jehlicka, Thomas Otterstätter, Jonas Schlaich: Mobile Phone Data for Telematic Applications. In: proceedings of International Multi-Conference on Engineering and Technological Innovation (IMETI) 2008, Orlando, Florida, U.S.A., June 2008.

[LME+09] Karl-Heinz Lüke, Holger Mügge, Matthias Eisemann, Anke Telschow: Integrated Solutions and Services in Public Transport on Mobile Devices. In proceedings of the 9th International Conference on Innovative Internet Community Services (I^2CS 2009), Jena, Germany, June 2009.

[LST+08] Karl-Heinz Lüke, Wolfgang Schlüter, Anke Telschow, Carsten Sommer, Oliver Bley, Manfred Wermuth: Location-Based Mobile Ticketing with Electronic Fare Management. In proceedings of the Intelligent Public Transport Systems (IPTS) Conference, Amsterdam, The Netherlands, April 2008.

[MLE07] Holger Mügge, Karl-Heinz Lüke, Matthias Eisemann: Potentials and Requirements of Mobile Ubiquitous Computing for Public Transport. In proceedings of the INFORMATIK 2007 (GI Jahrestagung), pp. 237-246, Bremen, Germany, September 2007.

[Wer+10] Manfred Wermuth et al.: Multifunktionales Handy-Ticketing – Schlussbericht des Forschungsprojektes. BMWT, Förderkennzeichen 19 P 5026, Braunschweig, Germany August 2010.

[WeSo07] Manfred Wermuth, Carsten Sommer: Neuere Entwicklungen beim Handy-Ticketing. In: Der Nahverkehr 7-8/2007, pp. 51-55, Alba Fachverlag GmbH Co. KG, Düsseldorf, Germany, Juli 2007.

Towards Sensor-supported Indoor Localization Using Cloud-based Machine Learning Techniques

Dirk Bade, Daniel Gleim

Distributed Systems Group
University of Hamburg
Vogt-Koelln-Strasse 30
22527 Hamburg
dirk.bade@informatik.uni-hamburg.de
8gleim@informatik.uni-hamburg.de

Abstract: Determining one's own location is crucial for using location-based services. For this purpose, several localization techniques have already been proposed throughout the years. But still, no consensus about the most appropriate technique with respect to infrastructure, resource and runtime requirements is reached. As a consequence, the setup of a localization system for indoor environments requires a lot of knowledge and experience which is among the main reasons for such systems not being ubiquitously available. This paper presents an indoor localization approach which can easily be set up by everyone as it neither requires technical nor environmental knowledge. It is based on a localization library for mobile phones and a cloud-based machine learning service for scene analysis. Although this combination seems promising, a first evaluation shows that this approach failed to live up to expectations.

1 Introduction

Nowadays, mobile phones are accompanying us everywhere. Being well equipped with powerful hardware and communication capabilities, several sensors and sophisticated applications, they have already become personal assistants in our everyday live. And due to the ability to sense their context, especially the current location, their functionality can be even more enhanced by accessing context-based services, which provide appropriate information at the right place and time. Nevertheless, determining the current location is still challenging. Being outdoor, one normally uses GPS to do so which is quite accurate, but energy exhausting. An affordable alternative is to use cell towers or Wifi access points. While the former is only useful for coarse-grained localization, the latter requires extensive information about reference points [Küp05].

As a consequence, setting up a localization system from scratch requires a lot of knowledge and experience in order to choose the best technique and an appropriate localization algorithm. Moreover, a preparation phase is mandatory to gather information about available access points and their signal propagation schemes. This information is afterwards used by localization algorithms to determine a client's location. Executing such algorithms

locally on the client's device requires all such information to be stored on the device and to be updated regularly. As an alternative, a supporting infrastructure could be maintained which not only manages the reference information, but also executes the localization algorithms. In this case, clients just need to send a localization request containing locally sensed signal strength data to a server which in turn responds with the client's location or which uses the data to provide a location-based service.

This question of where to put the functionality is also known as the *mobile dilemma* [Fuc05]. As reference information can be quite voluminous and localization algorithms often have a high runtime complexity, it is argued that an infrastructure-based approach is preferable. But this yields another problem: scalability. Depending on the extent of the area (i.e. the number of access points) and the number of clients, the infrastructure may temporarily (e.g. at daytime) be faced with high loads. Hence the supporting infrastructure needs to scale quite well, ideally dynamically depending on the current needs.

Summarized, setting up an (indoor) localization infrastructure is not a trivial task and requires a lot of knowledge, experience and resources. Several companies (e.g. *Google*, *Microsoft*, *Skyhook*, etc.) already started to offer appropriate solutions, but only recently these companies also focused on localization within (public) buildings like shopping malls, airports, or libraries. Following the example of *Open Street Map*[1], our vision is to let the community be responsible for collecting all required reference information for localization within indoor areas. By using public cloud infrastructures to process this reference information as well as localization requests, the approach allows to set up a distributed, self-determined and easy-to-use localization infrastructure. Moreover, the approach abstracts from specific localization algorithms and their suitability for only certain (indoor) environments. Following a data-centric view, we make use of ensemble machine learning techniques using multiple classifiers to choose an algorithm which best fits the training data. Our solution contrasts other approaches in that it is based on a machine learning blackbox residing in a public cloud infrastructure, hence minimizing the effort and cost to set up and maintain a localization infrastructure for everyone. Moreover, we present a prototypical implementation of a localization library for *Android* devices used as the basis for two applications: one for collecting necessary reference information and to train the algorithms residing in the cloud and the other application for navigating in indoor environments which is used in our evaluation.

The rest of the paper is structured as follows: Section 2 presents basic fundamentals about localization techniques, machine learning and cloud infrastructures. In Section 3 a set of requirements is inferred from an application scenario. Based on these, Section 4 introduces our system design, whose prototypical implementation is presented in Section 5. The results of our evaluation that has been conducted using the prototype are highlighted in Section 6. Finally, Section 7 concludes the paper and gives our prospects for future work.

[1] http://wiki.openstreetmap.org/wiki/About (18. Sept. 2012)

2 Fundamentals

This section will detail the fundamentals necessary to understand the rest of this paper. Section 2.1 covers indoor localization, machine learning techniques used for localization are surveyed in Section 2.2 and cloud infrastructure essentials are presented in Section 2.3.

2.1 Indoor Localization

Throughout the last years, several localization techniques have come up in academia as well as in commercial products. Most of these rely on an arbitrary signal emitter, a corresponding receiver, some kind of algorithm to calculate the current location and some approaches also rely on additional infrastructure components. Existing localization principles fall into one of three different classes [Küp05, LDBL07, HB01]:

Triangulation This class of algorithms makes use of either lateration or angulation techniques. Lateration tries to estimate the current location by measuring the distance to a set of reference points. For this purpose, either the time-of-flight or the attenuation of a signal's strength is used as an indicator for the distance. Angulation in contrast makes use of the angle in which signals are received from two or more reference points to estimate the current location.

Scene Analysis A scene is uniquely represented by a set of features (e.g. physical properties such as the signal strength of radio waves emitted by a number of senders) called a *fingerprint*. By observing these properties at a certain location and comparing these with a reference set, an estimation of the current location is possible.

Proximity By sensing a unique feature of the environment, one can conclude to be near a certain location. For example, when receiving a cellular network base station beacon, one can infer the coarse relative location.

Dead Reckoning Using only locally available information such as orientation and speed dead reckoning techniques try to estimate the current location relative to the last known location.

All of these techniques differ with respect to certain performance metrics [LDBL07]: accuracy, precision, complexity, robustness, scalability, and cost. Therefore, it depends on the application domain which technique is best used under given conditions. Because of the ubiquitous infrastructure the most popular techniques for localization are nowadays based on GPS satellites, GSM/CDMA cellular networks and Wifi base stations. But when it comes to indoor localization, GPS cannot be used because a direct line of sight with the satellites is required and cellular networks only allow for coarse-grained location estimation [Küp05]. As a consequence, most approaches to indoor localization rely on the Wifi technology. But there is no common consensus about which localization principle and algorithms are best suited. Because lateration and angulation techniques are influenced by

signal attenuation and multipath effects [BB05, PL05], fingerprinting is often used especially for indoor scenarios. In the literature, several fingerprinting algorithms are proposed (see e.g. [LDBL07] for a survey). Some of which rely on an explicit derivation of the functional relationship between the actual position and the raw measurements. But due to the complexity of radio wave propagation schemes, especially indoor, such a relationship is difficult to formulate.

Brunato and Battiti argue that statistic/probabilistic methods should perform best, as these try to derive the unknown functional dependency between the actual position and the sensed features of a scene (i.e. received signal strength from several different access points) [BB05]. Moreover, such methods avoid modelling the signal propagation, which is quite difficult especially in indoor scenarios where propagation suffers from multipath effects and shadowing [LDBL07]. For the statistic/probabilistic methods to be used in scene analysis a set of reference measurements has to be collected which is afterwards used to generalize so called *fingerprints*, i.e. to find patterns in the concrete reference samples which are later compared with actual localization measurements. In the following, the basic principle of these methods is roughly sketched.

2.2 Machine Learning Techniques

Machine learning includes a number of statistic/probabilistic methods used to classify samples with dependent and independent variables. The main motivation is that data sets may be quite large and the interrelationship of variables is a priori unknown, so that it is difficult to find a (complex) pattern within samples of the data set. Hence, the initial task is to create a generalization, i.e. to identify a set of classes, in which all members have something in common. Such a generalization is also called a *classification model* and may afterwards be used to classify a previously unseen sample [Mar09].

Machine learning, in particular classification, can also be used for localization [RMT+02]. In this case, the classification model holds a set of classes based on reference signal strength indicators (RSSI) which have been recorded to train the algorithms. The task would be to classify the user's current location on the basis of all currently received RSSI. A location's class would be a label that uniquely represents the location. For this to work, a radio map is initially required. Such a map represents a set of reference measurements (samples) at certain points in the area where localization shall take place later on [HPALP09]. A sample is an $(n+1)$-dimensional vector containing n attributes representing the RSSI values and one label for the location. This reference radio map is then used to train an algorithm in order to build a classification model. After this model is trained it is able to classify an unseen sample, i.e. to name the location where this sample has been taken.

A multitude of different machine learning algorithms which can be used for location learning and localization exists. In the literature, some of the most prominent methods are *(Weighted-)k-Nearest-Neighbours*, *Support Vector Machines*, *Bayesian Modelling*, *Decision Trees*, *Neural Networks*, etc. It is out of the scope of this paper to detail these meth-

ods, but surveys and further references can be found e.g. in [Mar09, MD07, AAH+09, LDBL07, BB05]. Moreover, in order to enhance localization accuracy, additional filter algorithms, e.g. *Bayes* or *Kalman* filters which take the last measurements into account may also be applied [HPALP09].

Given the multitude of algorithms it is difficult to choose one which is appropriate for a given localization scenario. Therefore, so called *ensemble learning* algorithms are suggested, which combine a number of classification models to achieve more accurate results [AAH+09]. This approach will be further explained in Section 4.2.

2.3 Cloud Computing

In order to offer some kind of service one has to provide appropriate computing, storage, and networking resources. Traditionally, a server had to be deployed somewhere in the backend serving the requests of clients. With increasing demands new hardware or additional machines were manually added, whereas attention had to be paid to even be prepared for temporary peak loads. The cloud computing paradigm tries to ease the issue of dynamic resource requirements by abstracting from the real infrastructure. Using virtualization techniques a server may easily be up- and downgraded and additional servers may be deployed right away. This way, the infrastructure can instantly be adapted to the current needs.

One has to distinguish between three different models of computing clouds [BBG11]:

IaaS *Infrastructure as a Service* allows to administrate the infrastructure at the hardware level. It is up to the user to deploy necessary applications and services as well as to start, stop, upgrade or downgrade server instances.

PaaS With *Platform as a Service* the user only provides the applications and services while the infrastructure is automatically managed.

SaaS *Software as a Service* is the most abstract view. In this case, a complete application or software suite is offered by the cloud and the user cannot influence the underlying infrastructure at all.

Additionally, one has to distinguish between *private* and *public* clouds. While the former is used within organizations to abstract from their own IT-infrastructure, the latter is open to everyone. Several companies already offer any of the above cloud models, further details can be found e.g. in [BBG11].

In the next section, the requirements for localization using cloud-based machine learning techniques are further detailed.

3 Requirements Analysis

Before our system design is finally presented in Section 4 a requirements analysis based on an application scenario is conducted in this section.

Application Scenario In a university's library it is often difficult to find the books or magazines one is looking for. Several reading rooms equipped with hundreds of shelves full of books hinder orientation significantly. The library already introduced a sophisticated schema for book signatures, labelled all shelves, and implemented a Web-based search, but still students get lost. As a consequence, the librarians decided to set up a localization system that allows students to navigate to the books they are looking for using their mobile phone.

For this purpose, reference information about available Wifi access points as well as signal propagation has to be collected. Moreover, a server on which the localization algorithms can be trained and which takes localization requests from the students and guests is required. But none of the librarians is familiar with localization techniques and the university's network infrastructure hinders mobile devices from accessing private network services.

Inferred Requirements In order to create a radio map of the area of interest reference information needs to be collected. For this purpose, someone needs to take signal strength samples at certain positions and label the locations. An appropriate application running on a mobile device to support the user performing this task should serve this purpose best.

The next step is to train the localization algorithm. As knowledge about such techniques should not be essential, the choice for a certain algorithm should not be in responsibility of the librarians, but rather solely depend on the collected data.

As the signal propagation scheme may change throughout time, it should be possible to update the model resulting from the training process. This can be done either explicitly by the librarians or during normal operation using the requests issued by the users themselves.

Mobile devices shall be unburdened from computational and storage requirements as much as possible. The localization service shall be provisioned by a server in the infrastructure. This saves energy of the mobile devices and besides it also simplifies the management of model updates, because updated classification models do not need to be distributed to all users.

No dedicated infrastructure components shall be required except a way to communicate over some network with the server. Even the server itself shall not necessarily need to be provided by the library, but can instead reside somewhere in the Internet for public use. As it is incalculable if the users accept and use the localization service, the cost for running and maintaining the server shall be kept to a minimum. Moreover, the management of quality of service requirements like availability and scalability shall not be in responsibility of the librarians.

The user's privacy shall be respected. Given that the librarians and the server operator are independent parties, neither of these shall be able to localize users, i.e. the final localization step - the mapping of a symbolic label to a concrete location - needs to be done by the client.

Summarized, the librarians require an easy way to collect all the reference information without prior knowledge of the environment as well as an independent computing infrastructure, which is easy to set up and maintain and which is ideally paid for on a per-use basis to keep costs to a minimum. In the following section, our approach to realize such an application scenario is presented.

4 System Design

In this section, the overall system design is presented. Starting with a bird's-eye view on the system in Section 4.1, the cloud-based machine learning solution is presented in Section 4.2. Afterwards, the data model to be used by the machine learning algorithms is presented in Section 4.3. Section 4.4 introduces the client-side localization library used for collecting and transmitting reference information as well as issuing localization requests and validating results.

4.1 System Overview

The overall system overview is depicted in Figure 1. All the reference information are collected by a user and the resulting radio map is transmitted to a cloud storage. This storage may additionally be used to store further meta information like maps, points of interest, etc., but this is beyond the scope of this paper. The machine learning algorithms, residing in the cloud as well, access the storage to start training their models. After training has been finalized other users may issue localization requests containing a sample of the received signal strengths from all access points. The machine learning algorithms classify the sample and return the initially assigned label of the user's current location.

4.2 Machine Learning Blackbox

As already mentioned in Section 2, multiple localization methods exist, but only some are suitable also for indoor environments. Wifi-based localization represents the best choice for most application domains as it offers a good trade-off between infrastructure, resource and runtime requirements. A detailed comparison of indoor-relevant methods, regarding operational and installation efforts, can be found in [Gle12]. Extensive research during the last decades resulted in a multitude of techniques for determining the users' location with an accuracy of only a few meters. In particular, fingerprinting techniques are often used in combination with statistic/probabilistic methods as these try "'*to derive the unknown*

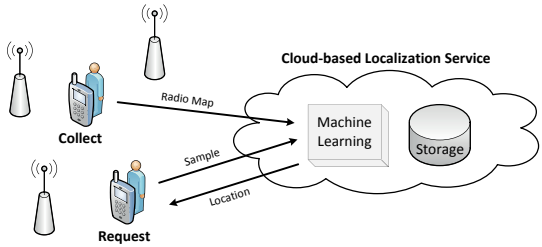


Figure 1: System Overview

functional dependency on the basis of observations"' [BB05]. Well known representatives are e.g. *Neural Networks*, *Bayesian modelling* and *Support Vector Machines*, but it is still discussed which of these are ideally used for localization under certain conditions.

As prior knowledge about the environment shall not be required, one cannot say which algorithm will perform best. Ideally, a whole set of different algorithms is trained concurrently using the reference information. The algorithm with the best classification performance is then chosen for later use. Depending on the amount of reference information, training several algorithms is very resource intensive and time consuming. In contrast, handling a localization request is computationally negligible, which is one of the reasons why scalable and elastic cloud infrastructures are well suited for these tasks.

From a user's perspective, the algorithm suite can be seen as a blackbox. What happens inside the box, which algorithms are used and what data format is required, is neither of interest for the one collecting the reference information, nor the one issuing a localization request. The only requirement is a well-defined interface which allows for i) transmission of reference information, ii) training and iii) updating of classification models as well as iv) requesting a location.

4.3 Data Model

As part of the blackbox interface a uniform data model is required. The design of the data model is essential for the ability to correctly classify samples[2]. In Section 2.2 a sample has been defined as an $(n+1)$-dimensional vector with n attributes representing the RSSI values and one label for the location. A radio map is an m-dimensional vector containing m samples. While the order of the samples within the radio map is irrelevant, the order of attributes within one sample is important, because to ensure, that the previously mentioned privacy aspects are met, there does not exist an explicit mapping from the unique access point identifier (so called *Basic Service Set Identification*, BSSID) to the measured signal strength received from that access point. Table 1 depicts the final data model.

[2]A discussion about different models can be found in [Gle12]

	BSSID 1	BSSID 2	BSSID 3	BSSID 4	BSSID 5
"'Room A1"'	-100	-100	-95	-61	-37
"'Room A2"'	-100	-100	-75	-100	-76
"'Room A3"'	-85	-64	-100	-100	-100
"'Room A4"'	-93	-100	-83	-42	-100

Table 1: Data model to be used for classification [Gle12]

The first row containing the BSSIDs does not belong to the data model, it is only shown to ease understanding. Important to note is, that for every access point, that has been seen while collecting the reference information, an attribute has to be present in each sample. If at one location an access point is not visible, instead of using the received signal strength a constant (e.g. -100) representing a very low RSSI needs to be inserted [RMT+02].

Summarized, a sample is represented as an ordered vector of attributes representing the RSSI from all access points as well as a label denoting the location. A radio map used to train the machine learning algorithms is an unordered vector of an arbitrary number of samples. Of course, when a user requests a location by sending a sample to the localization service, the label is omitted.

4.4 Client Component Architecture

On the client side, two main tasks can be identified: i) collecting reference information and transmitting the information to the machine learning blackbox and ii) sending a request to the localization service. The former task needs to be initially done by some user before the localization can take place. For this purpose, the user has to take samples at certain locations within the area of interest. In a library for example, samples could be taken at the beginning and end of each shelf. At each location several measurements need to be taken. Thereby, multiple measurements can be combined in one sample, e.g. by calculating an average, and multiple samples per location are useful when training the algorithms later on. Moreover, the user needs to be prompted to provide a symbolic name for the location at which the samples are taken, e.g. "'Shelf Bio A-L"'. If this is done for all locations of interest, the reference information is complete and can be transmitted to the machine learning blackbox, where the training starts subsequently. This task can easily be supported by a mobile application, which aids the user (see Section 5).

The other main task that needs to be supported is accessing the localization service. From time to time, users may issue a localization request which is answered with the location's label. This information can be used in various ways and it depends on the concrete application how this information is further processed. A navigation application could display a pin on a map and guide the way to some target location. Or in the watchman domain it might just be necessary to create a log entry that proofs that the watchman visited the location at some point in time. Due to the multitude of possible applications the access to the localization service needs to be encapsulated in some independent module which can be easily integrated in arbitrary applications.

Localization Plausibility The localization result represents a discrete location, because we use symbolic labels and classify samples[3]. From time to time, wrong classifications must be expected either because the environmental conditions changed or because the classification algorithms could not correctly classify a given sample. But depending on the distance between the reference points, a misclassification can result in sudden location changes of several meters which needs to be avoided.

Therefore, the results received from the localization service should be validated on the client-side. For this purpose, we make additional use of dead reckoning techniques. Using the compass and the accelerometer, a client-side localization component tries to keep track of the client's orientation and movement. Dead reckoning itself is quite inaccurate due to inherent sensor deviations and cumulated errors, but it helps to determine the feasibility of a result received from the localization service. If validation fails a new localization request has to be performed.

Component Interplay Figure 2 depicts the main responsibilities of our client-side localization module. The *GUI/Application* does not belong to the module, although we realized several to help collecting the reference information and to display the location on a map. In general, the first step (1) is a localization request by the application. The *Wifi Manager* takes several measurements and (2) dispatches these to the *Data Manager* where they can be aggregated and filtered before they are encoded and (3) further dispatched to the *Network Manager*. This component (4) interacts with the *Localization Service* and exchanges samples and location information. Upon (5) receiving a location response, the *Sensor-based Localization*, which continuously keeps track of the user's motion and orientation, additionally (6) dispatches its own location estimate to the data manager. Finally, the localization estimates resulting from the classification as well as from the local sensors are (7) handed back to the application, which is then responsible for the further processing, e.g. to check the plausibility of the estimate. Because this last aspect relies on domain information as the location label has to be interpreted, it currently cannot be done by the localization module.

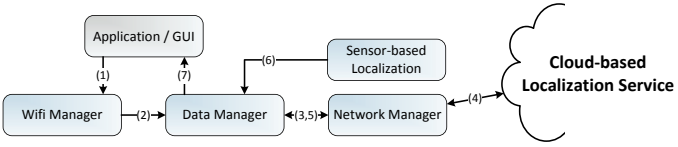


Figure 2: Client Component Interplay

[3]Besides classification also regression techniques could be used. Only the data model must be slightly adapted to not hold a symbolic name as a label, but some numeric value. In this case, the answer would not contain a discrete class label, but a numeric value which is based on the closeness to existing reference samples.

5 Prototypical Implementation

Following the design presented in the last section, we realized a prototypical implementation of the localization library for the *Android* platform. Additionally, based on this library an application for collecting reference information as well as a simple navigation application have been implemented. Figure 3 depicts screenshots of both applications. The implementation is straight-forward as the main logic for localization resides in the machine learning blackbox. Details about the implementation can be found in [Gle12].

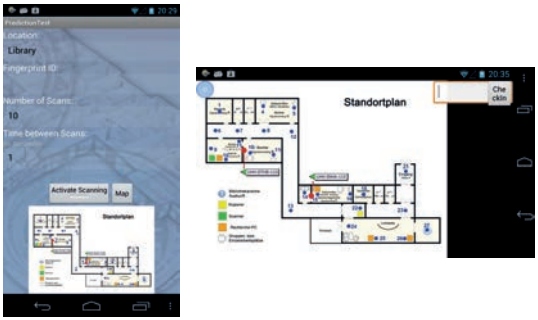


Figure 3: Application for Collecting Reference Information (left) and Navigation (right) [Gle12]

To the best of our knowledge, there is only one provider for machine learning in an SaaS-cloud, namely *Google Prediction* (v1.4) [Goo12]. Google Prediction is a blackbox containing several different machine learning algorithms which are all trained during the offline phase in which formerly measured reference points are learned. The performance of each algorithm is then evaluated using a training data subset and the best algorithm is finally chosen for the online phase in which users issue localization requests. It is also possible to update the classification model so that environmental changes can be accounted for. The service is accessed using a REST-based API. By exchanging JSON-encoded messages the protocol overhead is minimized which is particularly suitable for mobile communication.

To this end, our goal to ease the setup of a localization solution is attained: a lightweight client-side implementation for taking samples and accessing the remote localization service has been realized. On the service-side, no dedicated hardware has to be acquired and no additional software is necessary. Also, besides initially collecting reference information no prior knowledge about either localization techniques or environmental conditions is required. Moreover, using a 3rd-party cloud infrastructure solves several non-functional requirements like high availability, scalability, etc. and if meaningless symbolic labels are used for locations the user's privacy is taken adequately into account.

6 Evaluation

Although our research is still in progress the first evaluations failed to live up to expectations, because the most important requirement, namely correctness, is not met. We conducted two evaluations: i) on one level within an office building and ii) in a library (see Figure 4). In the office building a floor connects 28 rooms of different sizes. Cumulated 79 different access points (partly virtual) were visible throughout the whole floor. We collected reference information at 50 different locations at the floor as well as in the rooms. At each location 50 samples were taken within one day. The resulting radio map with which we trained the machine learning algorithms thus holds 2,500 samples, each containing 80 features representing the RSSIs of the all access points as well as a label. The library setting differs slightly: only 45 access points (partly virtual) were visible, but we took 150 reference measurements at 24 different locations each distributed evenly over three days.

Figure 4: Overview of the two Field Test Areas: Office Floor (left) and Library (right) [Gle12]

After training the machine learning algorithms we used three different input sets for localization requests: i) 50 randomly picked samples of the reference set, ii) an average over all 50 reference samples for each location and iii) live data we collected afterwards.

6.1 Results

Localization using samples from the first test set were always classified correctly. The same holds for all samples from the second test set in the office settings. The results for the third test set are depicted in Figure 5. The diagram shows the localization correctness of all locations in the office field test, subdivided into floor and room locations. At five floor (23 % of all floor locations) and seven (25 %) room locations the machine learning blackbox was not able to output any correct classification. At eleven (50 %) different floor and nine (32 %) room locations, at least up to 70% of our localization requests were correctly answered which is still bad. At three floor and seven room locations we were in 71%-90% correctly localized and at only three floor locations and five room locations the results were nearly always correct. The diagram also clearly shows a significant difference between floor and room locations: as there were no obstacles between the measurement points at the floor that could influence radio wave propagation it is much harder to correctly classify such locations due to similar RSSI values at each location.

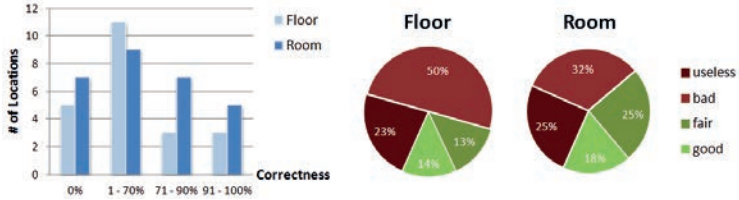


Figure 5: Correctness of Localization in the Office Setting

In the library setting we were not able to correctly localize ourselves at all. While samples from the reference set were always correctly classified, any change in one of the features resulted in the same wrong location. The localization correctness in this setting converges to 0 % for the second and third input set. Although we re-validated the experiment and used different sizes and samples of our overall reference set, correctness did not increase. And due to all machine learning algorithms residing in a black box debugging is not possible.

7 Conclusion and Future Work

The necessity of indoor localization solutions that could easily be set up and used by everyone is undoubtful. Our approach towards realizing such a solution includes a lightweight localization library to be used by client applications on mobile devices and a public cloud-based localization service which is represented by a machine learning blackbox. This combination allows to easily set up a localization system with neither prior knowledge of the environment nor any special technical knowledge. Moreover, several non-functional requirements such as scalability, minimized costs and privacy issues are adequately fulfilled by the cloud infrastructure.

An evaluation evidenced that setting up a localization system using our prototypical client application for collecting reference information and triggering the machine learning algorithms is indeed very easy. But the evaluation also revealed that our approach using the only publicly available cloud-based machine learning service, namely Google Prediction, for location learning does not yield accurate results at all.

Therefore, our prospects for future work focus on exchanging the SaaS-based machine learning blackbox with more suitable solutions. In particular, this involves moving existing distributed machine learning algorithms into the cloud, ideally into a managed PaaS-cloud.

References

[AAH+09] Theodoros Anagnostopoulos, Christos Anagnostopoulos, Stathes Hadjiefthymiades, Miltos Kyriakakos, and Alexandros Kalousis. Predicting the location of mobile users: a machine learning approach. In *Proceedings of the 2009 international conference on Pervasive services*, ICPS '09, pages 65–72, New York, NY, USA, 2009. ACM.

[BB05] Mauro Brunato and Roberto Battiti. Statistical learning theory for location fingerprinting in wireless LANs. *Computer Networks*, 47(6):825–845, April 2005.

[BBG11] R. Buyya, J. Broberg, and A.M. Goscinski. *Cloud Computing: Principles and Paradigms*. Wiley Series on Parallel and Distributed Computing. Wiley, 2011.

[Fuc05] T. Fuchß. *Mobile Computing: Grundlagen und Konzepte für mobile Anwendungen*. Hanser, Carl, 2005.

[Gle12] Daniel Gleim. Sensorgestützte WLAN-Positionsbestimmung durch maschinelle Lernverfahren zur effizienten Indoor Navigation. Bachelor's thesis, UniversitätHamburg - Fachbereich Informatik - Verteilte Systeme und Informationssysteme, April 2012.

[Goo12] Google Inc. Google Prediction API. https://developers.google.com/prediction/docs/gettingstarted, accessed 6. Sept. 2012.

[HB01] Jeffrey Hightower and Gaetano Borriello. Location Sensing Techniques. UW CSE 01-07-01, University of Washington, Department of Computer Science and Engineering, Seattle, WA, July 2001.

[HPALP09] Ville Honkavirta, Tommi Perälä, Simo Ali-Löytty, and Robert Piché. A Comparative Survey of WLAN Location Fingerprinting Methods. In *Proceedings of the 6th Workshop on Positioning, Navigation and Communication 2009*, pages 243–251, March 2009.

[Küp05] A. Küpper. *Location-based services: fundamentals and operation*. John Wiley, 2005.

[LDBL07] Hui Liu, H. Darabi, P. Banerjee, and Jing Liu. Survey of Wireless Indoor Positioning Techniques and Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(6):1067–1080, November 2007.

[Mar09] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC, 1st edition, 2009.

[MD07] Eric D. Manley and Jitender S. Deogun. Location Learning for Smart Homes. *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*, 2:787–792, May 2007.

[PL05] K. Pahlavan and A.H. Levesque. *Wireless Information Networks*. Wiley Series in Telecommunications and Signal Processing. John Wiley, 2005.

[RMT+02] Teemu Roos, Petri Myllymäki, Henry Tirri, Pauli Misikangas, and Juha Sievänen. A Probabilistic Approach to WLAN User Location Estimation. *International Journal of Wireless Information Networks*, 9:155–164, 2002. 10.1023/A:1016003126882.

Bayes-Filter für kombinierte GPS/GLONASS-Lokalisierung

Pierre Reisdorf, Marcus Obst und Gerd Wanielik
Professur für Nachrichtentechnik, Technische Universität Chemnitz
Reichenhainer Straße 70, 09126 Chemnitz
Telefon: +49 371 531 36688, Fax: +49 371 531 836688
Email: pierre.reisdorf@etit.tu-chemnitz.de

Abstract: In dieser Arbeit wird die Nutzung eines Bayes-Filters zur Verbesserung der Lokalisierungsgenauigkeit für den Anwendungsfall der Mehrsystem-Satellitennavigation vorgeschlagen. Dazu wird ein konkreter Bayes-Filter unter Zuhilfenahme von physikalischen Modellen implementiert und in einem statischen sowie dynamischen Szenario mit einem klassischen Lokalisierungsalgorithmus verglichen. Es kann gezeigt werden, dass das Bayes-Filter konventionellen Verfahren überlegen ist und gleichzeitig Anwendung in anderen Gebieten erlaubt.

1 Einführung

Für aktuelle Anwendungen, z.B. im Bereich der *Standortbezogenen Dienste*, ist eine genaue und zuverlässige Lokalisierung von großer Bedeutung. Die Positionslösung (auch GNSS-Fix genannt) soll dabei vorzugsweise stabil und kontinuierlich zur Verfügung stehen und Messunsicherheiten der Sensorwerte korrekt behandeln. Weiterhin ist es oft notwendig, dass auch aus der Position abgeleitete Größen, wie z.B. die Geschwindigkeit und die Ausrichtung, angeben werden können. Die in dieser Arbeitet betrachtete satellitengestützte Lokalisierung, die normalerweise ohne externes Zusatzwissen zu einer Positionslösung kommt, erfüllt diese Anforderungen allerdings oft nur unzureichend.
Durch die Hinzunahme eines *Bayes-Filters*[1] ist es prinzipiell möglich, oben genannte Anforderungen zu erfüllen. Ein Bayes-Filter ist in der Lage, asynchrone und unsichere Messungen von heterogenen Sensoren zu nutzen und stellt somit eine generische Möglichkeit zur Sensordatenfusion bereit. Durch die zusätzliche Einbindung von Modellwissen, das den beobachteten physikalischen Prozess statistisch beschreibt, können auch nicht direkt beobachtbare Zustandsgrößen angegeben und kurzzeitige Aussagen über die Entwicklung des Systems in der Zukunft (oft als Zustandsprädiktion bezeichnet) getroffen werden.
Der Fokus in diesem Paper liegt vorwiegend in der Beurteilung der Verbesserung der Positionierung mit einem Bayes-Filter gegenüber dem Least-Square-Verfahren. Abbildung 1 zeigt die prinzipelle Arbeitsweise eines Filters mit Eingangsgrößen von verschiedenen Sensoren und der Verarbeitung dieser Daten sowie Ausgangsgrößen als Schätzung.

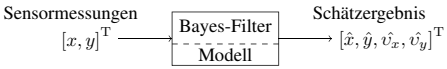


Abbildung 1: Blackbox-Modell eines Bayes-Filters mit der grundlegenden Verfahrensweise. Das Filter bekommt Sensormessungen als Eingangsgrößen und liefert Schätzungen über verschiedene Zustände als Ausgangsgrößen. Dargestellt sind als Eingangsgrößen die kartesischen Koordinaten x und y gezeigt. Als Ausgangsgrößen liefert das Filter eine Schätzung ebenfalls zu diesen Koordinaten und zusätzlich eine Schätzung zur Geschwindigkeit in x und y.

[1] Ein bayessches Filter ist ein Verfahren zur statistischen Zustandsschätzung.

2 Stand der Technik & Ähnliche Arbeiten

In der Literatur finden sich unterschiedliche Ansätze zur Nutzung von mehr als einem globalen Satellitennavigationssystem (GNSS). Neben der Nutzung eines Partikel-Filters in [NDM08] findet sich auch die Verwendung eines Kalman-Filters in [DTDC08]. Durch die wachsende Anzahl an Satellitensystemen (neben GPS und GLONASS irgendwann auch GALILEO usw.) kam nach und nach die Bestrebung auch mehr als ein Satellitensystem zur Positionierung zu verwenden. Damit einhergehend soll die Positionslösung dementsprechend verbessert werden. In [Mat11] wird sich mit der Nutzung von mehr als einem Satellitensystem beschäftigt. Neben der Verwendung einer berechneten Positionslösung als Eingangsgröße für einen Filter, könnten weitere Sensoren, wie in [DNV+09], als Input genutzt werden. In [RAG04] und [Can11] wird auf die Verwendung von Filtern mit den günstigen Eigenschaften zur Verarbeitung von Signalen ausführlich eingegangen.

3 Grundlagen

3.1 GNSS Positionierung

Die Positionierung mit einem GNSS erfolgt über eine indirekte Entfernungsbestimmung zwischen einem Empfänger und einem Satelliten. Die Entfernung, vorzugsweise auch Pseudoentfernung genannt, wird über das Messen der Signallaufzeit – Time of arrival (TOA) – zwischen der Antenne des Empfängers und eines Satelliten indirekt bestimmt. Für eine Positionierung im Raum werden drei zeitsynchrone Messungen zu drei unterschiedlichen Satelliten benötigt. Durch die Verwendung der Signallaufzeit und der nicht synchronisierten Uhren der Satelliten und des Empfängers existiert ein unbekannter Uhrzeitenfehler, der in jeder Messung enthalten ist. Um diesen Fehler zu bereinigen wird eine weitere Messung zu einem Satelliten benötigt, wodurch im Minimum vier zeitsynchrone Messungen für eine 3D-Positionierung erforderlich sind.
Eine Pseudoentfernung ρ lässt sich in Anlehnung an [KH06] wie folgt modellieren:

$$\rho = r + c(dt - dT) + d_e \tag{1}$$

In der gegebenen Gleichung ist c die Lichtgeschwindigkeit und r die wahre Entfernung zwischen dem Empfänger und dem Satelliten. Der Satellitenuhrzeitfehler dT eines Satelliten wird über die Navigationsnachrichten des jeweiligen Satelliten mit übertragen. Der Empfängeruhrzeitfehler dt ist unbekannt. Zusätzlich existieren weitere Fehler, wie Ionosphären-, Troposphären, Ephemeriden- oder Mehrwegefehler usw., welche vorliegend zu d_e zusammengefasst sind.

Bei einer Verfügbarkeit von mindestens vier Satelliten kann der Empfängerzustand

$$x_{\text{receiver}} = \begin{pmatrix} x & y & z & dt \end{pmatrix}^{\mathrm{T}} \tag{2}$$

mit einem Least-Squares-Algorithmus oder auch mit einem Bayes-Filter, zum Beispiel einem Kalman-Filter, berechnet werden. Wobei x, y, z die Empfängerposition in Earth-Centered, Earth-Fixed (ECEF) Koordinaten beschreibt und dt den Empfängeruhrzeitfehler.

3.2 Mehrsystem GNSS Positionierung

Neben dem amerikanischen GPS steht mit dem russischen GLONASS noch ein weiteres GNSS zur Positionierung zur Verfügung. GLONASS ist gleichwertig zum GPS und ebenfalls weltweit verfügbar. Beide Systeme können für eine gemeinsame Positionierung verwendet und abstrakt gesehen als ein GNSS verwendet werden. Wenn beide Satellitensysteme verwendet werden, ergibt sich neben dem Empfängeruhrzeitfehler für GPS (dt_{GPS}) ein zusätzlicher Uhrzeitfehlers des Empfängers für GLONASS (dt_{GLONASS}). Der Empfängerzustand wird somit zu

$$x_{\text{receiver}} = \begin{pmatrix} x & y & z & dt_{\text{GPS}} & dt_{\text{GLONASS}} \end{pmatrix}^{\mathrm{T}} \tag{3}$$

erweitert. Dadurch wird für eine 3D-Positionierung mit beiden Satellitensystemen eine Mindestanzahl von fünf Satelliten erforderlich. Auf dieses Problem des zusätzlichen Satelliten wird in [JT09] näher eingegangen und in [XLX11] werden dafür zwei Lösungen vorgestellt.

4 Bayes-Filter

Die Verwendung von Filtern – im linearen Fall oft als Kalman-Filter [JU04] umgesetzt – ermöglicht die Prädiktion von Systemzuständen in die Zukunft durch Zuhilfenahme von Modellwissen. In Abhängigkeit der verfügbaren Sensoren, der gewünschten Ausgabegrößen und dem gegebenen Anwendungsfall gibt es unterschiedliche Implementierungsmöglichkeiten. In [SAO$^+$11] wird bspw. auf eine Klasse von relevanten Bewegungsmodellen für die Fahrzeuglokalisierung näher eingegangen.
Ein Bayes-Filter ist ein rekursiver Schätzer, der anhand des letzten Zustands mit Modellwissen in die Zukunft prädiziert. Im vorligenden Paper werden das *Constant Position* (CP) und das *Constant Velocity*-Modell (CV) unter Beachtung der Satellitensysteme GPS und GLONASS sowohl für ein statisches als auch ein dynamisches Szenario untersucht und evaluiert. Das Filter ist dabei als loosly-coupled implementiert, womit das Filter nur ganze Positionslösungen als Eingangsgröße verarbeiten kann.

4.1 Fahrzeugbewegungsmodell

Durch die Verwendung von Fahrzeugen als Testobjekte werden die vorliegenden Bewegungsmodelle auch als Fahrzeugbewegungsmodelle bezeichnet. Sie beschreiben entsprechend der gewünschten Implementierung physikalische Bedingungen an ein Fahrzeug. In [SAO$^+$11] wird eine systematische Übersicht über unterschiedliche Bewegungsmodelle gegeben. Zusammen mit einem Bayes-Filter kann das Bewegungsmodell zum Stabilisieren und Glätten einer GNSS-Positionslösung verwendet werden. Ein wesentlicher Vorteil des Filters ist die Prädiktion von Zuständen in die Zukunft, womit kurze Ausfälle wie bspw. im urbanen Gebiet durch die Einschränkung der Sichtbarkeit zu den Satelliten überbrückt werden können.
Im vorliegenden Paper werden zwei Bewegungsmodelle verwendet, zum einen *Constant Position* (CP) und zum anderen *Constant Velocity* (CV). Auf beide Modelle wird nachfolgend eingegangen.

Als einfaches Modell dient das *Constant Position*-Bewegungsmodell. Dieses besitzt den Zustandsraum

$$^{\mathrm{CP}}\vec{x} = (x \quad y)^{\mathrm{T}}, \tag{4}$$

wobei x und y die kartesischen Koordinaten einer zweidimensionalen Position (als UTM-Koordinaten) beschreibt. Der Zustandsübergang vom Zeitpunkt k zum Zeitpunkt $k+1$ ergibt sich bei diesem Modell nach

$$\vec{x}_{k+1} = \vec{x}_k. \tag{5}$$

Das zweite Bewegungsmodell ist das *Constant Velocity*-Modell. Bei diesem Modell wird der Zustandsraum

$$^{\mathrm{CV}}\vec{x} = (x \quad y \quad \vartheta \quad v)^{\mathrm{T}} \tag{6}$$

um die Ausrichtung ϑ und die Geschwindigkeit v erweitert. Der Zustandsübergang ändert sich entsprechend des erweiterten Zustandsraumes zu

$$\vec{x}_{k+1} = \vec{x}_k + \begin{pmatrix} v\cos(\vartheta)\,T_k \\ v\sin(\vartheta)\,T_k \\ 0 \\ 0 \end{pmatrix}, \tag{7}$$

T_k der Zeit zwischen $_k$ und $_{k+1}$ entspricht.

4.2 Sensormodell

Ähnlich den Bewegungsmodellen existieren eine Vielzahl unterschiedlicher Sensormodelle, die eine Transformation vom Zustandsraum in den Messraum abbilden. In diesem Paper werden ausschließlich Sensoren verwendet, die eine direkte Beobachtung der Elemente des Zustandsvektors ermöglichen. Diese sind die Positionsdaten, wohingegen Odometriedaten nicht verfiltert werden.

5 Implementierung

Als konkrete Implementierung des Bayes-Filter wurde ein Unscentend Kalman Filter (UKF) [JU04] verwendet. Dieser beschreibt eine Weiterentwicklung des Kalman Filter [vdM04] und ist für nichtlineare Probleme, wie es bei dem vorliegen *Constant Velocity*-Modell der Fall ist, besser geeignet. Neben einem einstellbaren Prozessrauschen für die Ausrichtung ϑ und die Geschwindigkeit υ – wird zur Prädiktion vin Zuständen erforderlich – erwartet der der UKF die Angabe von Skalierungsparametern (α, β, κ), welche in Tabelle 1 aufgeführt sind. Zusätzlich hat sich bei den Messdaten eine Standardabweichung des Prozessrauschens der Geschwindigkeit (σ_υ) von 5,0 m/s und eine Standardabweichung des Prozessrauschens der Ausrichtung (σ_ϑ) von 0,6 rad/s als empfehlenswert herausgestellt.

Tabelle 1: Verwendete UKF-Parameter, die für den Betrieb des Filters benötigt werden

Parameter	Beschreibung	Wert
σ_υ	Standardabweichung des Prozessrauschens der Geschwindigkeit	5,0 m/s
σ_ϑ	Standardabweichung des Prozessrauschens der Ausrichtung	0,6 rad/s
α	Skalierungsparameter Alpha	0,001
β	Skalierungsparameter Beta	2
κ	Skalierungsparameter Kappa	0

6 Evaluationsmethodik

6.1 Sensor Konfiguration

Zur Aufnahme der Messdaten wurde das Testfahrzeug Carai [SRM+10] verwendet. Der Carai ist mit einem NovAtel OEMV GNSS Receiver ausgestattet, welcher sowohl die Ausgabe von GPS- wie auch von GLONASS-Messrohdaten unterstützt. Die Pseudoentfernungen von beiden Systemen sind mit 10 Hz aufgenommen wurden. Zum Vergleich der eigenen algorithmischen Lösung wird eine hochgenaue Referenztrajektorie benötigt. Dafür wird ein NovAtel SPAN Receiver mit GNSS Positionierung und Inertial navigation system (INS) verwendet. Jener besteht aus einem Zwei-Frequenz-Empfänger sowie der Unterstützung für Real Time Kinematic (RTK). Zusammen mit der Honeywell HG1700 inertial measurement unit (IMU) ergibt das für die Referenztrajektorie eine Genauigkeit im Zentimeterbereich.

6.2 Evaluationskriterien

Von Interesse ist zum Vergleich von unterschiedlichen Algorithmen jeweils die Abweichung zur Referenztrajektorie. Dabei hat sich der Root Mean Square Error (RMSE) als guter Vergleichswert erwiesen. Daneben ist die Standardabweichung (σ, 2σ, 3σ) sowie die Einhaltung dieser durch den entsprechenden Algorithmus selber von Bedeutung. Für das statische Szenario wurde der Fehler sowie das σ für jede Richtung angegeben.

7 Quantitative Ergebnisse

Die beiden zu evaluierenden Modelle entsprechen jenen zu untersuchenden Szenarien. Zum einen werden die Lösungen mit Least-Squares gegenüber dem Bayes-Filter mit dem CP-Modell für ein statisches Szenario untersucht. Zum anderen werden die Lösungen mit Least-Squares und des Bayes-Filter mit dem CV-Modell für ein dynamisches Szenario gegenüber gestellt. Beide Modelle entsprechen demnach mit ihren Eigenschaften dem gewünschten Szenario.

7.1 Statisches Szenario

Tabelle 2 zeigt die Ergebnisse für eine GNSS-Lokalisierung mit einer klassischen Least-Squares-Lösung sowie einem Bayes-Filter mit dem CP-Modell. Der Mittelwert des Positionsfehlers wird wie erwartet kaum verbessert, die Standardabweichung wird jedoch erheblich verkleinert, d.h. das Bayes-Filter liefert eine verlässlichere Lösung.

Tabelle 2: Vergleich der klassischen Least-Squares-Lösung mit dem Ergebnis eines Bayes-Filters (CP-Modell) für ein statisches Szenario unter der Annahme, dass die Fehler normalverteilt sind.

Algorithmus	Fehler x [m]	Fehler y [m]	σ x [m]	σ y [m]
Least-Squares	−0,75	−2,14	0,90	0,79
Bayes (CP)	−1,26	−1,77	0,24	0,18

7.2 Dynamisches Szenario

Als Einzelsystemempfang (nur ein Satellitensystem wird zur Positionierung verwendet) unter Nutzung des GPS liefert das Bayes-Filter mit dem CV-Modell zufriedenstellende Ergebnisse. Abbildung 2 und 3 zeigen die 2D-Positionsfehler inklusive der 3σ-Konfidenzintervalle für die Least-Squares-Lösung und für das Bayes-Filter. Leicht erkennbar ist die Verringerung der Standardabweichung unter Verwendung des Filters. Zusätzlich passt sich die Standardabweichung beim Filter dynamischer an, wohingegen sie bei der Least-Squares-Lösung *nur* von der Satellitenkonstellation abhängig ist und deshalb eher statisch wirkt.

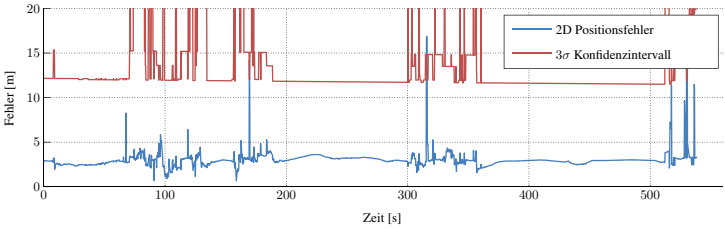


Abbildung 2: Ergebnisse der Least-Squares-Lösung unter Nutzung des Satellitensystems GPS

Neben der grafischen Darstellung in Abbildung 2 und 3 zeigen Tabelle 3 und 4 die Werte für den RMSE, die prozentuale Anzahl an Fixe, wie die angegebenen σ-Konfidenzintervalle eingehalten werden und zusätzlich die Größe der durchschnittlichen Standardabweichung $\overline{\sigma}$. Tabelle 3 beinhaltet die Ergebnisse für ein Einzelsystemempfang und Tabelle 4 die Ergebnisse für Mehrsystemempfang. In beiden Fällen unterscheiden sich der RMSE kaum voneinander. Die durchschnittliche Standardabweichung wird beim Bayes-Filter erheblich herabgesenkt, allerdings auf Kosten der Einhaltung der entsprechenden σ-Konfidenzintervalle. Dieser Effekt verstärkt sich unter

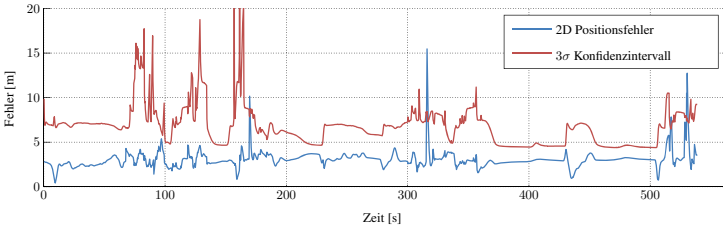


Abbildung 3: Ergebnisse des Bayes-Filter unter Nutzung des Satellitensystems GPS

Verwendung von zwei GNSS-Systemen zusätzlich, jedoch in diesen Fall auch bei der Least-Squares-Lösung. Da die Ergebnisse als Einzelsystemempfang mit GPS besser aussehen, ist in diesem Messabschnitt ein schlechterer Empfang der GLONASS-Satelliten ein Grund für die Ergebnisse. Diese Effekte müssen in weiterführenden Arbeiten genauer untersucht werden.

Tabelle 3: Vergleich der klassischen Least-Squares-Lösung mit dem Ergebnis eines Bayes-Filters (CV-Modell) für ein dynamisches Szenario als Einzelsystemempfang unter der Annahme, dass die Fehler normalverteilt sind.

Algorithmus	σ	2σ	3σ	$\overline{\sigma}$	RMSE
Least-Squares	98,28	99,54	99,72	13,87 m	3,17 m
Bayes (CP)	18,56	86,03	99,12	6,76 m	3,19 m

Tabelle 4: Vergleich der klassischen Least-Squares-Lösung mit dem Ergebnis eines Bayes-Filters (CV-Modell) für ein dynamisches Szenario als Mehrsystemempfang unter der Annahme, dass die Fehler normalverteilt sind.

Algorithmus	σ	2σ	3σ	$\overline{\sigma}$	RMSE
Least-Squares	28,82	84,08	98,37	7,31 m	4,27 m
Bayes (CP)	9,59	30,68	56,89	4,35 m	4,11 m

7.3 Prädiktion einer Positionslösung

Die Prädiktion einer möglichen Position in die Zukunft ermöglicht dem Filter gegenüber einer Least-Squares-Lösung mehr GNSS-Fixes. In einem vorliegenden Messszenario mit 2190 Fixes für den Filter gegenüber 2155 Fixes für die Least-Squares-Lösung bedeutet dies rund 1,6 % mehr Fixes für den Filter. Bei Szenarien mit erheblich mehr Einschränkungen durch die Umgebung, siehe urbanes Gebiet, fallen die Unterschiede in der Anzahl der möglichen Fixe sicherlich wesentlich deutlicher aus. In Abbildung 4 ist die Prädiktion eines GNSS-Fixes in die Zukunft mit einer vergrößerten Messunsicherheit dargestellt. Für eine Least-Squares-Lösung stehen unter der Brücke nicht genügend Satelliten zur Verfügung, wohingegen das Bayes-Filter mit einer größer werdenden Unsicherheit weiterhin eine Positionslösung liefert.

8 Zusammenfassung

Es konnte gezeigt werden, dass ein Filter die Positionslösung stabilisieren sowie auch glätten kann. Zusätzlich hat ein Filter den positiven Effekt der zeitlichen Überbrückung von Sensorausfällen, was bspw. in einem urbanen Gebiet durch die Einschränkung der Sichtbarkeit zu den GNSS-Satelliten oft vorkommen kann. Ein weiterer Vorteil

Abbildung 4: Prädiktion einer Positionslösung unter eine Brücke (rot), bei der die normale Lösung mit Least-Squares (blau) keine Position aufgrund fehlender Sichtbarkeit zu ausreichend vielen Satelliten berechnen kann. Neben der Möglichkeit einer Positionsangabe liefert das Filter ebenfalls eine Unsicherheit, was als Ellipse dargestellt wird.

des Filters ist die Schätzung von nicht beobachtbaren Größen wie der Geschwindigkeit und der Ausrichtung des Fahrzeugs. Unter Verwendung von einem Satellitensystem konnte gezeigt werden, dass die Lösung mit dem Bayes-Filter verlässlicher wird. Bei der Verwendung von zwei Systemen wird die durchschnittliche Standardabweichung ebenfalls herabgesenkt, jedoch auf Kosten der Einhaltung der entsprechenden Konfidenzintervalle. Der Fehler liegt dabei vermutlich an GLONASS, was in weitergehenden Arbeiten untersucht werden muss.

Literatur

[Can11] J.V. Candy. *Bayesian Signal Processing: Classical, Modern and Particle Filtering Methods*. Adaptive and Learning Systems for Signal Processing, Communications and Control Series. John Wiley & Sons, 2011.

[DNV+09] G. Duchateau, O. Nouvel, W. Vigneau, D. Betaille, F. Peyret und H. Secretan. How to Assess and Improve Satellite Positioning Performances in Urban Environments. In *16th ITS World Congress, Stockholm, 2009: Proceedings*, 2009.

[DTDC08] P. Delmas, C. Tessier, C. Debain und R. Chapuis. GNSS bias correction for localization systems. In *Proc. 11th Int Information Fusion Conf*, Seiten 1–6, 2008.

[JT09] J.-C. Juang und Y. Tsai. On exact solutions of the multi-constellation GNSS navigation problem. *GPS Solutions*, 13:57–64, 2009.

[JU04] Simon J. Julier und Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401 – 422, mar 2004.

[KH06] E. D. Kaplan und C. Hegarty. *Understanding GPS : Principles and Applications*. Artech House, 2.. Auflage, 2006.

[Mat11] P. G. Mattos. Accuracy and Availability Trials of the Consumer GPS/GLONASS Receiver in Highly Obstructed Environments. In *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, Seiten 2740–2744, Oregon Convention Center, Portland, Oregon, September 2011.

[NDM08] D. F. Nahimana, E. Duflos und J. Marais. Reception state estimation of GNSS satellites in urban environment using particle filtering. In *Proc. 11th Int Information Fusion Conf*, Seiten 1–5, 2008.

[RAG04] Branko Ristic, Sanjeev Arulampalam und Neil Gordon. *Beyond the Kalman Filter – Particle Filters for Tracking Applications*. Artech House, 2004.

[SAO+11] Robin Schubert, Christian Adam, Marcus Obst, Norman Mattern, Veit Leonhardt und Gerd Wanielik. Empirical evaluation of vehicular models for ego motion estimation. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 534—5 39, June 2011.

[SRM+10] Robin Schubert, Eric Richter, Norman Mattern, Philipp Lindner und Gerd Wanielik. *Advanced Microsystems for Automotive Applications 2010 - Smart Systems for Green Cars and Safe Mobility*, Kapitel A Concept Vehicle for Rapid Prototyping of Advanced Driver Assistance Systems, Seiten 211–219. Springer, 2010.

[TBF05] Sebastian Thrun, Wolfram Burgard und Dieter Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press, September 2005.

[vdM04] Rudolph van der Merwe. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. Dissertation, Oregon Health & Science University, April 2004.

[XLX11] Longxia Xu, Xiaohui Li und Yanrong Xue. Two Methods of Processing System Time Offset in Multi-constellation Integrated System. In Min Zhu, Hrsg., *Electrical Engineering and Control*, Jgg. 98 of *Lecture Notes in Electrical Engineering*, Seiten 359–365. Springer Berlin Heidelberg, 2011.

SmartBEEs: Enabling Smart Business Environments Based on Location Information and Sensor Networks

Florian Dorfmeister, Marco Maier, Mirco Schönfeld
Mobile and Distributed Systems Group
Ludwig-Maximilians-Universität München
Oettingenstr. 67, 80538 München
{forename}.{surname}@ifi.lmu.de

Stephan A. W. Verclas
T-Systems International GmbH
Elisabeth-Selbert-Str. 1
80939 München
stephan.verclas@t-systems.com

Abstract: This paper presents the design and current prototypic implementation of the *Smart Business and Enterprise Environments* (*SmartBEEs*) platform. The proposed architecture is targeted on the seamless integration of indoor and outdoor positioning systems, context and location models, as well as real-time sensor data readings for the automated creation and distribution of event-triggered, manual tasks based on a set of predefined event-condition-action rules. Possible fields of application for our platform can be seen in industrial production sites, logistics, retail and health care, as well as any other type of sensor-equipped environment. In contrast to many other systems that rely on indoor positioning, however, the focus of our work is not on indoor routing or pedestrian navigation. Instead, the *SmartBEEs* architecture aims at leveraging the current tasks and activities of its users as well as up-to-date positioning information and distance estimations to a given target location in order to enable a reasonable and efficient distribution of tasks among registered users in order to improve overall response times and productivity in smart environments.

1 Introduction and Motivation

By keeping a specific set of phenomena of interest under steady surveillance, sensor networks can help to monitor the current state of their surroundings and can thus be regarded as an important building block for the famous vision of ubiquitous computing [Wei91]. These networks' possible fields of application vary from military usage and large-scale environmental monitoring to everyday areas such as health care and assisted living, as well as building and business process automation. In addition to dedicated sensor nodes integrated into a site's infrastructure facilities, the mobile devices of on-site human users can be regarded as sensors, too. These user-carried devices are capable of providing additional application-specific information, such as a user's current location, her activities and interactions with the environment. At the same time, these mobile devices can naturally also be used as mere input and output devices, e.g., by displaying information about the current overall state of the system or allowing for the creation of tasks by users. Based on these considerations, the *SmartBEEs* platform aims at facilitating the realization of smart enterprise environments with a special focus on the efficient allocation of automatically created, event-triggered tasks to registered mobile users, i.e., the employees that are present on-site.

Depending on the kind of application, the users that are present in the system are likely to have different roles, with each of them being tied to certain (access) rights, duties and responsibilites. According to this role model, upcoming tasks are hence to be allocated to and carried out by different subsets of users. In a smart environment, however, not only the creation of new tasks might well be triggered autonomously by advisedly interpreting sensor data, but also the decision of which user to select out of the set of matching users can be taken automatically by the system. In our vision this decision-making can, for instance, be based on additional information such as the eligible employees' current positions, activities and previously assigned tasks. Aiming primarily at a site's habitual users and not at navigation-seeking first-time visitors, the focus of our platform is hence not on indoor navigation, but rather on leveraging context and location information for an optimized allocation of event-triggered tasks to human users. Therefore, however, location modeling, positioning, tracking and routing mechanisms are naturally of great importance here, too, just as is the case for the majority of location based services (LBS).

As an example scenario, imagine a hospital staffed with doctors, nurses and technicians, as well as cleaning and maintenance personel, who are all equipped with a personal mobile computing device. Additionally, there is a

number of movable apparatus, e.g., for medical emergencies, whose current status and positions are monitored by the system as well. When a patient's infusion system reports a failure, the system queries its context model for the ward's nurses' current whereabouts and activities, as well as their ongoing tasks. Based on the calculated distances to the target location and the task's priority, the best matching nurse is selected. Without her colleagues being interrupted in their ongoing taks and without the ward's patients being disturbed by any acoustic alarms, the selected nurse will be informed about the problem on her mobile device and can take care of it. In another wing of the building several light sensors detect the failure of ceiling lights and report their findings to the system. This time, in order to minimize the amount of extra effort induced by the task, the system now selects one of the maintenance men who is already heading to the depot and sends a new ticket concerning the replacement of the light bulbs to his mobile device. In both cases, the *SmartBEEs* platform is hence able to find the best match among all employees, respectively, thereby optimizing the overall response times and increasing productivity.

The remainder of this paper is organized as follows: Section 2 presents a brief overview on related research approaches. In Section 3 we discuss the requirements that the *SmartBEEs* platform has been designed according to. Section 4 describes our proposed system architecture and its most important components. In Section 5 we give an introduction to our current proof-of-concept implementation, before Section 6 finally concludes this paper.

2 Related Work

In this section we will give a brief review over existing approaches aiming at the creation of location based services and context-aware applications in business environments. There are already several works that especially target on bringing ubiquituous computing mechanisms to business environments such as, e.g., manufacturing sites, factories and hospitals. [WJ03] describes a smart factory focusing on the decentralized and fully automated management of mobile resources in a manufacturing environment. The authors present the vision of reducing a factory's tool costs and raising the efficiency of mobile resources usage by having all kinds of resources communicating with their environment about their current status and location. This approach is followed up in [BJS04] and [JWN04], which also focus on the management and coordination of production resources and tools in smart factories. While integrating well into existing business management processes and tool chains, none of these approaches takes into consideration the contexts, tasks and locations of the system's human users.

Concentrating on human task execution in manufacturing processes, Wieland et al. describe mechanisms for the usage of explorative applications for the distribution of tasks to workers in production environments [WLJ+06]. For this purpose, the authors utilize the concept of so-called *virtual task containers* (*VTCs*), which can be placed at specific locations and contain information about upcoming tasks, which will be presented to workers when entering corresponding areas. The system is able to automatically create these tasks, e.g., by monitoring sensor readings or by following maintenance schedules. In [WNL11] the authors present a slightly different solution by displaying current tasks on a work-map on the workers' mobile devices in a context-aware fashion. Both the task's target location as well as the positions of tools and materials needed for the task's execution are shown on the map. Workers can thus choose which task in their vicinity they want to take care of. With the goal of trying to preserve the users' privacy, however, the system itself does not take into consideration the positions of human users and is hence not capable of performing any reasoning processes based on the system's overall state and its users' current contexts. [WKNL07] introduces the concept of context-aware workflows in order to bridge the gap between business and production processes. [WLS+10] puts a special focus on using such workflows for improving a smart factory's failure management and also introduces different priorities based on a task's urgency. [LW07] presents a context model that has especially been designed to fulfill the requirements for modeling manufacturing environments, which is able to model workflows, resources and sensors. Although being very sophisticated and mature concerning the integration of existing standards such as BPEL [Org07], however, all of these works mostly neglect the users' current contexts from the system's point of view, and are thus also unable to perform the kind of reasoning and decision-making described in the motivating scenario.

Fuhrer et al. have investigated mechanisms for building a smart hospital based on RFID technology [FG06]. Their system aims at optimizing business processes in healthcare, reducing errors and improving workflows and the patients' safety by means of patient identification, tracking and identification of blood transfusion bags, smart operating theaters, as well as tracking patients, staff and equipment using RFID. However, the system neither takes

into account any other types of sensors, nor the users' tasks or contexts, and is thus not matching our scenario. In connection with the *iHospital* project, Sanchez et al. have investigated mechanisms for staff activity recognition that can be used in order to enable context-aware communication, personalized information retrieval and multi-tasking [STF08]. Therefore, the authors have manually gathered almost 200 hours of documented observational data, which were used to train a Hidden Markov Model (HMM) for recognizing prevalent activities. The results of activity recognition can then be used, e.g., for gaining availability information in order to negotiate interruptions at appropriate times. The *iHospital* yet lacks the ability of automatically creating tasks based on sensed events and thus also does not leverage these context information for an automated assignment of upcoming tasks to its users in the way we envision it for the *SmartBEEs* platform.

Coronato et al. present a semantic location model for the integration of a variety of positioning systems as well as reasoning mechanisms capable of logically combining location estimations from different positioning systems [CEP09]. The authors distinguish between semantic and physical locations and enable a mapping among these onto each other. In order to demonstrate the system's practicability, several use cases for another smart hospital scenario are presented. Naturally, with each of the latter concentrating on entity identification and location-aware information retrieval only, that approach does not even aim at the detection and distribution of tasks among the hospital's staff and is hence also not matching our use case.

3 Preliminary Considerations and Requirements

We will now discuss the main challenges and requirements that have to be taken into consideration for the design of the *SmartBEEs* architecture in order for the platform to be able to reach the goals outlined in the motivating scenario and at the same time allow for extensibility and applicability in different kinds of environments.

Sensor integration and abstraction In order to offer a maximized degree of interoperability to possibly existing hardware and software systems, the platform should support the integration of different types of sensors, bus systems and mobile devices. Additionally, the system should introduce an easily manageable level of abstraction, so that application developers and system operators do not have to care about the low-level details of sensor integration or the heterogeneities of communication protocols and data formats. Obviously, the platform should also allow for the modification of the set of deployed sensors, as well as their positions, status and calibration parameters at runtime and has to remain operative in case of sensor node failures.

Modeling roles, duties and timetables According to their expertise and position, employees (i.e., users) are likely to be assigned to different roles, responsibilites and authorizations. Hence, the system has to provide a simple means for defining, managing and altering both the specification and the assignment of roles and corresponding usage or access rights to users. Consequently, an assignment of duties and associated tasks to roles has to be rendered possible. Apart from that, the platform should also support a notion of time and temporal role assignments, such as different times of day, different days of a week, opening hours and working shifts, etc. in order to enable an adequate mapping of real world constraints and conditions.

Location modeling, positioning and estimating distances Like any other location based service, our system needs to store different types of information about its site of deployment and the entities present in the covered area in some kind of location model. In order to be able to efficiently process different kinds of requests, the location model should at least allow for position queries as well as nearest-neighbor queries. These queries should be processed using effective distances as caused by obstacles such as walls, e.g., derived from actual floorplans. In addition, the location model should support the specification of meta information about locations, such as access rights or restricted accessibility, e.g., for employees steering a pallet truck or users of a wheelchair.

As far as positioning is concerned, the platform should be able to handle multiple types of positioning subsystems at the same time, both for outdoor and indoor positioning, in order to enable a maximum degree of coverage, possibly at the whole site. Therefor, the platform's positioning module should be able to fuse location information from

different sources, translate between diverse data formats and resolutions as well as between different coordinate reference systems in order to create a uniform and meaningful representation of locations and positions. For the process of determining and selecting the best-matching users for a given task at a given point in time, the platform must be able to compute near real-time route length estimations for all eligible users to a given target location. It is worth noting that these routes may consist of several hops, as could be seen in the motivating scenario with the ceiling lights. Moreover, also the current positions of movable objects, such as tools and appliances needed for a task's execution, as well as access control policies have to be considered in the route length calculations. For our use case, though, an algorithm that merely determines the lengths of valid routes would be sufficient, since the routing result will not be used for navigation, but simply serves as a location-motivated hint for finding an optimal match from the set of appropriate users. For reasonably guiding users, however, there is an additional need for mapping geometric locations to user-friendly symbolic ones, which can be presented as target locations to users.

Data interpretation, rule and task modeling In order for the desired platform to be functional, one of its key features can be seen in the rule-based interpretation of sensor data readings. Therefor, there has to be a reasoning component, which is capable of continuously analyzing incoming sensor data in order to decide whether any actions have to be taken. Consequently, it must be possible to either manually or automatically train the system with information about certain system states and situations that trigger a new task. In most cases, it will be possible to derive these kind of facts from existing process descriptions or by observing actual workflows. In order for the system to effectively assign tasks to its users, there is also a need to model all known tasks together with their priorities, properties and dependencies. A task may consist of several subtasks that have to be executed in sequence one after another. Both tasks and subtasks should feature an estimated time duration and might either be defined to be atomic, indicating that this (sub)task has to be completed at a stretch, or interruptible, stating that a (sub)task might be postponed for the execution of a higher prioritized task. For the sake of simplicity, however, only subtasks taking place at different locations have to be modeled explicitly, whereas a sequence of real-world subtasks appearing at one location can be considered to be one single subtask from the system's perspective. Hence, for non-atomic tasks, each location change should be modeled as an independent subtask at least. A task's priority should furthermore be regarded as an important input parameter for the process of selecting a matching user, since depending on a task's urgency the system should be able to either opt for quick response times or little additional time and effort, otherwise. Finally, in order to be effective, the platform should also provide mechanisms for detecting task duplicates and for realizing the successful execution of a task, even in case of an unsolicited performance by any person other than the selected user.

Modeling context and historical data The system also has to provide a means for modeling and managing its current overall context, as well as allow for efficiently storing and retrieving all kinds of historical data from different levels of abstraction, e.g., raw sensor readings, position traces and past contexts of users and movable objects as well as statistics about issued tickets and response times. These data can be useful, e.g., for time-series based event recognition, sensor calibration or auditing processes. A challenge here is to find efficient mechanisms for reducing the size and amount of data to keep without overly sacrificing resolution of the stored information.

Non-functional requirements Finally, the platform also has to fulfill a number of non-functional requirements, such as security, reliability, scalability and usability [CdPL09]. It must be rendered infeasible, e.g., for unauthorized parties to inject, capture, replay, modify and remove tasks or to manipulate the process of user selection. The system must also be both reliable and safe and should be scaling to hundreds of users, sensors and arbitrarily sized locations. After all, the system should be easy to operate and use, especially for the end users, who should be supported in their daily work by the system, and not be burdened with additional, usability-related obstacles. Furthermore, other non-functional concepts such as generality, extensibility as well as modularity and substitutability also have to be kept in mind for the design of the platform's architecture. This is to guarantee that the proposed system is not bound to a fixed set of possible areas of deployment, but open for adaptations, modifications and extensions in order to be applicable to a wide range of applications, environments and usage scenarios.

Given these requirements, which obviously do not cover a single topic only, but rather a wide field of ongoing research, we will present our system's architectural design and describe some of its most important modules and components in the next section.

4 System Architecture and Components

As depicted in Figure 1, the *SmartBEEs* system architecture mainly consists of two loosely coupled, vertically aligned layers offering different functionalities and levels of abstraction to application developers and system operators. Each layer is composed of several modules providing specific services, whose actual implementations are substitutable by any components offering the required functionality. Depending on the data format of the available location information, for instance, different location models or routing algorithms might be implemented. This replaceability is realized by explicitly defining the interfaces a specific module has to provide at least. At several points the system also allows for the concurrent usage of more than one implementation in the form of submodules, thereby enabling, e.g., hybrid usage of different location models or positioning systems at once.

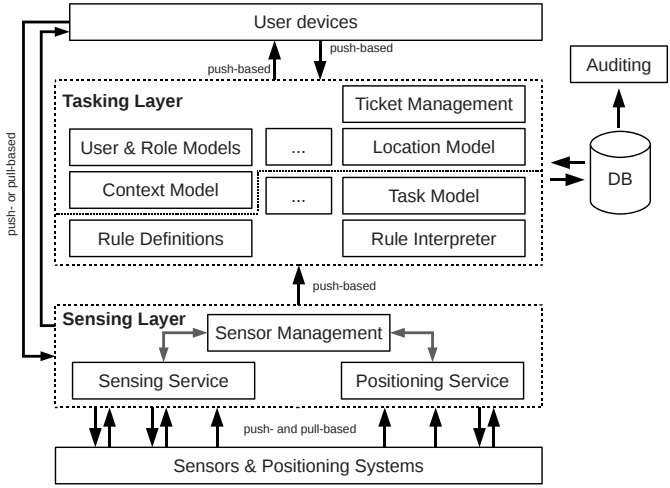


Figure 1: The layered architecture of the *SmartBEEs* platform and its most important components.

The lower placed *Sensing Layer* (*SL*) is responsible for managing and monitoring the total number of deployed sensors by collecting all kinds of available sensor data readings. By transforming these readings from raw data streams and heterogeneous data representations into a well-defined output format, the *SL* serves as an abstraction layer for the different types of sensors and sensing subsystems connected to it. Naturally, also the location information of users and movable objects – probably obtained by more than one positioning system – are collected and prepared for upper-layer processing here. Eventually, this layer also standardizes the inhomogeneity of push- and pull-based sensor communication by uniformly making use of a simple push-based interface for providing the upper layer with freshly acquired sensor data. The *SL* thus insulates the upper layer and its modules from coping with the heterogeneities of the employed sensor hardware, communication protocols and data formats.

All application logic – such as the role and location models and the necessary reasoning functionality – is located at the *Tasking Layer* (*TL*). The latter is hence responsible for interpreting sensor readings, evaluating location information and eventually making well-grounded decisions based on these context information and a set of predefined rules, e.g., derived from business process descriptions or based on established work routines and best practices. The *TL* therefor also knows all users and their responsibilites, as well as their current contexts and assigned tasks. Each time the *SL* reports fresh sensor data readings to the *TL*, the incoming data will be interpreted and matched against the known set of rules, what might result in the creation of a new *event-triggered task* (*ETT*). Apart from that and due to the fact that all sensor data readings are being pushed by the sensing layer, the *TL* is

also responsible for storing all incoming sensor data as well as all users' and movable objects' location traces in its database for later use. Finally, the tasking layer also contains modules for managing tickets and for handling the application-level communication with the users' mobile devices, which are placed on another loosely coupled and thin, third layer. All kinds of sensor data related communication involving these mobile devices, however, such as a smartphone updating its own position estimation or battery status, is taken care of by the sensing layer.

The main benefit of the proposed system architecture can be seen in the fact that the sensing layer does not have to be aware of the possible meanings and interpretations of sensor data and reasoning about combinations thereof, whereas the tasking layer does not have to care about how these low level context information can be collected. In the following sections, we will describe the two different layers' most important modules in more detail.

4.1 Low-level Communication Handling and Processing of Raw Sensor Readings

The *SL* mainly consists of the three components shown in Figure 1, which are being composed in order to altogether provide an adequate and easily manageable level of abstraction from the mostly inconvenient and cumbersome details of sensor communication and data formats for the upper layer.

In this connection, the *Sensor Management* module is responsible for keeping the accounts of the total number of deployed sensors along with all relevant information regarding a sensor's identity and type, initial location, desired update rates, as well as calibration parameters, available data formats and communication interfaces. This also is the place where both individual sensors and the whole set of active sensors can be maintained and modified, e.g., by relocating an existing sensor or by integrating new sensors into the system. Sensors of a known type can easily be added to the platform at runtime by simply creating another instance of the corresponding sensor type. This process is facilitated by the module offering a simple import mechanism for sensors of known type and with compatible communication interfaces. For this purpose, sensors and their attributes can be modeled using a predefined XML structure. The information that have at least to be specified for the automatic integration of a sensor S of known type T_S can be represented by a vector $D_S = (id_S, T_S, interface_S, dataformat_S, calibration_S, updateRate_S)$, which the sensor manager will try to interpret in order to automatically create and register the described sensor in the system. On the contrary, the integration of previously unknown and unimplemented sensor types or communication interfaces possibly requires modifications both to the sensing layer's modules' code and hardware setup, which at present cannot be performed at runtime. However, with this act being expected to be something of a rarity in the course of normal operation, we consider this approach to be practical.

Being controlled and monitored by the just described sensor manager, the *Sensing Service* module is responsible for collecting all kinds of sensor readings from the available sensors. Depending on the types of deployed sensors, this module is hence listening for sensor data readings that are automatically being pushed from the sensors, as well as periodically querying pull-based sensors for current readings. Based on the sensor descriptions stored in the *Sensor Management* module, this component is able to communicate with all kinds of sensors, pre-process and optionally calibrate their raw measurements before translating them into a simple and uniform XML-based data format, which will then be passed to the tasking layer for further interpretation and rule evaluation. A pre-processed reading of sensor S at time t_j can thus be described as a vector $R_{S_{t_j}} = (id_S, value_{S_{t_j}}, t_j)$. Hence, by translating raw data streams into a semantically meaningful representation and a well-defined output format, this module is constituting the abstraction layer that is needed for handling the different kinds of sensors connected to it in a uniform way, just as required in Section 3.

The tasks and functionalities of the *Location Service* module are quite similar to those of the *Sensing Service*, with the sole exception that this module handles location and position updates only. With position information probably being the most important – and also the most prevalent – type of context information for all location based services, we consider it useful to have a particular focus on this kind of information in our architecture. The platform might thereby prove beneficial even in environments without any sensors being deployed and user-triggered tasks only. Just as the sensing service, the location service is likely to be communicating with and gathering position updates from a multitude of data sources. Depending on the types of positioning systems deployed, different providers of location information might either report their own locations (e.g., a GPS-enabled smartphone) or supply positioning results obtained for other entities, such as terminal-assisted or network-based positioning systems do (see

[LDBL07] for a survey on positioning techniques). In the latter case, hence, location measurements for different entities might well be originating from one and the same location sensor. In order to distinguish between the tracked entities, usually some kind of system specific entity identifiers exist, e.g., such as the Ubisense tag ID. In the *SmartBEEs* platform, these identifiers are hence being used for enabling an unambiguous mapping of position updates to users and objects on the tasking layer. The sensor manager therefor creates separate instances of individual, virtual positioning sensors for monitoring and reporting the location of a single entity. The corresponding unique sensor identifiers needed for distinguishing between entities can be generated, e.g., by concatenating the actual sensor id with the respective entity identifiers. The actual sensor id thus refers to the positioning system itself, whereas the concatenated entity id states which entity has been tracked. Based on these information, the upper layer is able to correctly assign location updates to the entities associated with a given set of identifiers in order to integrate these information into its location model and use it for its distance estimations.

4.2 Interpreting Sensor Data and Creating Tasks

Once the *Sensing Layer* has transformed the raw sensor data readings into a semantically enriched and uniform data format, the now structured data will be pushed to the *Rule Interpreter* module of the tasking layer. By matching incoming sensor readings against a set of pre-defined rules stored in the *Rule Definitions* component, the interpreter can decide whether the new condition requires any actions – such as the creation of a new task – to be taken. For instance, the latest reading of a sensor value may indicate that a configured threshold has been exceeded. A corresponding rule might hence demand to send a ticket to an appropriate employee, who can take care of the issue. In many cases, however, not only the incoming sensor data has to be evaluated, but also the currently known overall state of the system, i.e., the system's context has to be taken into consideration. This is necessary given the fact that, e.g., the priorities and responsibilites of certain tasks might change in different overall conditions. Therefor, the interpreter is also likely to query the database for accessing both time-series data for the same sensor, as well as query the context model in order to be able to factor all necessary context information into its decision process. All rules whose conditions match the current situation will be executed, meaning that the respective actions they require to be taken will be presented to the *Ticket Management* module as upcoming tasks.

4.2.1 Modeling and Processing of Rules

In order for the pre-defined rules to be interpretable by both human administrators and machines, the *SmartBEEs* platform envisions simple rule definitions in XML. A rule definition always consists of an unique identifier, a canonical name for easy identification by human users, a set of conditions and relations among these, and finally a set of actions that are to be triggered in case the combination of the rule's condition expressions evaluates to true. Conditions can be expressed using relational operators such as $equals$, $lower$ and $greater$, e.g., in order to describe critical limits, thresholds or different states. Here the identification of sensors and the correlation of sensor data to its sources is realized by making universal usage of the sensor IDs stored in the *Sensor Management* component throughout the different layers and components of the platform. Temporal constraints can be added to conditions in order to define different system behaviors, for instance, based on the duration or repetition rate of certain states. Furthermore, single conditions can be arbitrarily combined with each other by using logical operators such as and, or and not. During the process of matching sensor data readings against the set of rules, the interpreter first filters out those rules that are completely unaffected by the current sensor. From the remaining set, the interpreter removes all rules whose conditions do not correspond with the value of the current sensor data in a second step. The still remaining rules are being ordered descendingly by their corresponding tasks' priorities and will then be sequentially investigated for exact matches in order to find higher prioritized tasks first. Missing data readings from other sensors and context information can be retrieved from the database and will be cached in order to enable efficient access while validating subsequent rules. Eventually, each rule that is detected to be an exact match is likely to cause the *Rule Interpreter* module to create a new task of the required type and corresponding parameters, which will then be sent to the *Ticket Management* component for distribution (see Section 4.4). At this point it is worth mentioning that there might also be rules that demand the cancelation of an active task, e.g., in case that relevant sensor readings switch back to normal, thereby indicating the successful execution of a task.

4.2.2 Modeling Tasks and Entities

Similar to the definitions of rules, our platform allows for tasks to be modeled using XML files. Hence, task descriptions, too, always contain an unique identifier and a canonical name, as well as a list of user roles that are associated with this kind of task, the task's description and an integer value indicating its priority. The task identifier is used in the rule descriptions' actions for unambiguously referencing a certain task. A task's description either holds a concise definition of the task itself or a list of subtasks. This recursive definition presents a simple means for creating more complex tasks that are taking place at more than one location. The explicit description of a (sub-) task at least consists of its location, a hint to its estimated time duration and a list of entities assigned to this task. The task of transporting goods from location A to B, e.g., might require using a pallet truck that first has to be collected from its current position before moving to location A. This can be expressed using two subtasks, namely "*moving to A with a pallet truck*" and "*transporting goods to B*", indicating that both the pallet truck's and the goods' locations have to be visited in this order. Furthermore, a task can either be marked as $atomic$, meaning that its completion must not be interrupted by other tasks, or non_atomic, stating that the execution of other tasks – possibly such with a higher priority – might suspend a user's current task. In case a parent task has been marked as *atomic*, this value naturally recursively overwrites all of its child elements' values.

The different types of entities present in the *SmartBEEs* platform can be classified into three groups, i.e., *static*, *movable* and *mobile*, with only the latter being able to move around autonomously. The locations of static objects, such as infrastructure components and stationary appliances are stored in the database and might be updated manually, if necessary. Position updates for all movable and mobile objects, on the contrary, are being provided by the previously described *Location Service* component of the sensing layer. For the sake of simplicity and consistency, we define all users of the system to be *mobile*, whereas tools and materials, appliances, vehicles, etc. are classified as being *movable*. Only the latter can be referred to as associated entities in a task's description, while *mobile* entities are the only ones to be commissioned with tasks. Entities can have different states, such as being available, busy or in use. Additionally, based on an ongoing task's duration estimation, the system can approximately determine the times when a busy user will be available again or when ressources in use will be freed up.

4.3 Location-Based and Context-Aware Allocation of Tasks

In case the just described process of rule interpretation triggers the creation of a new task, in the next step the decision of which subset of users will be confronted with this *event-triggered Task* (*ETT*) has to be made. This decision can, for instance, be based on the current assignment of tasks to users, the contexts and locations of all eligible *mobile* and all relevant *movable* entitites, as well as the task's target location. Depending on what kind of goal shall be achieved and the *ETT*'s priority, the *SmartBEEs* platform at present allows this selection process to be realized using three different methods. In the following sections, we describe the underlying location model and the mechanisms for context- and location-aware user selection and task allocation in more detail.

4.3.1 Location Modeling

In order to be able to make decisions based on up-to-date positioning information and distance estimations therefrom derived, a LBS needs to have a clear model of its area of deployment and the latter's topology. Typically, context- and location-aware applications are therefor making use of some kind of location model, which usually holds information about a site's geometry or symbolic location identifiers, or any combination thereof (cf. Section 2). For the *SmartBEEs* platform, we decided to make use of a so-called hybrid location model, combining the advantages of both symbolic and geometric approaches. On the one hand, the geometries of locations and the exact topology of a site are required for routing and distance estimations, e.g., based on Euclidian distance. On the other hand, we are in need of symbolic location identifiers, too, in order to inform users about target locations in a user-friendly format. Using a hybrid location model solves this problem by allowing for a mapping of geometric coordinates, for instance, given in the WGS84 format or any local coordinate system, to semantically meaningful location identifiers, such as "*Room E 004*", that people who are familiar with a place will easily understand. Our location model features a graph-based approach for its symbolic location identifiers, with the graph's vertices

representing rooms, hallways and outdoor spaces, the edges representing interconnections (e.g., doors, stairs or elevators) between them. Both types of elements in the graph can be marked with meta information such as accessibility and access rights, possibly indicating that not all users in the system are allowed or able to use them. Returning to the previous example, a person transporting goods on a pallet truck will not be able to use stairways. By this, one can easily integrate overall access rights and accessibility information into the proposed architecture, while using a single location model for all users. Thus, the location model itself is context-aware, meaning that, e.g., the validity of meta information stored in the model can change according to a user's current state or the system's overall context. Different subsets and combinations of meta information might, for instance, be valid for normal operation and emergency situations, or day- and nighttime, respectively. Each node in the graph can be assigned with geometric information about its shape and extent, thereby enabling fine-grained distance estimations. Finally, in order to be able to cope with different positioning systems, the *Location Model* component can be extended by an arbitrary number of submodules that are capable of projecting a specific positioning system's coordinates onto the model's coordinate system. Based on the unique sensor identifier provided with a position update, the information can automatically be routed to the corresponding submodule, which will perform the translation between the possibly different coordinate systems.

4.3.2 Context-Aware User Selection

Each time an upcoming task raises the necessity to issue a new ticket, the system has to determine whether it is a duplicate of an already assigned task and – if not – decide which users to select for being confronted with it. In order to be able to make a reasonable choice, the *SmartBEEs* platform takes into consideration several aspects, such as the list of already assigned tasks, the new task's urgency and target location as well as the contexts and positions of relevant entities. Given that the new task is not an exact duplicate of an already ongoing task, the system determines the static set of eligible users in a first step. From this set of users with authorizations and duties matching the current task now the best matching user has to be found. In order to do this, the *Ticket Management* module queries the location model, providing as input parameters the set of eligible users, as well as the task's priority and an identifier for the selection mechanism to be used. Depending on pre-defined policies the process of how to select a matching user can be based on different decision algorithms, e.g., according to a task's urgency.

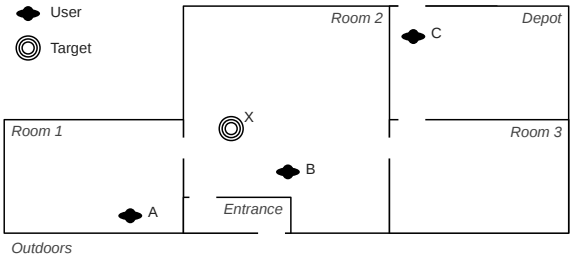


Figure 2: An example setup of the *SmartBEEs* platform for a combined indoor and outdoor region with three registered users A, B and C. Obviously, B is placed next to the target location X. However, based on additional context information, the overall state of the system and the urgency of the new task, another user might be selected for the task.

In Figure 2 a simple example scenario with three users and the target location of a new task is shown. Let us assume that user B is already working on an atomic task with an estimated remaining duration d_B, whereas users A and C have not been assigned to any tasks yet. In the simplest case the system opts for selecting the user whose current position is the closest to the target location (*minDistance*). Hence, B will be chosen without taking into consideration her ongoing task, for instance, in order not to disturb other users in their routine working. However, if the new task appears to be time-critical, the system will rather choose the *minExecutionTime* algorithm in order to select the user who is the most likely to be able to react in the shortest time possible. If this is the case, the system determines whether the expected times $d_{\overline{AX}}$, $d_{\overline{CX}}$ it takes A and C to reach the target location, respectively, are

lower or greater than $d_{\overline{BX}} + d_B$, and chooses the minimum. Therefor, the system is able to roughly estimate durations from calculated distances based on an average pedestrian walking speed. On the contrary, if the task is not time-critical at all, the system will aim at minimizing the overhead resulting from different users executing the task. For this purpose, the *minExtraRoute* mechanism can be used, which calculates alternative routes for all eligible users and compares these in order to select the one with the minimum additional traveling distance. Furthermore, the static and movable entities assigned to a certain task might influence the result of user selection as well. Therefor, the current location of a required entity will be inserted as a waypoint for the route length estimations. In the example above, for instance, if executing the task required material from the depot, user C would be chosen using all three alternative mechanisms. Eventually, the ID of the selected user will be returned to the *Ticket Management* module, which initiated the query.

4.4 Ticket Management and Machine-to-Human-Communication

Eventually, the user that has been selected will receive a ticket on her mobile device containing all the information needed for executing the task. In order to be easily interpretable by human users, each ticket at least displays the name and a description of the task, as well as the task's location and priority. Amongst other things, the task's description also contains information about the entities associated to the task, such as their id, name, state and current location. In addition, given that the user is already busy performing another non-atomic task, the ticket may also contain a hint to whether the ongoing task should be suspended in favor of the new one or not. In case the task consists of several subtasks at different locations, both the task's overall objective as well as the list of subtasks is presented to the user. Upon receiving a ticket on her mobile device, the user will be prompted whether she is able to accept the ticket or not. This is in order to take account of the user's ongoing tasks, goals or activities that the system has not been aware of. If the user accepts the ticket, the *Ticket Management* component will change the ticket's lifecycle state to *inProgress*, otherwise the ticket will be forwarded to the second best matching user. This also happens when the selected user does not respond to the ticket prompt within a certain time limit depending on the task's priority. At this point in time, the ticket's state remains being set to *open*. Tickets can also be aborted by users after accepting it as well as be withdrawn by the ticket manager, e.g., caused by the *Sensing Layer* reporting yet another change of state that leads to the cancelation of the corresponding task. In case the execution of a task is being aborted before completion, again another user will be selected. Once the ticket has been processed successfully, the ticket's state is set to *completed* and a record will be kept in the database. The dynamic set of active tickets (i.e., those with lifecycle states prior to *completed*) is being used for determining whether a new task issued by the *Rule Interpreter* component is a duplicate of an already created ticket in order to prevent any inefficiencies due to multiple assignments of the same task.

In case the system detects the successful completion of a task, the *Ticket Management* component will set the ticket's lifecycle state to *completed* and notify the respective user on her mobile device accordingly. This is an important aspect since it might happen, e.g., that a task has been completed unsolicitedly by someone else before the selected user arrives at the target location. Depending on the kind of task and the types of sensors involved in monitoring and triggering it, the execution of the task can either be detected automatically by interpreting all-clear sensor readings or by a human user notifying the system about the task's execution.

Apart from simply displaying tickets and offering a basic feedback channel as described above, however, the mobile devices of users can also be used for the creation of *user-triggered tasks (UTTs)*. As already stated before, the smartphone might autonomously report sensor data readings to the *Sensing Service* component, such as its location, lists of devices nearby and its user's current activity. Naturally, this kind of information will be reported to the platform's *Sensing Service* component (cf. Section 4.1). Beyond that, a user might yet also want to manually create a task and send it as a *UTT* to the *Ticket Management* module. This is especially important for tasks that the deployed sensors might have missed or are not even able to detect. In order the facilitate this process and allow for unambiguous task descriptions among all users, the mobile devices are able to support their users in putting together a task by presenting context-aware suggestions from pre-defined sets of task descriptions and locations.

In this section we have given a detailed introduction to the *SmartBEEs* system architecture along with its different layers and its most important components. The next section describes our testbed environment and experimental setup as well as the current state of our prototypic implementation of the platform.

5 Prototypic Implementation

In order to demonstrate the proposed architecture's feasibility we have developed a prototypic implementation of the *SmartBEEs* platform, which we are continually improving and extending. Not all of the features described in the previous sections can be considered fully functional yet, but the architecture's core functionality and its components interplay has already been successfully tested and demonstrated.

At the time of writing the different kinds of sensors deployed in our setup range from simple electrical and resistive sensors, such as temperature and door contact sensors, to smoke detectors, digital scales and off-the-shelf smart appliances. The resistive sensors in use are connected to the system using Arduino boards and a Azeti Sonargate sensor hub. The digital scales are connected over a serial port to a laptop acting as a proxy, whereas the smart appliances' status can be queried using a RESTful web service interface provided by a corresponding gateway. Multiple instances of each of these sensors exist, which are placed at different locations in our testbed environment. As described in Section 4.1, XML descriptions were created for each type of sensor. Hence, new instances of these sensors can easily be added to (or removed from) the system at runtime with only the instance specific parameters such as the location of deployment being modified, thereby demonstrating the *Sensing Layer*'s ability to hide the heterogeneities of data formats, sensor interfaces and communication protocols. The tracking of movable and mobile entities is currently based on the Ubisense ultrawideband positioning system [SG05] for an indoor area and GPS for the adjacent outdoor region. The setup depicted in Figure 2 is actually showing our real testbed environment in Munich. For indoor areas that are not being captured by the Ubisense system, i.e., rooms 1 and 3 and the depot, we are therefor making usage of symbolic location identifiers based on the doorway an entity most likely used when leaving the Ubisense enabled area. For the sake of simplicity, these semantic location identifiers are linked to the center coordinates of the respective room.

The *Tasking Layer* offers a RESTful web service interface, which is used by the *Sensing Layer* for pushing sensor data readings and position updates as structured XML data using HTTP PUT requests. Each incoming sensor data will be checked against the pre-defined set of rules. The definition of these rules, as well as tasks, users, roles and timetables have to be created manually in an XML format. The following piece of code shows an example of a very simple rule definition in XML. More complex rules involving several sensors exist as well.

```
<rule>
  <id>r_17</id>
  <name>Fridge door opened</name>
  <condition>
    <equals sensor="s_13" value="0" duration="240s"/>
  </condition>
  <action id="t_17"/>
</rule>
```

The given rule is simply stating that in case sensor s_{13} – which is a door contact sensor mounted to a fridge – reports the fridge's door to be permanently opened for a period of more than 240 seconds, the task with identifier t_{22} should be triggered. At the moment, there are about 20 rules modeled in our prototype, which can be modified and extended at runtime, too. If the current system state matches a rule's condition, the corresponding task will be presented to the ticket management component, which will create a new ticket and assign it to a user based on the present users' roles, current locations and availability information.

At the moment, the implementation of the location model described in Section 4.3.1 has not yet been fully completed. Instead, our current prototype relies on a bitmap based floorplan as well as a simplified location model that at least allows for the mapping of symbolic location identifiers to geometric coordinates and vice versa. Hence, for estimating distances between entities and given target locations, our prototype is at present using a pixel-based implementation of the A* pathfinding algorithm [HNR68]. This approach was selected to demonstrate the platform's feasibility only and is performing well for the limited geometric extent of our testbed, but will be replaced by a more sophisticated and scalable alternative based on the proposed location model's graph structure.

The mobile client is implemented as an Android application, which consists of a background service providing the *Sensing Layer* with GPS updates and a set of activities displaying tickets and allowing for the creation of user-

triggered tasks. Upon reception of a new ticket, a user can choose whether to accept or decline a ticket, which will be handled by the ticket management component respectively. In case the system is unable to infer the execution of a task automatically, the user can mark a ticket as completed on her mobile device.

6 Conclusion and Outlook

This paper introduced the *SmartBEEs* platform, which aims at facilitating the integration of location- and context-awareness of infrastructure components, movable objects and mobile users in a smart environment in order to enable both an automatic detection and a reasonable assignment of upcoming tasks to mobile users. For this purpose, first a motivating scenario in a hospital environment and a list of requirements for such a platform have been presented, which cover the whole range from sensor data acquisition to the management of tickets. We then described the layered design of the *SmartBEEs* system architecure as well as the functionalities of and interrelationships between its most important components, before eventually presenting the setup and current state of our prototypic proof-of-concept implementation.

As for our future work on this topic, we intend to investigate mechanisms for staff activity recognition in order to increase the platform's degree of context awareness. Another interesting question is how to automatically derive ECA-rules for event detection and actual task descriptions from existing process models and observable workflows. So far, we have also left out all kinds of privacy aspects, which we are greatly aware of, though. The continuous tracking of employees, for instance, is both a moral and legal concern that will have to be taken care of in form of a privacy preserving solution. Moreover, we would like to conduct a user study for gaining information about the applicability and the acceptance rate of such systems in professional environments. Eventually, a number of performance related issues will have to be tackled as the numbers of sensors, users and rules in the system increase.

References

[BJS04] Martin Bauer, Lamine Jendoubi, and Oliver Siemoneit. Smart Factory - Mobile Computing in Production Environments. In *Proceedings of the MobiSys 2004 Workshop on Applications of Mobile Embedded Systems (WAMES 2004)*. WWW, June 2004.

[CdPL09] Lawrence Chung and Julio do Prado Leite. On Non-Functional Requirements in Software Engineering. In Alexander Borgida, Vinay Chaudhri, Paolo Giorgini, and Eric Yu, editors, *Conceptual Modeling: Foundations and Applications*, volume 5600 of *Lecture Notes in Computer Science*, pages 363–379. Springer Berlin / Heidelberg, 2009.

[CEP09] Antonio Coronato, Massimo Esposito, and Giuseppe Pietro. A multimodal semantic location service for intelligent environments: an application for Smart Hospitals. *Personal Ubiquitous Comput.*, 13(7):527–538, October 2009.

[FG06] Patrik Fuhrer and Dominique Guinard. Building a Smart Hospital using RFID technologies. In Henrik Stormer and Andreas Meier, editors, *Proceedings of the 1st International Workshop on eHealth (ECEH06)*, volume P-91 of *Lecture Notes in Informatics (LNI)*, pages 131–142. 1st International Workshop on eHealth (ECEH06), GI - Gesellschaft für Informatik e.V., October 2006.

[HNR68] P.E. Hart, N.J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100 –107, july 1968.

[JWN04] Lamine Jendoubi, Engelbert Westkämper, and Jörg Niemann. The smart factory - pervasive information technologies for manufacturing management. In Universität Karpatz, editor, *Machine Tools and Factories of the Knowledge*, pages 13–20, Karpazt, Poland, April 2004. Cirp-Verlag.

[LDBL07] Hui Liu, H. Darabi, P. Banerjee, and Jing Liu. Survey of Wireless Indoor Positioning Techniques and Systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067 –1080, nov. 2007.

[LW07] Dominik Lucke and Matthias Wieland. Umfassendes Kontextdatenmodell der Smart Factory als Basis für kontextbezogene Workflow-Anwendungen. In Jörg Roth, Axel Küpper, and Claudia Linnhoff-Popien, editors, *4. GI/ITG KuVS Fachgespräch Ortsbezogene Anwendungen und Dienste*“, pages 47–51. Dr. Hut-Verlag, September 2007.

[Org07] Organization for the Advancement of Structured Information Standards (OASIS). *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*, April 2007.

[SG05] Pete Steggles and Stephan Gschwind. The Ubisense Smart Space Platform. *Adjunct Proceedings of the Third International Conference on Pervasive Computing*, 191:73–76, 2005.

[STF08] Dairazalia Sánchez, Monica Tentori, and Jesús Favela. Activity Recognition for the Smart Hospital. *IEEE Intelligent Systems*, 23(2):50–57, March 2008.

[Wei91] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265:94–104, 1991.

[WJ03] Engelbert Westkämper and Lamine Jendoubi. Smart Factories - Manufacturing Environments and Systems of the Future. In Germany Saarbrücken, editor, *Proceedings. 36th CIRP International Seminar on Manufacturing Systems*, pages 13–16, Saarbrücken, June 2003. CIRP.

[WKNL07] Matthias Wieland, Oliver Kopp, Daniela Nicklas, and Frank Leymann. Towards Context-Aware Workflows. In Barbara Pernici and Jon Atle Gulla, editors, *CAiSE'07 Proceedings of the Workshops and Doctoral Consortium Vol.2, Trondheim, Norway, June 11-15th, 2007*, pages 577–591. Tapir Acasemic Press, June 2007.

[WLJ+06] Matthias Wieland, Frank Leymann, Lamine Jendoubi, Daniela Nicklas, and Frank Dürr. Task-orientierte Anwendungen in einer Smart Factory. In Thomas Kirste, Birgitta König-Ries, Key Pousttchi, and Klaus Turowski, editors, *Mobile Informationssysteme - Potentiale, Hindernisse, Einsatz. Proceedings MMS'06*, volume P-76 of *Lecture Notes in Informatics (LNI)*, pages 139–143, Bonn, February 2006. Gesellschaft für Informatik.

[WLS+10] Matthias Wieland, Frank Leymann, Michael Schäfer, Dominik Lucke, Carmen Constantinescu, and Engelbert Westkämper. Using Context-aware Workflows for Failure Management in a Smart Factory. In *Proceedings of the Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies: UBICOMM 2010*, pages 379–384, Florence, Italy, October 2010. IARIA.

[WNL11] Matthias Wieland, Daniela Nicklas, and Frank Leymann. Benefits of Business Process Context for Human Task Management. *International Journal of Trade, Economics and Finance*, 2(4):304–311, August 2011.

Konzepte und Implementierung zur Verbesserung der Privatsphäre bei ortsbezogenen mobilen Diensten

Dennis Ludewig, Carsten Kleiner

Hochschule Hannover
Fakultät IV, Abteilung Informatik
Ricklinger Stadtweg 120
30459 Hannover
dennis.ludewig@stud.fh-hannover.de
ckleiner@acm.org

Abstract:

Standortbezogene Dienste werden heutzutage immer häufiger mittels Smartphone benutzt. Jedoch kann durch Bekanntgabe von Standortpositionen unter Umständen der Benutzer ausspioniert und die Privatsphäre verletzt werden. Gerade Continuous Queries können die Privatsphäre verletzen, da diese bei Continuous Queries schwieriger zu schützen ist als bei einmaligen Anfragen.

Als konkreter Anwendungsfall wird im Forschungsprojekt *Datenschutz- und Sicherheitsaspekte mobiler Dienste (DaSimoD)* an der Hochschule Hannover ein Modell namens *Pay-as-you-Drive (PAYD)* entwickelt. Dabei zahlen Versicherungsnehmer ihre Prämie flexibel nach dem jeweiligen Fahrumfang und -verhalten. Die zur Auswertung benötigten Positionsdaten werden dabei mit einem mobilen Endgerät gesammelt und an den Dienstanbieter übertragen. Da aus diesen Positionsdaten, die ein umfangreiches Bewegungsprofil darstellen, weitere persönliche Verhaltensweisen abgeleitet werden können, ist es erforderlich, diese Daten vor unberechtigtem Zugriff zu schützen. Um die Privatsphäre zu wahren, wird eine PAYD-Architektur entwickelt und prototypisch implementiert, die eine *Trusted-Third-Party (TTP)* nutzt. Diese TTP-Architektur ist Ausgangsbasis der weiteren Analyse. Dazu wird die Architektur auf Angriffsmöglichkeiten untersucht, besonders auf das *Direct und Indirect Location Privacy Problem*. Ziel ist es Verbesserungsmöglichkeiten zu finden und diese detailliert aufzuzeigen. Dabei werden verschiedene Angriffe auf die Privatsphäre vorgestellt und bewertet. Die vielversprechendsten Verbesserungsmöglichkeiten sind für die TTP-Architektur implementiert und getestet worden. Dieses Paper beschreibt die Konzepte und Implementierungen der umgesetzten Verbesserungsmöglichkeiten.

1 Einleitung

Standortbezogene Dienste (engl. *Location Based Services*, kurz: *LBS*) stellen dem Nutzer zusätzliche Informationen anhand der aktuellen Position bereit. Dieses hat für den Nutzer einen Mehrwert, birgt jedoch auch Gefahren, da unter Umständen die Privatsphäre des Nutzers verletzt wird. Bei einmaligen Anfragen kann die Position einfach verschleiert werden. Dieses ist jedoch bei kontinuierlichen Anfragen, sogenannten Continuous Queries, nicht so einfach möglich. Da die Anfragen in Bezug zueinander stehen, ist es wesentlich schwieriger seine Position glaubhaft zu schützen. Zudem können spezielle Anforderungen in Betracht kommen, sodass beispielsweise eine zeitliche oder räumliche Reduzierung der Genauigkeit nicht möglich ist.

Als konkreter Anwendungsfall wird ein Modell namens *Pay-as-you-Drive (PAYD)* untersucht. Dieses Modell kann bei KFZ-Versicherungen oder bei Versicherungsdienstleistern zum Einsatz kommen. Die Idee bei PAYD ist, dass die Versicherungsnehmer ihre Versicherungsprämie flexibel nach ihrem Fahrumfang oder -verhalten zahlen. Die dazu erhobenen Daten sollen nur für den Zweck der Prämienberechnung eingesetzt werden und es darf keine Möglichkeit bestehen weitere personenbezogenen Daten abzuleiten. Dazu werden im Forschungsprojekt *Datenschutz- und Sicherheitsaspekte mobiler Dienste (DaSimoD)* an der Hochschule Hannover zwei PAYD-Architekturen implementiert und untersucht [BKLZ11]. Die erste Architektur aggregiert zum Schutz der Privatsphäre die Positionsdaten lokal auf dem Endgerät. Die Metainformationen, wie Straßentyp oder Höchstgeschwindigkeit, werden dabei online abgefragt. Am Ende erhält die Versicherung vom Endgerät nur aggregierte Daten, den Report, zugeschickt. Dieser Report enthält keine Positionsdaten, sondern nur zusammengefasste Daten

wie gefahrene Kilometer oder zu welchen Zeitbereichen der Nutzer unterwegs war. Kritikpunkte an der Architektur sind, dass große technologische Anforderungen an das Endgerät gestellt werden und der Client sehr komplex ist. Zudem ist die Bereitstellung eines anonymen Geo-Datenservers durch Dritte sicherlich nicht zu erwarten. Aus Versicherungssicht ist außerden der Punkt der Sicherheit problematisch, da alles auf einem unkontrollierbaren Endgerät stattfindet. Diese Architektur ist jedoch nicht Gegenstand dieses Papers, da diese schon ausführlich in [BKZ10] und [BKLZ11] beschrieben wurde. Die zweite Architektur nutzt eine *Trusted-Third-Party (TTP)* um die Privatsphäre zu waren [BKLZ11]. Dabei werden die Positionsdaten zur Auswertung an einen vertrauenswürdigen Dritten geschickt. Sowohl der Nutzer, als auch die Versicherung müssen der TTP vertrauen, dass diese die Privatsphäre wahrt und auch die Daten korrekt aggregiert. Jedoch sollen auch weitere technische Schutzmöglichkeiten für die TTP-Architektur untersucht und implementiert werden, um die Privatsphäre des Nutzers besser zu schützen. Diese weiteren Schutzmöglichkeiten werden in diesem Paper verglichen und die Implementierung wird beschrieben. Die meisten Menschen würden PAYD nur nutzen, wenn ihre Privatsphäre ausreichend geschützt ist. Die Akzeptanz hängt deshalb besonders von der Sicherheit und dem Schutz der persönlichen Daten ab.

2 Related Work

Das Pay-as-you-Drive-Modell wird weltweit bereits in vielen Ländern eingesetzt und von unterschiedlichen Versicherungsnehmern genutzt [GB11]. Die meisten PAYD-Implementierungen erheben jedoch nur wenig Daten und/oder haben keine ausreichenden Maßnahmen zum Schutz der Privatsphäre [TDK+11]. Für die Versicherung wäre es wünschenswert in Zukunft genauere Daten zu erheben, um die Versicherungsprämie genauer und noch gerechter zu berechnen. Für den Versicherungsnehmer ist die eigene Anonymität wichtig. Daher wäre es wünschenswert, dass eine PAYD-Architektur sowohl die Privatsphäre der Versicherungsnehmer schützt, als auch der Versicherung ausreichende Kennzahlen zur Berechnung der Versicherungsprämie überträgt.

In [BKZ10] und [BKLZ11] wird die PAYD-Architektur des DaSimoD-Projekts beschrieben, die die Daten lokal aggregiert. Zudem wird die TTP-Architektur in [BKLZ11] genau beschrieben. Dazu zählen auch die Vor- und Nachteile sowie Implementierungsdetails. Die in diesem Paper beschriebene Verbesserungen basieren auf der Architektur, die im DaSimoD-Projekt implementiert wird.

In vielen Veröffentlichungen werden einzelne Schutzmaßnahmen für LBS detailliert dargestellt und beschrieben. Auf diese Veröffentlichungen soll aufgrund des Umfanges hier nicht weiter eingegangen werden. Stattdessen werden die relevanten Paper direkt bei den jeweiligen Schutzmaßnahmen referenziert.

3 TTP-Architektur

Die TTP-Architektur, die Abbildung 1 skizziert, übermittelt die Positionen zur Auswertung an die TTP. Dabei werden die Positionsdaten zunächst lokal auf dem Endgerät gespeichert und können dann in regelmäßigen Abständen an die vertrauenswürdige Stelle übermittelt werden. Diese aggregiert die Daten und hält die aggregierten Daten dann für die Versicherung zur Abholung bereit. Die TTP wird von der Versicherung ebenso als vertrauenswürdig angesehen und übernimmt daher die Aggregation. Dazu müssen auf dem TTP-Server Kartendaten vorhanden sein bzw. es wird ein Server angefragt der ebenfalls im identischen Vertrauensbereich liegt.

Da der TTP das komplette Bewegungsprofil übermittelt wird, werden Pseudonyme eingesetzt. So soll die wahre Identität des Nutzers der TTP verborgen bleiben. Dabei bekommt das Endgerät als erstes ein Pseudonym von der Versicherung zugeteilt. Dieses kann vom Endgerät zu jeder Zeit erneuert werden. Das Endgerät überträgt mit diesem Pseudonym die Positionsdaten an die vertrauenswürdige Stelle. Nach erfolgreicher Aggregation werden die Daten bis zur Abholung durch die Versicherung vorgehalten. Da die Versicherung die ausgestellten Pseudonyme kennt und den einzelnen Versicherungsnehmern zuordnen kann, können so die aggregierten Daten zugeordnet werden. Dieses hat den entscheidenden Vorteil, dass die TTP die wahre Identität des Benutzers nicht kennt. Zudem muss ein gewisser Aufwand betrieben werden, um aus verschiedenen Pseudonymen ein komplettes Bewegungsprofil zu erstellen. Der größte Nachteil ist, dass der TTP die kompletten Positionsdaten, wenn auch unter einem Pseudonym, übermittelt werden müssen. Für weitere Details zu der Architektur sei auf [BKLZ11] verwiesen.

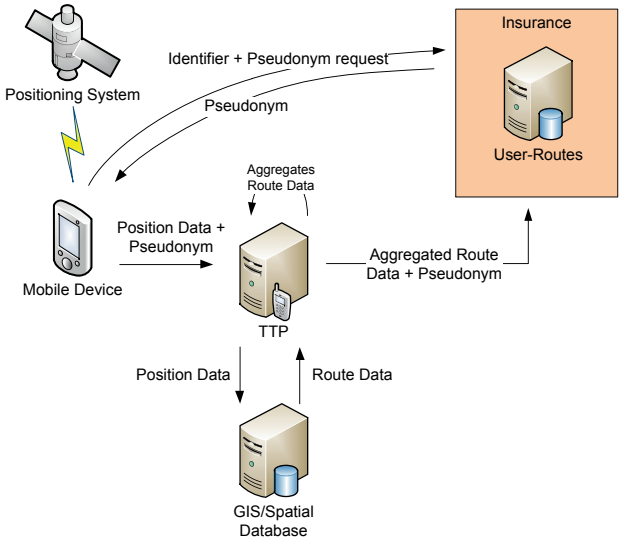


Abbildung 1: Auswertung durch vertrauenswürdige Stelle

4 Angriffsmöglichkeiten

Das Hauptproblem bei der Speicherung ortsbezogener Daten ist das Ableiten weiterer Informationen aus den vorhandenen Daten. Abbildung 2 listet dabei die möglichen Angriffe auf ortsbezogene Daten auf. Die Angriffe können in zwei Arten von *Location Privacy Problem* unterteilt werden: das *Direct Location Privacy Problem* und das *Indirect Location Privacy Problem* [Dec08].

Das *Direct Location Privacy Problem* beschreibt die direkte Ableitung weiterer Informationen aus der Position des Nutzers. Dieses können persönliche Interessen, der Arbeitgeber oder der eigene Freundeskreis sein. So kann die Angabe, dass sich eine Person an einer bestimmten Position aufgehalten hat, die Privatsphäre beeinträchtigen. Beispielsweise kann der Arbeitgeber oder die Parteizugehörigkeit offengelegt werden, wenn die Position zu einem Betriebsgebäude oder zu einer Parteizentrale gehört. Ebenso ist es denkbar, dass auf Hobbies gefolgert werden kann, wenn die Position ein Fitnessstudio, Vereinsheim, Schwimmbad oder ähnliches widerspiegelt. Auch denkbar, aber deutlich kritischer, ist der Gesundheitszustand, auf den gefolgert werden kann, wenn die Position ein Krankenhaus oder (Fach)arzt ist. Dieses ist keine vollständige Liste, sondern zeigt nur verschiedene Probleme, die aus den gespeicherten Positionsdaten hervorgehen können.

Durch Hinzunahme der Zeit können detailliertere Informationen erfasst werden. Somit erhöht sich das Risiko, dass die Privatsphäre verletzt wird. So kann nicht nur auf den Sportverein gefolgert werden, sondern mithilfe der Uhrzeit auf bestimmte Treffen oder bestimmte Sportarten. Weiter ist es möglich, dass die Position nur zu bestimmten Zeiten Rückschlsse auf die Privatsphäre zulässt. So ist der Besuch eines öffentlichen Gebäudes aus Sicht des Datenschutzes nicht kritisch zu beurteilen. Durch die Hinzunahme der Zeit könnte jedoch gefolgert werden, dass die Person kein Besucher des Gebäudes ist, wenn das Gebäude außerhalb der Öffnungszeiten besucht wird. Dieses Direct Location Privacy Problem bezieht sich dabei immer auf die direkte Identität des Nutzers, d.h. für Nutzer ohne Pseudonym.

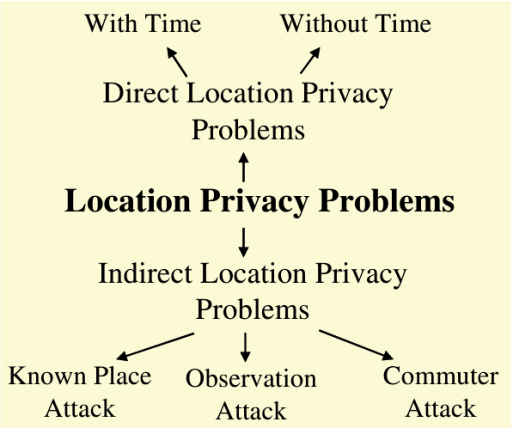


Abbildung 2: Klassifizierung der Angriffe auf LBS [Dec08]

Viele Dienste sind unter einem Pseudonym nutzbar, da die Dienste die wahre Identität des Nutzers nicht kennen müssen. Diese Dienste haben meist trotzdem noch das *Indirect Location Privacy Problem*. Solche Angriffe stellen einen Angriff auf das Pseudonym dar. In dem Fall werden ortsbezogene Daten mit anderen externen Daten kombiniert, um so die Identität hinter dem Pseudonym offenzulegen. So können mithilfe öffentlicher Informationen, wie Telefonbuch oder Landkarte, Rückschlüsse auf die wahre Identität gefolgert werden, wenn die Position beispielsweise das Grundstück des Nutzers widerspiegelt. So etwas wird *Known Place Attack* genannt.

Kontinuierliche Positionsangaben, z.B. von einem Nutzer der zur Arbeit fährt, können ebenfalls eine Möglichkeit für einen Angriff darstellen. Wenn der Nutzer hinter dem Pseudonym aufgrund seines regelmäßigen Weges identifiziert wird, wird das *Commuter Attack* genannt.

Die letzte Möglichkeit ist die *Observation Attack*. Wenn der Angreifer einen bestimmten Bereich beobachtet, dieses kann persönlich oder mithilfe einer Überwachungskamera geschehen, kann das Pseudonym offengelegt werden. Falls sich mehrere Benutzer in dem Bereich aufhalten und den Dienst relativ zeitgleich nutzen, kann noch nicht auf einen einzelnen Nutzer geschlossen werden. Jedoch kann der Benutzerkreis eingeschränkt und bei weiteren Anfragen verfeinert werden. Daher ist es ratsam, das Pseudonym regelmäßig zu wechseln.

5 Angriffsmöglichkeiten und Schutzmöglichkeiten für TTP

In diesem Kapitel werden mögliche Angriffe auf die TTP-Architektur aufgezeigt. Für jeden Angriff werden eine oder mehrere Lösungsmöglichkeiten vorgestellt und bewertet. Dabei ist besonders auf die Beeinträchtigung der Lösungsmöglichkeiten untereinander zu achten. Ziel der Angriffe ist es, die Identität des Versicherungsnehmers offenzulegen und somit unter Umständen die Privatsphäre zu verletzen, da ein Bewegungsprofil vorliegt. Auf Basis dieser Angriffe werden Verbesserungsmöglichkeiten vorgestellt und bewertet. Diese sollen den Angriff einschränken, wenn nicht sogar ganz verhindern.

Die beschriebenen Angriffe sind nach *Location Privacy Problems* wie in Abbildung 2 aufgeteilt. Da in der Architektur Pseudonyme zum Einsatz kommen, können keine *Direct Location Privacy Problems* auftreten. Dieses gilt allerdings nur, wenn nicht mehrere Server kompromittiert werden. So kann das *Direct Location Privacy Problem* auftreten, wenn neben der TTP auch die Versicherung kompromittiert wird. In diesem Fall können die Positionsdaten mit Pseudonymen mithilfe der Versicherung direkt Identitäten zugeordnet werden. Dieses Problem lässt sich

in dieser Architektur nur minimieren bzw. verhindern, wenn die TTP die Positionsdaten schnell aggregiert. Somit sind keine detaillierten Information mehr verfügbar. Die folgende Einteilung findet deshalb in den einzelnen Kategorien von *Indirect Location Privacy Problems* statt. Dabei werden zuerst die Probleme beschrieben und danach eine oder mehrere Lösungsmöglichkeiten bzw. Verbesserungen vorgestellt.

Besondere Schwierigkeiten ergeben sich bei der Verbesserung der Privatsphäre für Pay-as-you-Drive, da aufgrund von Anforderungen nicht alle potentiellen Möglichkeiten infrage kommen. So haben PAYD-Anwendungen die folgenden Anforderungen:

- hohe Genauigkeit der Positionsangaben notwendig
- regelmäßige Positionsbestimmung notwendig, um die Fahrstrecke genau zu erfassen
- anonyme Nutzung von PAYD nicht möglich, da Versicherungsprämie bestimmt werden muss

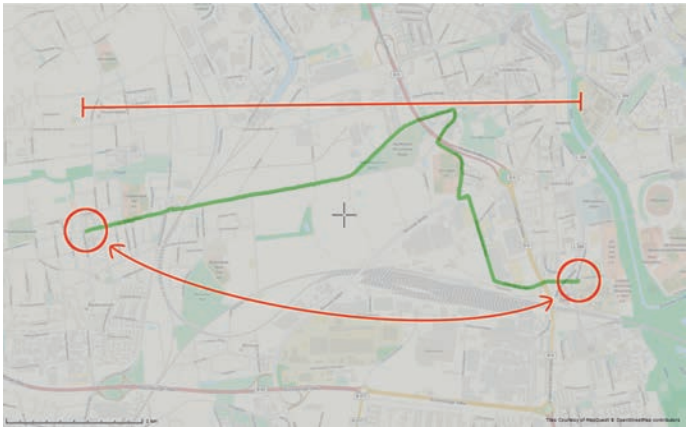


Abbildung 3: Angriffsmöglichkeiten auf PAYD

Abbildung 3 skizziert die potenziellen Angriffsmöglichkeiten auf das PAYD-Szenario. Dabei symbolisieren die beiden roten Kreise einen Angriff durch die Known Place Attack. Anhand von weiteren externen Informationen kann eventuell auf die wahre Identität des Nutzers geschlossen werden. Die geschwungene rote Linie stellt den Angriff durch die Commuter Attack dar. Dieses ist, wie der Name vermuten lässt, typischerweise der tägliche Weg zur Arbeitsstelle. Dabei wird durch die Verbindung *Zuhause - Arbeit* eine Identifizierung ermöglicht. Der letzte Angriff ist die Observation Attack. Dabei wird der Nutzer manuell oder automatisch beobachtet und so eine Identifizierung ermöglicht. Dieses wird von der geraden roten Linie symbolisiert.

Für jede Verbesserung wird auf ein oder mehrere Paper referenziert, in denen die Verbesserung eingeführt wurde. Aus Platzgründen wird hier für jede Verbesserung nur ein kurzer Überblick zur Idee der Verbesserung gegeben. Die Verbesserungen werden in dem jeweiligen Paper im Detail erklärt.

5.1 Known Place Attack

Die *Known Place Attack* beschreibt einen Angriff auf die Identität, wenn die Position bekannt ist. Das heißt, anhand einer Positionsangabe kann auf eine Identität geschlossen werden. Dabei können öffentliche Informationen zur

Hilfe gezogen werden, wie eine Landkarte oder ein Telefonbuch. Obwohl die Versicherung von der TTP getrennt ist und die TTP nur über ein Pseudonym verfügt, besteht so eine gewisse Chance, die wahre Identität zu ermitteln.

Die TTP empfängt unter einem Pseudonym mehrere Positionsangaben. Gerade der Anfang oder das Ende einer Fahrt kann die Person identifizieren [Kru07]. Dazu kann die Adresse anhand der Positionsdaten mit Hilfe einer Landkarte recherchiert werden. Dieses kann nun mit etwas Glück durch die Rückwärtssuche im Telefonbuch in einen realen Namen aufgelöst werden. Doch auch ohne Realnamen können nun mehrere Pseudonyme einander zugeordnet werden, wenn auf die gleiche Adresse gefolgert werden kann. Das heißt, dass mehrere Pseudonyme durch die Positionsangabe verknüpft werden. Dieses geht allerdings nur, wenn die Position einer Identität eindeutig zugeordnet werden kann. Durch weitere Positionen kann die Menge der Identitäten eingeschränkt werden, bis die wahre Identität herausgefunden wurde.

So kann neben dem Angriff auf den Wohnsitz einer Identität ebenso die Arbeitsstelle als Risiko der Privatsphäre dienen. Dabei können Daten von der privaten Homepage oder von sozialen Netzwerken wie Facebook oder Xing benutzt werden. Um ein weiteres Beispiel zu nennen, kann dieses ebenso für eine Vereinsmitgliedschaft geschehen. Dabei könnte eine Mitgliederliste, Bestzeitenlisten vom Sport, Auszeichnungen oder Ansprechpartner für den Verein durchsucht werden.

5.1.1 k-Anonymity

Durch *k-Anonymity* ([Swe02], [Dec08]) wird die Genauigkeit der Positionsangaben reduziert. Dieses kann sowohl räumlich als auch zeitlich geschehen. Ziel ist es, die Genauigkeit der Positionsangaben so weit zu reduzieren, dass $k-1$ weitere Nutzer in Betracht kommen, der potentielle Nutzer zu der Anfrage zu sein. Insgesamt kommen so k Nutzer in Frage.

Um k-Anonymity einzusetzen, muss ein weiterer Server eingesetzt werden, der die Genauigkeit überwacht. Dieser Server wird *Mediator* genannt. Diesem Mediator teilen die Nutzer ihre Positionsangaben mit, damit dieser die Genauigkeit reduziert. Ebenso kann die gewünschte Anonymität in Form von k übertragen werden. Jeder Nutzer kann somit sein eigenes Level der Anonymität definieren. Erst wenn dieses Level erreicht ist, sendet der Mediator die Daten an die TTP weiter. Dem Mediator muss somit, auch wie der TTP, vertraut werden. Wenn beispielsweise der Mediator die k-Anonymity falsch berechnet, wäre die Anonymität der Nutzer nicht mehr gewährleistet. Ebenso darf der Mediator die Daten nicht so verfälschen, dass der Nutzer eine höhere Prämie zahlen muss.

5.1.2 Dummy-Requests

Neben den korrekten Positionsdaten könnte das mobile Endgerät zusätzlich Dummy-Requests ([Dec08]) an die TTP senden. Diese Dummy-Requests würden auch von der TTP aggregiert und an die Versicherung übermittelt. Dabei ist zu beachten, dass die korrekten Positionsdaten und Dummy-Requests jeweils unter einem anderen Pseudonym eingereicht werden. Das Endgerät sendet nun der Versicherung die Pseudonyme der tatsächlich gefahrenen Strecken. Somit kann die TTP nicht bestimmen, welche Strecken der Nutzer tatsächlich gefahren ist, wenn trotz Pseudonym die Identität offengelegt werden kann.

Das Hauptproblem ist die Erzeugung der Dummy-Requests. Da es sich bei PAYD um *Continuous Queries* handelt, ist es besonders schwierig einen korrekten Bezug der Anfragen untereinander herzustellen. Eine Berechnung der möglichen Routen auf dem Endgerät fällt aus, da das mobile Endgerät über keinerlei Karten verfügt. Zudem könnte, wenn Kartendaten oder ein Kartenausschnitt auf dem Endgerät vorhanden wäre, die Berechnung der eigenen Strecke selbst vorgenommen werden.

Das mobile Endgerät könnte Fahrstrecken zur Verwendung der Dummy-Requests von der Versicherung oder durch direkte Verbindung zu anderen mobilen Endgeräten bekommen. So könnte die TTP die Positionsdaten anonymisieren und an die Versicherung übertragen. Diese teilt dem mobilen Endgerät die Streckendaten mit, damit diese als Dummy-Requests verwendet werden können. Wesentliche Nachteile dieser Verbesserung sind, dass die Streckendaten sowohl regelmäßig aktualisiert werden und zudem viele Streckendaten verfügbar sein müssen. Andernfalls können diese Dummy-Requests schnell aussortiert und damit auf die korrekten Positionsdaten geschlossen werden.

5.1.3 Ban-zone

Eine Abwehrmöglichkeit für den Angriff ist, dass für die kritischen Bereiche keine Positionsangaben an die TTP übermittelt werden. Ein kritischer Bereich ist dabei jede Position, bei dem auf die wahre Identität des Nutzers geschlossen werden kann. Dieses ist etwa der eigene Wohnsitz, die Arbeitsstelle oder Mitgliedschaften bei einer Partei, einem Fitnessstudio oder in einem Sportverein. Dieses kann aber auch andere Institutionen treffen, über die der Nutzer identifiziert werden kann.

Da diese Ban-zone ([Dec08]) nur einen kleinen Bereich der Fahrstrecke ausmacht, kann auf die detaillierte Auswertung verzichtet werden. Zudem könnte das mobile Endgerät die in der Ban-zone gefahrenen Daten selbst aggregieren und an die TTP übermitteln. So wäre es denkbar, dass das Endgerät für die Ban-zone die gefahrenen Kilometer, Uhrzeit und Durchschnittsgeschwindigkeit selbstständig berechnet und überträgt. Dieses würde die Berechnung der Versicherungsprämie nicht verfälschen, da im Gegensatz zur TTP nur die weiteren Metadaten wie Straßentyp und daraus resultierende Geschwindigkeitsüberschreitung verloren gehen. Alternativ kann dafür die Architektur *Auswertung auf dem Endgerät* [BKZ10] verwendet werden. Dieses stellt eine Kombination der beiden Architekturen dar. Es wäre also ein hybrider Client. Nachteilig wäre hierbei, dass der Client durch die Architektur wieder komplexer wird.

Sinnvoll ist die Verwaltung der Ban-zone auf dem mobilen Endgerät. Somit erfahren weder die Versicherung noch die TTP von den schützenswerten Positionen. Der Nutzer muss also seine eigenen Ban-zones selber definieren. Damit keine Ban-zone vergessen wird, könnte der Nutzer dafür von der Versicherung eine Checkliste mit Fragen und Beispielen bekommen, die die persönliche Auswahl der Ban-zones erleichtern. Listing 1 beschreibt eine beispielhafte Checkliste zum Festlegen der Ban-zones.

Ist eine Ban-zone definiert für...	**OK?**
Aktuellen Wohnort	
Parkplatz des Autos für Wohnort	
Aktuelle Arbeitsstelle	
Parkplatz des Autos für Arbeitsstelle	
Alle Aufenthaltsorte vom eigenen Verein/Fitnessstudio	
Regelmäßige wöchentliche Treffen	
Regelmäßige Arzt-/Krankenhausbesuche	
Sonstige regelmäßige Treffen	

Tabelle 1: Checkliste für Ban-zones

Der Nutzer könnte nun selbsständig mithilfe einer Karte seine Ban-zones definieren. Das größte Risiko ist die Wahl der richtigen Ban-zone. So könnte ein Angreifer einfach den Mittelpunkt des Kreises bestimmen. Dieses ist noch einfacher, wenn ein fixer Radius bei jedem Versicherungsnehmer verwendet wird. Ein Ratschlag an den Versicherungsnehmer wäre nun, den Mittelpunkt beliebig zu setzen, nur so, dass sein Grundstück/Auto noch im Radius liegt. Dieses birgt jedoch das Risiko, dass der Mittelpunkt falsch gesetzt wird bzw. viele Nutzer den Mittelpunkt ähnlich setzen. Daher ist diese Methode nicht zu empfehlen.

In [Kru07] wird eine bessere Variante zum Setzen der Ban-zone vorgeschlagen. Diese wird von Abbildung 4 illustriert. Der Nutzer wählt genau den Punkt aus, aufgrund dessen die Ban-zone errichtet werden soll. Um diesen Punkt wird ein Kreis mit Radius r gezogen. In diesem Kreis wird ein zufälliger Punkt gewählt und ein weiterer Kreis mit Radius R gezogen, wobei $r < R$ gültig sein muss. Dieses stellt insgesamt sicher, dass ein Angreifer den schützenswerten Ursprungsort nur sehr schwer erraten kann.

Das Endgerät muss nun überprüfen, ob eine Position innerhalb einer Ban-zone liegt. Ist dieses der Fall, werden die Positionen in der Ban-zone vom Endgerät selbst aggregiert. Andernfalls werden die Positionen bis zur Übermittlung der Versicherung auf dem Endgerät gespeichert. Zusätzlich ist durch das Endgerät zu beachten, dass keine Ban-zone innerhalb einer Fahrstrecke vorkommt. Ist dieses der Fall, muss die Strecke logisch aufgeteilt werden, damit beide Teile unter einem anderen Pseudonym eingereicht werden. Andernfalls würde die Privatsphäre verletzt werden. Dieses ist der Fall, da durch die plötzliche Nichtübertragung die Ban-zone verraten wird. Da durch die Positionsangaben vor und nach der Ban-zone Uhrzeiten übermittelt werden, kann zudem auf die Verweildauer in der Ban-zone gefolgert werden.

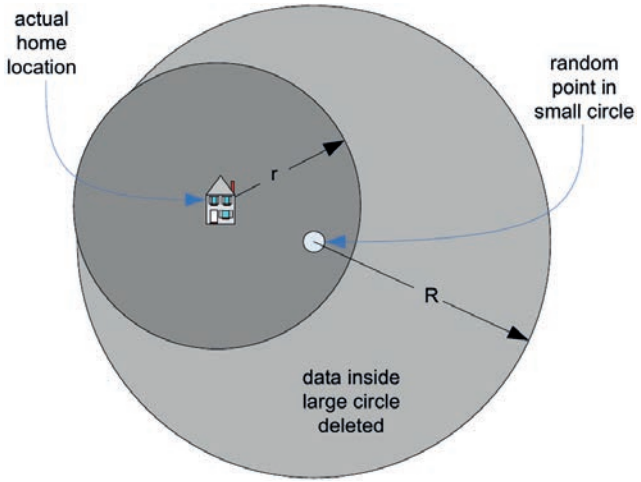


Abbildung 4: Skizze zur Berechnung der Ban-zone [Kru07]

5.1.4 Privater Fahrmodus

Ban-zones werden für Bereiche festgelegt, die der Nutzer regelmäßig besucht und anhand derer eine Identifizierung möglich ist. Jedoch gibt es noch andere Fälle, wo die Festlegung einer Ban-zone mangels Regelmäßigkeit keinen Sinn ergibt. Diese sollen von einem *Privaten Fahrmodus* abgedeckt werden. In diesem privaten Modus sollen wie bei der Ban-zone keine Positionsdaten an die Versicherung übermittelt werden. Eine Ban-zone zu definieren, zum Beispiel für eine jährlich stattfindende Hauptversammlung oder für spontane Arztbesuche, ist unter Umständen aufwändiger, als in den privaten Fahrmodus zu wechseln.

Wie auch bei der Ban-zone übernimmt das Endgerät die Auswertung der Positionsangaben im privaten Fahrmodus. Dabei werden neben der Anzahl der Kilometer auch die Durchschnittsgeschwindigkeit und Dauer an die TTP übermittelt. Der private Fahrmodus ist nur als Ergänzung zur Ban-zone anzusehen, da ein dauerhaftes Fahren im privaten Modus für beide Seiten nicht sinnvoll ist. Der Versicherung werden zu wenig Kennzahlen übertragen und dadurch wird der Nutzer weniger Versicherungsprämie einsparen.

Alternativ könnte für den privaten Modus auch die Architektur *Auswertung auf dem Endgerät* eingesetzt werden. Hierbei übernimmt für private Strecken das Endgerät die Auswertung und die restlichen Strecken werden von der TTP ausgewertet. Der Client wäre damit ein hybrider Client aus Architektur *Auswertung auf dem Endgerät* und *Auswertung durch vertrauenswürdige Stelle*. Für die Versicherung hat dieses den entscheidenden Vorteil, dass auch für private Strecken detaillierte Informationen zur Aggregation herangezogen werden.

5.2 Commuter Attack

Die *Commuter Attack* beschreibt den Angriff auf die Privatsphäre durch das Analysieren einer Abfolge von Standortpositionen. Dieses ist typischerweise der Arbeitsweg, von dem sich auch der Name des Angriffs ableitet. Es können aber auch andere Strecken Verwendung finden, wie z.B. der Weg zur Parteizentrale, Fitnessstudio oder Sportverein.

Im Gegenzug zur *Known Place Attack*, wo nur eine einzelne Position zur Identifizierung genügt, wertet die Com-

muter Attack eine Fahrstrecke aus. So könnte beispielsweise bekannt sein, welche potentiellen Nutzer an Position A infrage kommen und ebenso, welche Nutzer an Position B. Jedoch kann ein Nutzer nur identifiziert werden, wenn durch die Wegstrecke ersichtlich ist, welcher Nutzer von A nach B fährt.

5.2.1 Mix-zone

Um die Commuter Attack zu unterbinden, ist es erforderlich, dass der TTP keine durchgängigen Positionsangaben mehr gesendet werden. Das heißt, dass das Endgerät das Pseudonym wechselt, damit keine zusammenhängenden Strecken unter einem Pseudonym entstehen können. Der Wechsel des Pseudonyms wird durch Mix-zones ([Dec08], [BS03]) geregelt.

Die Idee der Mix-zones kommt von *Mixe*, die die Anonymität im Internet sichergestellt haben. Dazu haben Mixe Daten von verschiedenen Clients zwischengespeichert und, wenn eine gewisse Anonymitätsmenge vorhanden ist, diese Daten in zufälliger Reihenfolge weitergeleitet. Dadurch wurde die Anonymität der Clients gegenüber den Servern sichergestellt, da nur beim Mix bekannt ist, von welchem Client die Anfrage kommt.

Auf Mix-zones in LBS übertragen bedeutet das, dass eine Mix-zone ein Bereich ist, in dem die Nutzer ihre Identität wechseln können. Die TTP nutzt zum Identifizieren der Nutzer Pseudonyme. Es wird also das alte Pseudonym durch ein neues Pseudonym ausgetauscht. Da ein Austausch des Pseudonyms an einem beliebigen Ort nicht sinnvoll ist, da so die TTP das neue Pseudonym wieder mit dem alten Pseudonym verknüpfen kann, wurden Mix-zones eingeführt. In dieser Mix-zone werden von den Nutzern keine Positionsinformationen übertragen und ermöglichen daher einen sicheren Wechsel des Pseudonyms.

So eine Mix-zone kann auf den Wegstrecken mehrmals vorkommen und so die Chance erhöhen, dass ein Pseudonymwechsel nicht nachvollziehbar ist. Dieses kann noch gesteigert werden, wenn mehr potentielle als tatsächliche Nutzer in dieses Verfahren einbezogen werden. Dieses könnte mit *Temporal Cloaking* erreicht werden. Hierauf soll hier nicht weiter eingegangen werden, sondern es wird auf Kapitel 5.3.1 verwiesen. Es sei nur erwähnt, dass die Mix-zone mit Temporal Cloaking eine sehr gute Kombination darstellt.

Die Mix-zone sollte durch den Server implementiert werden. Dabei könnten beispielsweise zunächst alle Kreuzungen mit Ampeln herausgefunden werden. Diese Positionen können in der Datenbank gespeichert werden. Um diese Position herum wird eine Mix-zone mit einer bestimmten Größe eingerichtet und den Clients mitgeteilt.

5.2.2 Provider Change

Anstatt durch Mix-zones zu verhindern, dass alte und neue Pseudonyme nicht miteinander verknüpft werden können, kann auch ein *Provider Change* ([Dec08]) stattfinden. Ein Provider Change besagt, dass die Positionsdaten nun zu einem neuen Provider, also zu einer neuen TTP, geschickt werden. Eine Verknüpfung der Pseudonyme ist jetzt nicht mehr möglich, da ein Austausch unter den einzelnen TTPs nicht erlaubt ist. Zudem können keine Angriffe auf die Wegstrecke stattfinden, da ein Provider nur einen Teil der Wegstrecke vom Nutzer bekommen hat. Folglich erhöht diese Methode die Sicherheit in Bezug auf Servereinbruch. So können bei einem Servereinbruch oder wenn die TTP von einem *Evil Admin* administriert wird, nicht die gesamten Positionsdaten abgegriffen werden, sondern nur der Teil, der auf dem Server gespeichert ist.

Nach Aggregation der Positionsdaten schicken die TTP die aggregierten Daten an die Versicherung bzw. die Versicherung ruft die aggregierten Daten von allen TTPs ab. Die Versicherung muss nun die Daten von den verschiedenen TTPs zusammenführen. Bei der bestehenden Architektur muss die Versicherung Daten zahlreicher Routen zusammenfassen, bekommt aber alle von einer TTP.

5.2.3 Path Confusion

Path Confusion ([HG05], [Dec08]) basiert auf dem gleichen Prinzip wie Mix-zones. Ziel ist es, neue Pseudonyme nicht den alten zuordnen zu können. Statt Mix-zones auf dem Endgerät zu implementieren, findet Path Confusion auf dem Server, dem Mediator, statt. Dabei wird der Umstand ausgenutzt, dass sich die Wege zweier oder mehrerer Nutzer kreuzen. Der Mediator blendet dabei nicht wie bei den Mix-zones die Positionen aus, sondern reduziert die

Genauigkeit. Dadurch kommen mehr potentielle Nutzer nach einem Pseudonymwechsel in Betracht. Für PAYD müssen für den Mediator ein weiterer Server zum Einsatz kommen. An diesen werden die Positionen geschickt und nach erfolgreicher Bearbeitung an den TTP-Server weitergeleitet.

5.3 Observation Attack

Durch Beobachten des Nutzers kann die Identität aufgedeckt werden. Dieses kann sowohl manuell durch direkte Beobachtung geschehen als auch automatisiert durch Anlagen, die den Verkehrsfluss analyisieren und überwachen. Ein Beispiel ist ein System wie *Toll Collect*. Der Nutzer wird während einer Anfrage an den LBS beobachtet. Hierdurch ist die Zuordnung des Pseudonyms zu seiner Identität möglich.

5.3.1 Reduction Precision: Temporal

Um die Beobachtung und Zuordnung zu erschweren, ist es sinnvoll, ein Spatial und/oder Temporal Cloaking ([Dec08]) durchzuführen. Im PAYD-Szenario ist eine hohe Genauigkeit wichtig, ein Spatial Cloaking ist daher nicht sinnvoll. Da die Anfragen aber zwischengespeichert und vom Endgerät verändert werden, kann so die Zeit zur Position verändert und somit Temporal Cloaking durchgeführt werden. Dieses vergrößert den potentiellen Nutzerkreis drastisch, wenn ein hohes Temporal Cloaking stattfindet. Im PAYD-Szenario können die Zeitangaben nicht nur sekündlich verändert werden, sondern ein ungefährer Zeitrahmen ist für die Versicherung ausreichend und sinnvoll. So ist die Versicherung interessiert, ob ein Nutzer zum Berufsverkehr oder zur Nebenzeit unterwegs ist. Auch ist es für die Versicherung interessant zu wissen, zu welcher Tageszeit die Fahrten stattgefunden haben. Dieses ist gerade bei jungen Fahrern entscheidend, z.B. bei Nachtfahrten.

Statt genaue Zeitangaben an die TTP zu senden, ist es somit möglich, die Zeiten auf- oder abzurunden. So wäre es denkbar, die Angaben jeweils auf die volle oder halbe Stunde zu runden. Beispielsweise könnte die erste Positionsangabe unter einem Pseudonym von der genauen Zeit 08:16:23 auf 08:00 Uhr gerundet werden. Alle weiteren Positionen werden demnach zeitlich um 16 Minuten und 23 Sekunden reduziert. Anhand der Zeitangabe hat die Versicherung immer noch eine ausreichende Kennzahl zur Berechnung der Versicherungsprämie.

5.3.2 k-Anonymity

Auch zum Verhindern von Observation Attacken kann k-Anonymity, wie in Abschnitt 5.1.1 beschrieben, zum Einsatz kommen; k-Anonymity reduziert hier die Genauigkeit der Positionsangaben. Dadurch wird es einem Angreifer deutlich erschwert, durch Beobachtung auf die Identität zu schließen.

5.3.3 Dummy-Requests

Wie in Abschnitt 5.1.2 können diese auch als Schutz für Observation Attacks eingesetzt werden. Dabei ist, ebenfalls wie bei k-Anonymity, das Ziel, dass der Nutzer nicht direkt identifiziert werden kann. Dieser versteckt sich zwischen einer Menge von anderen Nutzern. Dieses hängt von der versendeten Anzahl der Dummy-Requests ab. Somit kann nicht eindeutig gesagt werden, ob der Nutzer wirklich an dem Ort war. Dieses erschwert die Observation Attack.

5.4 Auswahl einer sinnvollen Kombination von Verbesserungen

Tabelle 2 stellt die untersuchten Verbesserungsmöglichkeiten gegenüber. Diese gilt es zu vergleichen, um geeignete Kombinationsmöglichkeiten auszuwählen. Dabei ist besonders darauf zu achten, dass sich die Verbesserungen nicht gegenseitig blockieren oder gemeinsam zu einer Schwachstelle werden.

	Known Place Attack	Ban-zone	Privaten Fahrmodus	k-Anonymity	Dummy-Requests	**Commuter Attack**	Mix-zone	Provider Change	Path Confusion	**Observation Attack**	Reduction Precision: Temporal	k-Anonymity	Dummy-Requests
Known Place Attack													
Ban-zone		×	+	-	-		+	○	○		○	○	○
Privaten Fahrmodus		+	×	-	-		-	-	-		-	-	-
k-Anonymity		-	-	×	○		○	○	○		+	+	○
Dummy-Requests		-	-	○	×		○	○	○		○	○	+
Commuter Attack													
Mix-zone		+	+	○	○		×	○	○		+	○	+
Provider Change		○	-	○	○		○	×	○		○	○	○
Path Confusion		○	-	○	○		○	○	×		+	○	○
Observation Attack													
Reduction Precision: Temporal		○	-	+	○		+	○	+		×	○	○
k-Anonymity		○	-	+	○		+	○	+		○	×	○
Dummy-Requests		○	-	○	+		+	○	○		○	○	×

Tabelle 2: Kombinationsmöglichkeiten der Verbesserungen

Es wurden drei Angriffskategorien vorgestellt, für die mindestens eine Abwehrmaßnahme pro Kategorie ausgewählt werden soll. Nur so kann die Privatsphäre der Nutzer ausreichend geschützt werden. Die Ban-zone lässt sich gut mit den privaten Fahrmodus oder Mix-zones kombinieren, da bei allen dreien eine Überprüfung stattfinden muss, ob das Endgerät in einem bestimmten Bereich ist. Zudem kann die Fahrstrecke und Zeit in diesem Bereich aggregiert werden. Dieses kann von allen drei verwendet werden. Für die Known-Place Attack mit der Ban-zone machen k-Anonymity und Dummy-Requests keinen Sinn, da durch die Ban-zone keine Positionen verschickt werden sollen. Wenn der private Fahrmodus aktiv ist, macht eine Kombination mit keiner Verbesserung einen Sinn. Durch den privaten Fahrmodus werden keine Daten an die TTP gesendet. Daher ist keine weitere Verbesserung nötig. k-Anonymity lässt sich gut mit Temporal Reduction kombinieren, da k-Anonymity besonders profitiert, wenn die Zeit in einem größeren Zeitrahmen verändert werden kann. Mix-zones passen sehr gut zu Temporal Reduction, da sich dadurch die potentiellen Nutzer in einer Mix-zone erhöhen. Dieses würde die Chance für einen erfolgreichen Pseudonymwechsel erhöhen. Einen ähnlichen Effekt lässt sich durch Dummy-Requests erreichen, da mehr Requests von potentiellen Nutzern generiert werden. Ähnlich wie bei k-Anonymity profitiert auch Path Confusion durch Temporal Reduction. Der Algorithmus hat eine höhere Chance, dass sich die Wege der Nutzer kreuzen, wenn die Zeit in einem größeren Zeitrahmen verändert werden kann.

Da auf einen Mediator möglichst verzichtet werden soll, da diesem erneut vertraut werden muss, sollen Verbesserungsmöglichkeiten ohne Mediator gewählt werden. Dabei stellt die Mix-zone mit der Temporal Reduction eine sehr gute Kombination dar, da dieses die potentiellen Nutzer erhöht. Zudem passen Ban- und Mix-zones gut zusammen, da die Überprüfung und lokale Aggregation für beide Verfahren sehr ähnlich ist. Aus diesem Grund werden als Verbesserungen Ban- und Mixzones mit Temporal Reduction implementiert. Zusätzlich ist dieses auch für den privaten Fahrmodus der Fall. Dieser schützt besonders bei spontanen oder kritischen Fahrten.

6 Implementierung

In diesem Kapitel wird die Implementierung der Verbesserungen für die TTP-Architektur beschrieben. Abbildung 5 beschreibt die Kombination der einzelnen Konzepte in der Android-App. Dabei spiegelt die Abbildung das Vorgehen beim Aufzeichnen der Positionen wieder.

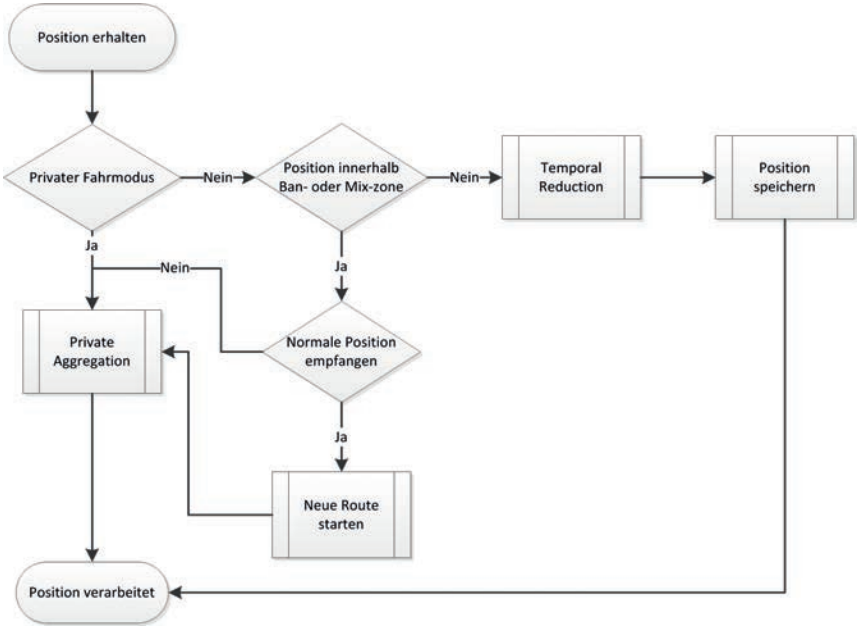


Abbildung 5: Kombination der Konzepte

Wenn die App eine Position empfängt, wird zunächst überprüft ob der private Fahrmodus aktiv ist. Ist dieses der Fall muss die Position nur lokal aggregiert und in der Datenbank gespeichert werden. Falls der private Fahrmodus nicht aktiv ist, wird überprüft, ob sich der Nutzer in einer Ban- oder Mix-zone befindet. Wenn dieses der Fall ist, wird ebenfalls, wie beim privaten Modus, die Position lokal aggregiert. Zusätzlich wird bei der Ban- und Mix-zone überprüft, ob schon eine normale Position empfangen wurde. Ist dieses der Fall muss eine neue Route angelegt werden, damit die neue Route unter einem neuen Pseudonym eingereicht werden kann. Andernfalls könnte die Privatsphäre verletzt werden. Wenn weder der private Fahrmodus aktiv ist, noch sich der Nutzer in einer Ban- oder Mix-zone befindet, findet die zeitliche Reduzierung statt. Dabei wird bei der ersten normalen Position ein Deltawert berechnet, mit Hilfe dessen alle nachfolgenden Positionen angepasst werden. Danach kann die Position gespeichert werden und die Verarbeitung der Position ist zu Ende.

7 Zusammenfassung und Fazit

Die bestehende TTP-Architektur wurde auf das *Direct und Indirect Location Privacy Problem* untersucht. Da bei der Architektur Pseudonyme zum Einsatz kommen, kann das Direct Location Privacy Problem nahezu ausge-

schlossen werden. Dieses ist nur möglich, wenn sowohl die Versicherung als auch die TTP kompromittiert werden. Mithilfe der Datensätze können die Positionsdaten mit Pseudonymen den wahren Identitäten zugeordnet werden. Dieses lässt sich weiter entschärfen, wenn die TTP die eingereichten Positionsdaten schnell aggregiert. Zu den Indirect Location Privacy Problems, die sich in *Known Place Attack*, *Commuter Attack* und *Observation Attack* aufteilen, wurden verschiedene Verbesserungsmöglichkeiten aufgezeigt. Diese wurden verglichen und bewertet. Die vielversprechendste Möglichkeit jeder Kategorie wurde dabei für die TTP-Architektur umgesetzt. Zum Verhindern der Known Place Attack wurden *Ban-zones* eingeführt. Für diese Ban-zones werden keine Positionsdaten an die TTP geschickt, sondern das mobile Endgerät übernimmt eine einfache Form der Aggregation. Um der Commuter Attack entgegenzuwirken muss die Wegstrecke der Positionen unterbrochen werden bzw. ein Pseudonymwechsel stattfinden. Dieses wurde durch Einsatz von *Mix-zones* erreicht, in dem die mobilen Endgeräte ihr Pseudonym sicher wechseln können. Durch *Temporal Reduction* wird die zeitliche Genauigkeit der Positionsangaben reduziert. Dieses ist möglich, da der Versicherung ungefähre Zeitangaben ausreichen. Dadurch wird ein Angriff durch die Observation Attack erschwert.

Insgesamt wurde für jede Angriffskategorie eine Verbesserung umgesetzt, die diesen Angriff verhindert. Zusätzlich wurde ein privater Fahrmodus implementiert, der, solange aktiv, keine Positionsdaten an die TTP schickt. Hierbei übernimmt das mobile Endgerät die Aggregation.

Ab 2015 sollen alle neuen PKW in der EU über einen eCall (emergency call) verfügen [Eur11]. So sind dann alle Neuwagen mit GPS und GSM ausgestattet. Es liegt nahe, dieses für weitere Anwendungen, wie Navigationsgerät oder Diebstahlschutz zu nutzen. Auch die Anwendung PAYD wird hierdurch attraktiver und könnte einen Aufschwung und eine größere Verbreitung erfahren, zumal sie mit den hier vorgestellten Verbesserungen kaum noch Bedrohungen für die Privatsphäre der Nutzer bedeutet.

Literatur

[BKLZ11] Jens Bertram, Carsten Kleiner, Dennis Ludewig und David Zhang. Privacy for Location-based Services in Pay-as-You-Drive Scenarios. In Martin Werner und Jörg Roth, Hrsg., *8. GI/KuVS-Fachgespräch Ortsbezogene Anwendungen und Dienste*, Berlin, 2011. Logos Verlag.

[BKZ10] Jens Bertram, Carsten Kleiner und David Zhang. Increasing User Privacy in Continuous Location-Based Services. In Alexander Zipf, Sandra Lanig und Michael Bauer, Hrsg., *6. GI/ITG KuVS-Fachgespräch Ortsbezogene Anwendungen und Dienste*, Geographisches Institut der Universitt Heidelberg, Heidelberg, 2010. Heidelberger Geographische Bausteine.

[BS03] Alastair R. Beresford und Frank Stajano. Location Privacy in Pervasive Computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[Dec08] Michael Decker. Location Privacy - An Overview. In *2008 7th International Conference on Mobile Business*, Seiten 221–230. IEEE, Juli 2008.

[Eur11] Europäische Kommission. Digitale Agenda: Europäische Kommission strebt bis 2015 Einführung eines lebensrettenden Notrufsystems für Verkehrsunfälle an, 2011. `http://europa.eu/rapid/pressReleasesAction.do?reference=IP/11/1010`; Abgerufen am 11.07.2012.

[GB11] Torsten J. Gerpott und Sabrina Berg. Pay-As-You-Drive Angebote von Erstversicherern für Privatkunden. *Zeitschrift für die gesamte Versicherungswissenschaft*, 101(1):3–29, 2011.

[GG03] Marco Gruteser und Dirk Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, Jgg. Proceeding of *MobiSys '03*, Seiten 31–42, New York, USA, 2003. ACM.

[GP09] Philippe Golle und Kurt Partridge. On the Anonymity of Home/Work Location Pairs. In Hideyuki Tokuda, Michael Beigl, Adrian Friday, A. J. Bernheim Brush und Yoshito Tobe, Hrsg., *Pervasive Computing*, Jgg. 5538 of *Lecture Notes in Computer Science*, Seiten 390–397. Springer Berlin / Heidelberg, 2009.

[HG05] Baik Hoh und Marco Gruteser. Protecting Location Privacy Through Path Confusion. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks SecureComm 2005*, Seiten 194–205. IEEE, 2005.

[HGXA06] B. Hoh, M. Gruteser, H. Xiong und A. Alrabady. Enhancing Security and Privacy in Traffic-Monitoring Systems. *IEEE Pervasive Computing*, 5(4):38–46, 2006.

[HGXA07] Baik Hoh, Marco Gruteser, Hui Xiong und Ansaf Alrabady. Preserving Privacy in GPS Traces via Uncertainty-Aware Path Cloaking. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, Jgg. 1 of *CCS '07*, Seite 161. ACM Press, 2007.

[HGXA10] Baik Hoh, M. Gruteser, Hui Xiong und A. Alrabady. Achieving Guaranteed Anonymity in GPS Traces via Uncertainty-Aware Path Cloaking. *IEEE Transactions on Mobile Computing*, 9(8):1089–1107, 2010.

[IL06] Muhammad Usman Iqbal und Samsung Lim. A Privacy Preserving GPS-based Pay-as-You-Drive Insurance Scheme. In *International Global Navigation Satellite Systems Society IGNSS Symposium 2006*, number July, 2006.

[JQ10] O. Jorns und G. Quirchmayr. Trust and Privacy in Location-based Services. *E&I Elektrotechnik und Informationstechnik*, 127(5):151–155, 2010.

[KÖ5] Axel Küpper. *Location-Based Services: Fundamentals and Operation*. Wiley, 2005.

[KFKK05] Tobias Kölsch, Lothar Fritsch, Markulf Kohlweiss und Dogan Kesdogan. Privacy for Profitable Location Based Services. In Dieter Hutter und Markus Ullmann, Hrsg., *Security in Pervasive Computing*, Jgg. 3450 of *Lecture Notes in Computer Science*, Seiten 164–178. Springer Berlin / Heidelberg, 2005.

[Kru07] John Krumm. Inference Attacks on Location Tracks. In Anthony LaMarca, Marc Langheinrich und Khai Truong, Hrsg., *Pervasive Computing*, Jgg. 10 of *Lecture Notes in Computer Science*, Kapitel 8, Seiten 127–143. Springer Berlin / Heidelberg, 2007.

[Kru08] John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2008.

[Swe02] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[TDK+11] Carmela Troncoso, George Danezis, Eleni Kosta, Josep Balasch und Bart Preneel. PriPAYD : Privacy-Friendly Pay-As-You-Drive Insurance. *IEEE Transactions On Dependable and Secure Computing*, 8(5):742–755, 2011.

[WWF09] Yiming Wang, Lingyu Wang und B. C. M. Fung. Preserving Privacy for Location-Based Services with Continuous Queries. In *IEEE International Conference on Communications ICC 09*, Seiten 1–5. IEEE, 2009.

A Spatial Hashtable Optimized for Mobile Storage on Smart Phones

Jörg Roth

Department of Computer Science
Univ. of Applied Sciences Nuremberg
Kesslerplatz 12
90489 Nuremberg, Germany
Joerg.Roth@Ohm-Hochschule.de

Abstract: This paper describes an approach to access large amounts of geo data on mobile devices with small runtime memories. We have this problem, e.g., if we want to execute a route planning application on smart phones that do not have server access. The *Spatial Hashtable* approach first orders all geo objects by their locality. This increases the probability to remain at the same hashtable position for subsequent queries and decreases the transfer from flash to runtime memory. Spatial Hashtables also provide spatial indexes that quickly allow an application to access entries by geometric conditions. We present three Spatial Hashtable mechanisms: *Pivot*, *Stripes* and *Double Stripes*.

1 Introduction

Mobile location-based applications such as tour guides often access spatial data via a mobile network on a central server. In some scenarios, however, it is reasonable to store a certain amount of spatial data on the mobile device. Reasons could be: a mobile network is not available in certain scenarios, it is too costly, or too slow. In addition, a certain business model could prefer not to install a service but to put the geo data on the mobile device.

Even though big mobile storages are nowadays available as flash memory (e.g. SD cards with 16-64 GB), the actual runtime storage for the heap in the internal memory is considerably small (e.g. 64 MB in current Android smart phones). Unfortunately, to load a block of data from flash memory takes (dependent on the real hardware) approx. 200 times longer compared to load a block from the internal memory.

For certain applications huge amounts of geo data are involved. For route planning in Germany, more than 2 million streets and over 10 million crossings have to be stored. The required space is approx. 1 GB. Thus, we have to think about a mechanism to load blocks of interest from the flash memory into the internal memory. In order to have a certain amount of related entries available in a specific block, we have to *localize* all geo data.

In this paper we introduce the notion of the *Spatial Hashtable* – a structure that combines a traditional hashtable with the ability to look up spatial data by geometric conditions. We present three approaches to localize the data.

2 Related Work

The problem has certain similarities to virtual memories in operating system [Vi01]. They also are based on a locality of objects. Whereas this type of locality means that related references have similar addresses, our locality means something different: objects with similar addresses describe areas in the real world that are spatially close together. Thus, virtual memory mechanisms can be used as underlying approach (especially swapping strategies), but we first have to achieve *spatial* locality of entries.

[MH97] assumes a grid with equal geographical tile size. Each tile contains the geo objects that reside in the respective geographical area, thus tiles do not contain equal numbers of entries. The approach strongly relies on virtual memory methods in the background. Once an entry inside a tile is accessed, the respective memory page is cached and thus access to proximate entries is faster. To improve the idea of grids, so called space-filling curves can order grid tiles in a specific ordering to preserve proximity [ARR+97]. However, grids always have the problem that geo data with an inhomogeneous geographical distribution (the usual case) lead to tiles with different number of entries.

Other approaches such as [ZXL02, HS03] focus on the replacement strategy for cache entries. They provide strategies to find out appropriate cache entries based on their geographic properties. For the special case of route planning there exist mechanisms that support a graph algorithm (usually a variation of A*) to access the routing network on the external memory [BB04]. Note that in this paper we do not focus on cache replacement strategies. We strongly believe that once geo data is spatially clustered, even non-spatial cache replacement strategies highly perform.

The Spatial Hashtable approach presented in this paper is based on ongoing research on geo data. *HomeRun* [Ro10a, Ro10b] is a platform for low-cost development of location-based services, especially of small services outside the mass market. HomeRun provides a set of basic services, e.g. the import of geo data from public sources. It contains an efficient communication infrastructure for service usage and offers functional blocks for mobile devices, e.g. to display maps. As basic functions already exist, a service developer can concentrate on the actual application. A development based on HomeRun is cost-effective and due to the modularization, can easily be modified later. As a major direction, we want to provide location-based services on smart phones (online and offline). In [Ro11] we presented a first approach to store and access huge amounts of geo data on smart phones. It was based on a relational database that used a special spatial indexing mechanism [Ro09]. It turned out that, even though relational databases are a convenient way to access geo data (even on smart phones), we suffer from low performance. For applications such as route planning, relational databases are not appropriate.

3 The Spatial Hashtable

3.1 Basic Considerations

Our Spatial Hashtable is based on the following assumptions:

- The spatial data is completely mastered on a development system that exports an optimized structure for the mobile device. The mobile device mainly reads spatial data rather than changing it.
- On the mobile device, the transfer between flash memory and runtime heap is considered as the major performance bottleneck. Once inside the heap, spatial data can be processed quickly, as current smart phones have fast CPUs.
- We expect a mobile application to perform queries with spatial locality, i.e. successive queries usually affect data of the same geographic region.
- Data entries can be loaded using two mechanisms. First: the application can use an index to address an entry, similar to an array index. Second, an application can formulate a certain spatial condition such as: all entries that are located inside a certain geographic rectangle.
- Additionally, we assume that all entries have the same structure with fixed fields, i.e. the Spatial Hashtable can be compared to a relational database table.

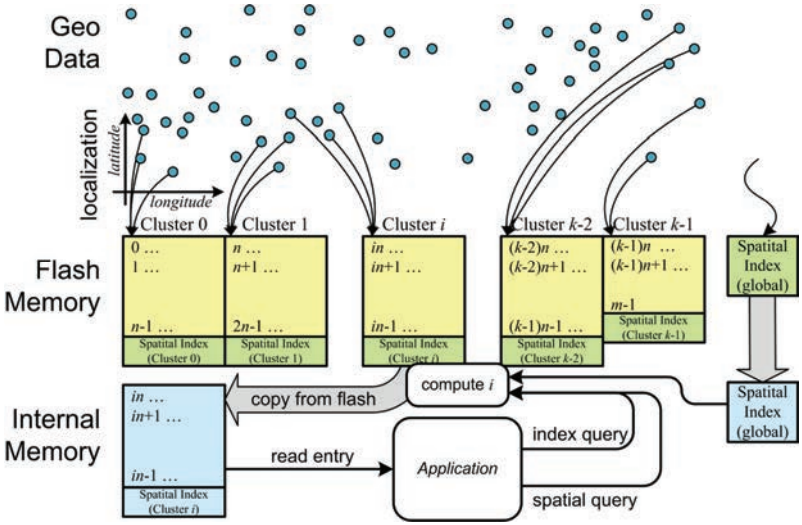


Figure 1: Basic Spatial Hashtable idea

Fig. 1 illustrates the Spatial Hashtable idea. We assume an amount of m geo data entries. We divide the overall spatial data into k clusters that are stored in separate files in the flash memory. The number of entries per cluster n is constant except for the last cluster that gets the rest of m entries. The mapping of the geo data entries to the specific cluster file is called *localization*. This step is essential for the later runtime performance. We present three localization mechanisms *Pivot*, *Stripes* and *Double Stripes* in more detail later.

One cluster file (or more if we use caching) has to fit into the smart phone's heap memory. The application can load cluster files via two mechanisms: first, it can access a geo data entry by its index. For an index x, the cluster file that contains this entry has the number $i=\lfloor x/n \rfloor$. The second mechanism provides geometric queries. For this, spatial indexes are written to flash memory during the localization procedure. A *global* spatial index is used to identify clusters that fulfill certain geometric conditions. Spatial indexes for each cluster are used to spatially search entries in a cluster.

Note that the loading and accessing mechanism is shielded behind an API. The application simply formulates a query for a certain entry – the runtime system executes the computation of the specific cluster index and loads it into memory transparently. Further note that in real systems, we allow multiple clusters to reside in the internal memory. This dramatically increases the hit rate. Note that, as the clusters are already localized, we can use simple cache replacement strategies such as LRU (Least Recently Used) that do *not* consider the spatiality of entries.

Localizing Geo Data

The main objective of the localizing algorithm is to define geographical regions with identical numbers of entries that divide the overall space without overlap. The probability for multiple requests to remain in the same cluster file should be high. Fig. 2 illustrates the problem.

Figure 2: The localization problem

All geo data entries (fig. 2 left) are distributed in a two-dimensional space. Usually we have different densities, e.g. a higher number of entries in cities compared to the countryside. Fig. 2 middle shows a segmentation of the space into clusters with equal numbers of entries. Obviously, there exist different ways to divide the space into clusters. Ideal clusters have similar spatial heights and widths, as the probability for nearby entries to reside in the same cluster is higher.

We require equal cluster sizes (in terms of record numbers, *not* area sizes) as we want to increase the usage of the spare internal memory of the smart phone. If cluster sizes were not equal, the reserved space for geo data is usually not fully used. Thus a simple tile grid (fig. 2 right) is not suitable for our approach, as only the spatial sizes would be equal, but not the number of entries.

In the remaining paper we use the German road map topology as example geo data to show the Spatial Hashtable mechanisms. Table 1 shows the corresponding numbers of entries.

Table 1: Typical amounts of data for route planning (Germany)

	Records in million	Bytes per record	Total size in MB
Crossings	4.7	84	374
Links between crossings	10.9	27	280
Complete roads including names	2.3	139	310
		Sum	964

In the following, we focus on the localization mechanism and present three approaches.

3.2 Pivot Localization

The idea behind the Pivot localization is to build a cluster around a selected point called *Pivot* point. After the Pivot point and *n*-1 nearest points are clustered, a next Pivot point is selected and so forth.

The pseudo code of the Pivot localization is as follows:

1. Select a Pivot entry of all unselected entries (see below).
2. Sort all unselected entries by their distance to the Pivot entry.
3. Select *n*-1 nearest entries (or less for the last cluster). The Pivot entry and these *n*-1 entries form a new cluster.
4. Continue with (1) until all entries are selected.

Fig. 3 illustrates the approach.

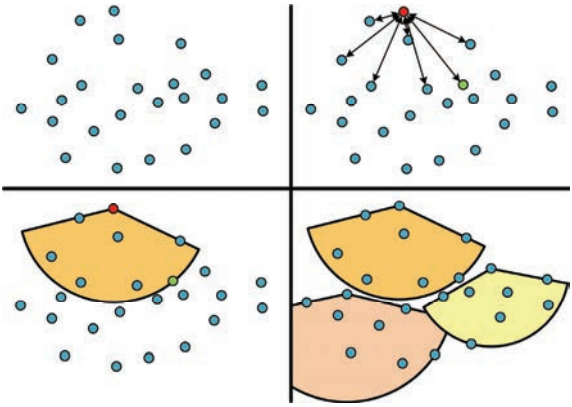


Figure 3: The Pivot localization

The selection of the Pivot point is important to form compact clusters. Especially the last clusters often get large and segmented if we used inappropriate Pivot points. There can be different ways to select the Pivot point, e.g.

- select the most northern unselected point;
- first compute the geometric centre of all points. For each round select the unselected point nearest to the centre.

It turned out that selecting the most northern point is an appropriate and simple to compute approach. Experiments show that the segmentation of the last clusters is less than in other approaches. Fig. 4 shows localization results of all German crossings (cluster size 800 kB).

During the localization procedure we store the coordinates p_i of the Pivot entries and the maximum distances r_i between all cluster entries to the Pivot entry. We store this list in the runtime memory, to quickly look up the appropriate cluster file. To find the cluster that contains a certain position q we proceed as follows:

1. Iterate through the list (p_i, r_i)
2. If distance(q, p_i) $\leq r_i$: stop with success, i is the cluster file
3. If the list is processed without any hit: stop with failure

Here, distance() computes the spatial distance between two coordinates. Note that this code assumes that q is actually available as entry. Usually, we perform a radius search, i.e. we want to look up all cluster files that contain entries inside a certain distance s to the given point q:

1. Iterate through the list (p_i, r_i)
2. If distance(q, p_i) $\leq r_i+s$: i is *one* cluster file (continue loop)
3. Process the entire list, the result is a list of selected cluster files.

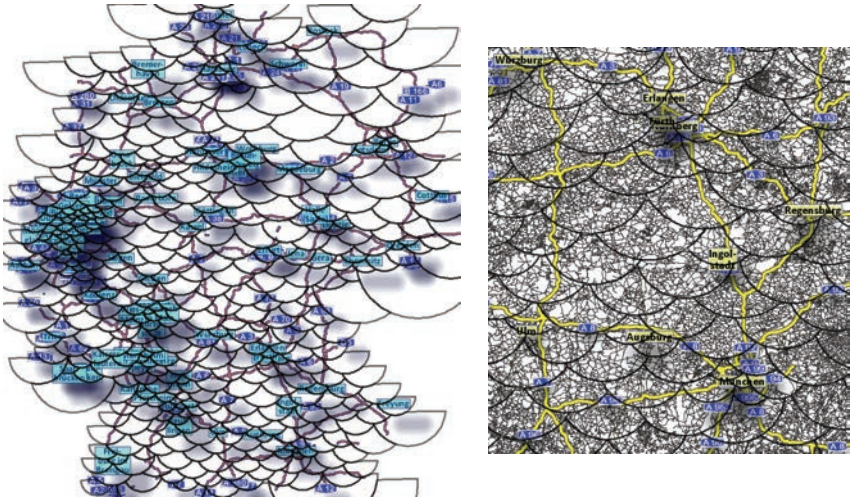


Figure 4: Pivot localization results

Note that this approach sometimes selects clusters that geometrically cannot overlap with the search area. This is because the actual cluster shapes are not circles; instead, they are circles where former cluster shapes are subtracted. An exact computation of the geometric condition would be computationally too expensive. However, if we use the most northern entry as Pivot entry, we can formulate a further condition to step (2):

$$\text{distance}((p_{i,lat}, q_{long}), (q_{lat}, q_{long})) \leq s$$

(Note that the $(p_{i,lat}, q_{long})$ is not a write error.)

In reality we do not linearly iterate through the list of clusters but use a spatial index (see section 3.5). This pseudo code is only to illustrate the mechanisms.

3.3 Stripes Localization

The Stripes localization follows another direction to form clusters. Only rectangular clusters are created. The benefit: spatial indexing is simplified as most indexing mechanisms are optimized for rectangular shapes.

As earlier mentioned, a simple grid is not appropriate as the number of entries per grid depends on the geometrical distribution density. Thus our idea is to first consider geometrical stripes that contain multiplies of n entries. These stripes are divided vertically with clusters of n entries. To get optimal clusters, we test multiple stripe widths and measure the quality of the resulting clusters using a function q.

The pseudo code of the Stripes localization is as follows:

1. Order all entries west to east.
2. Let u denote the number of not-clustered entries. Iterate i from 1 to $\lceil u/n \rceil$.
3. Create a meridian stripe of $\min(i \cdot n, u)$ entries starting from the west border of non-clustered entries.
4. Create clusters of n entries (of less for the last cluster) selecting entries from north to south.

5. For each cluster, compute the value q. Let the overall stripe value $q_i=\sum q/i$.
6. Continue iterating through i. Let i_{min} be the stripe index with the minimal value q_i.
7. Create clusters for i_{min} as tested in steps (3) and (4).
8. Continue with (2) until all entries are selected.

Fig. 5 illustrates the approach.

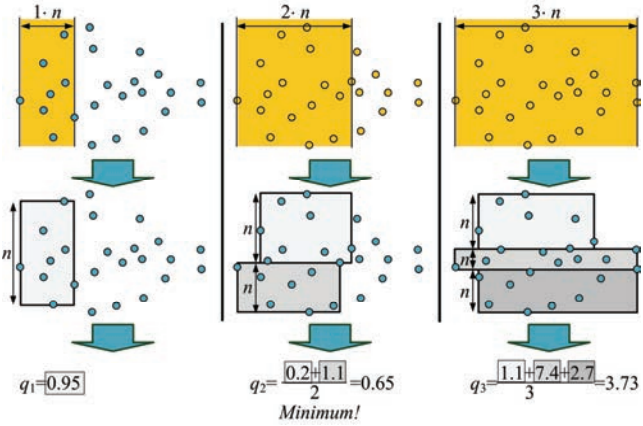


Figure 5: The Stripes localization

The function q should give low values for 'good' cluster shapes. The value for squares should be 0. In addition, turned rectangles (by 90°) should return the same value. We define q as follows:

$$q(\Delta_{lat},\Delta_{long})=\begin{cases}\dfrac{\Delta_{lat}}{\Delta_{long}}-1 & \text{if } \Delta_{lat}>\Delta_{long}\\ \dfrac{\Delta_{long}}{\Delta_{lat}}-1 & \text{otherwise}\end{cases}$$

for clusters with geographical extent Δ_{lat} and Δ_{long} (see fig. 6).

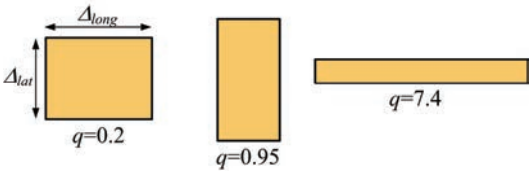


Figure 6: q value examples

Fig. 7 shows localization results of all German crossings (cluster size 800 kB).

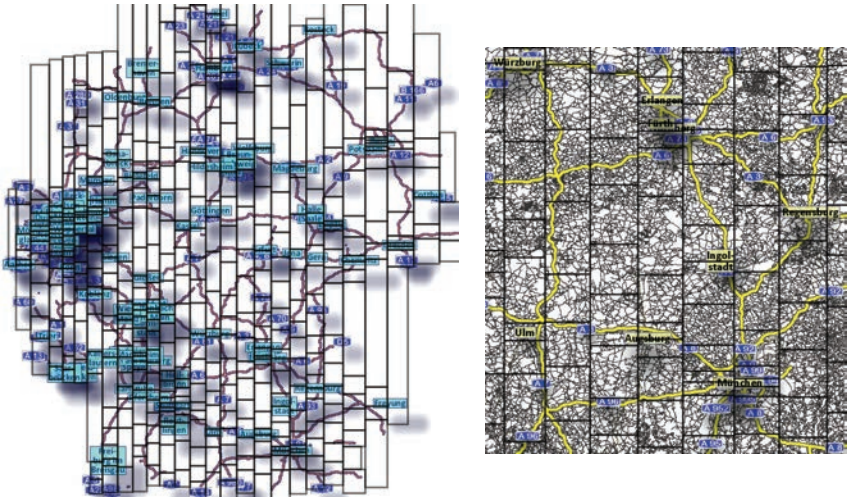


Figure 7: Stripes localization results

For spatial indexing, the localization mechanism collects the cluster's border ($north_i$, $south_i$, $west_i$, $east_i$). We perform these steps to look up all cluster files that contain entries inside a certain distance s to the given point q:

1. Iterate through the list ($north_i$, $south_i$, $west_i$, $east_i$)
2. If $q_{lat} \oplus s \leq north_i$ AND $q_{lat} \oplus -s \geq south_i$ AND $q_{long} \oplus s \geq west_i$ AND $q_{lat} \oplus -s \leq east_i$ then i is one cluster file (continue loop)
3. Process the entire list, the result is a list of selected cluster files.

Here, $\oplus$ denotes a function that moves a latitude or longitude value a certain amount of meters. Again note that in reality, we do not iterate through the list of clusters but use a spatial index (see section 3.5).

3.4 Double Stripes Localization

The Stripes localization has a certain disadvantage: stripes that both cover zones with high and low densities result in inhomogeneous cluster shapes: clusters that cover high densities tend to horizontal bars, whereas clusters that cover low densities tend to vertical bars. This is because a stripe defines the horizontal extent of all clusters inside the stripe. Unfortunately, typical geo databases both contain high density zones (e.g. cities) and low density zones (countryside).

The Double Stripes localization addresses this problem: stripes can be both divided vertically and horizontally. Fig. 8 illustrates problem. For a given distribution the Stripes localization generates horizontal bars for high density zones (left). Dividing the lower zone horizontally would result in a more suitable solution (right).

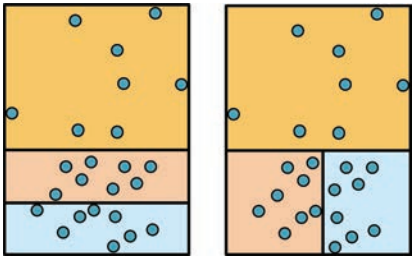


Figure 8: The problem of the Stripes localization

The Double Stripes localization uses the Stripes algorithm but replaces step (4). Here, we do not only test a single partitioning from north to south but test all permutations of horizontal and vertical partitions as illustrated in fig. 9.

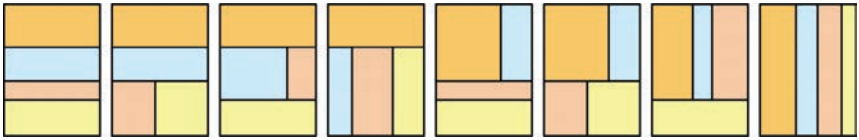


Figure 9: Eight possible ways to create four clusters for a given stripe

For each permutation we again compute the value q and look for the overall minimum.

Some considerations to the computational costs: For stripes with i clusters, there exists 2^{i-1} variations to partition a stripe. For stripe sizes larger than 25 clusters, it is too expensive to check all variations. Thus it is reasonable to formulate the loop in step (2): Iterate i from 1 to $\min(\lceil m/n \rceil, c)$, where c is a constant that limits the stripe size.

To reduce the number of variations even more, we additionally can limit the number of horizontal partitions by a number λ. For e.g., λ=3 we only allow 1, 2 or 3 horizontal partitions. For λ=2, the total number of variations for i clusters is the Fibonacci number F_{i+1}. For large numbers we can use the approximation

$$F_{i+1} \approx \frac{5+3\sqrt{5}}{10} \cdot \left(\frac{1+\sqrt{5}}{2} \right)^{i-1} \approx 1.171 \cdot 1.618^{i-1}$$

which is far lower than 2^{i-1}. However for larger values of λ, the benefit gets lower. Note that for λ=1 the Double Stripes approach produces the Stripes approach, thus Double Stripes could be considered as the more general approach.

For large stripes, however, the number of permutations still is far too large. Thus, we consider a third type of limitation that is based on the following observation: if we got an optimal partitioning of a stripe, every horizontal cluster border marks two optimal partitionings – one north of, one south of the border. Thus, we only have to "guess" an optimal border between north and south and we can check each part individually then. To check all borders recursively would lead to the same number of permutations. We thus use the following approach: Let μ be the maximum stripe size that is fully checked and Π be a set of numbers, each of it less than μ. We define a recursive function *getBest*:

```
getBest(from, count)
    if (count≤μ)
        return the best of all permutations of clusters (from… from+count-1)
    else
        for each j∈Π
            getBest(from, j)
            getBest(from+j, count-j)
            compute joined value of the two results
        return the best of the joined values
```

With the help of this approach, we can reduce the number of permutations that have to be checked from several billion to some thousands. Fig. 10 shows localization results of all German crossings with λ=4, c=50; μ=10; Π={5, 7}, cluster size 800 kB.

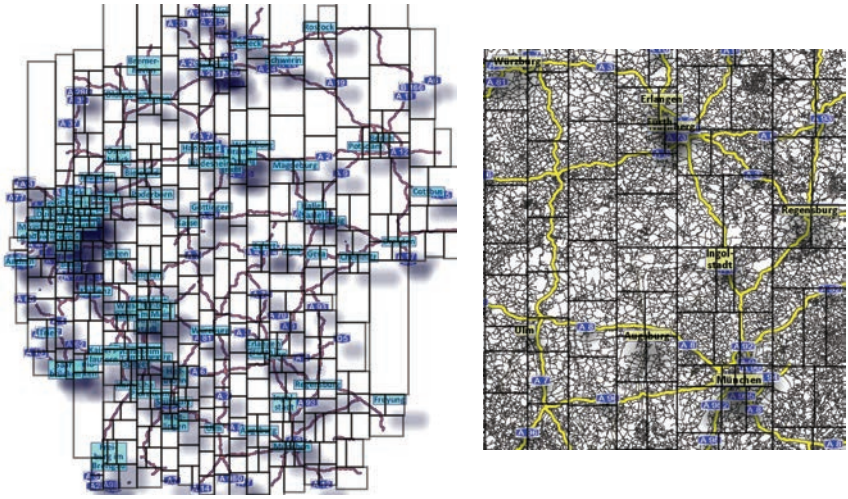


Figure 10: Double Stripes localization results

This approach, although computationally expensive, significantly improves the cluster shape. Table 2 shows a comparison of *q* values for the Germany data.

Table 2: Comparison of average *q* values

Localization	Cluster size 800 kB	Cluster size 400 kB
Stripes	1.182	1.371
Double Stripes	0.449	0.456

As a major result, Double Stripes rectangles are more square-like with a ratio of approx. 1.45 between the two lengths.

3.5 Spatial Indexing

To get a specific geo entry by its geometry we have to perform two steps:

- Identify the cluster file that fulfils the geometric condition.
- Read the entry inside the cluster file that fulfils the geometric condition.

Note that, dependent on the specific geometric condition, more than one cluster file and multiple entries may fulfill the condition. As multiple cluster files may not fit into the runtime memory, we assume that the application iterates through the result set in a cursor-based manner.

We speed up both lookup steps using an R-Tree spatial index [Gut84]. The *Gobal Spatial Index* (fig. 1) contains the borders of the cluster files. Separate spatial indexes inside the clusters provide access the specific entries. Note that we can pre-compute these R-Trees. This especially means that they can be balanced in order to decrease the average access time. R-Trees are typically used in a dynamic environment where geo data may change and the R-Trees must be balanced 'on the fly'. In contrast, we can spend time to balance the trees offline.

R-Trees use rectangular areas to approximate geometries. Thus, Stripes and Double Stripes are suitable approaches as they already provide rectangular cluster shapes. In contrast Pivot cluster areas may cause additional lookup time as the actual geometry has to be checked.

4 Evaluation

We want to investigate the benefit of the localization methods. An application that processes geo data should work as long as possible with a specific cluster file. We made two analyses:

1. How high is the probability for two entries in a certain spatial distance to reside in the same cluster file? This value can also be considered as a hit rate for cluster files.
2. In the street network: if two crossings are connected by a street – how high is the probability that these crossings reside in the same cluster file? This analysis is strongly related to the route planning application: following streets from crossing to crossing is a basic operation inside the path finding algorithm.

Besides our three localization methods we integrated another method as a comparison reference. We call it *Northsouth*: we just order all entries from north to south. This method randomizes the spatial relation as east to west proximity is destroyed. The Northsouth method thus represents a 'no localization' approach.

4.1 Proximity Hit Rate

We created two Spatial Hashtables from the 4.7 million crossings of Germany with a total size of 374 MB. We checked with two cluster sizes 400 kB and 800 kB (table 3).

Table 3: Cluster properties for different cluster sizes

	Cluster size 400 kB	**Cluster size 800 kB**
Entries per cluster	4876	9752
Number of clusters	959	480

Fig. 11 represents the outcome of the first analysis. For two pairs of entries with a certain distance we checked if they are in the same cluster file. We could also take this view: if an application loaded a certain entry and wants to load another entry in a certain distance: what is the expected hit rate to remain in the same cluster file?

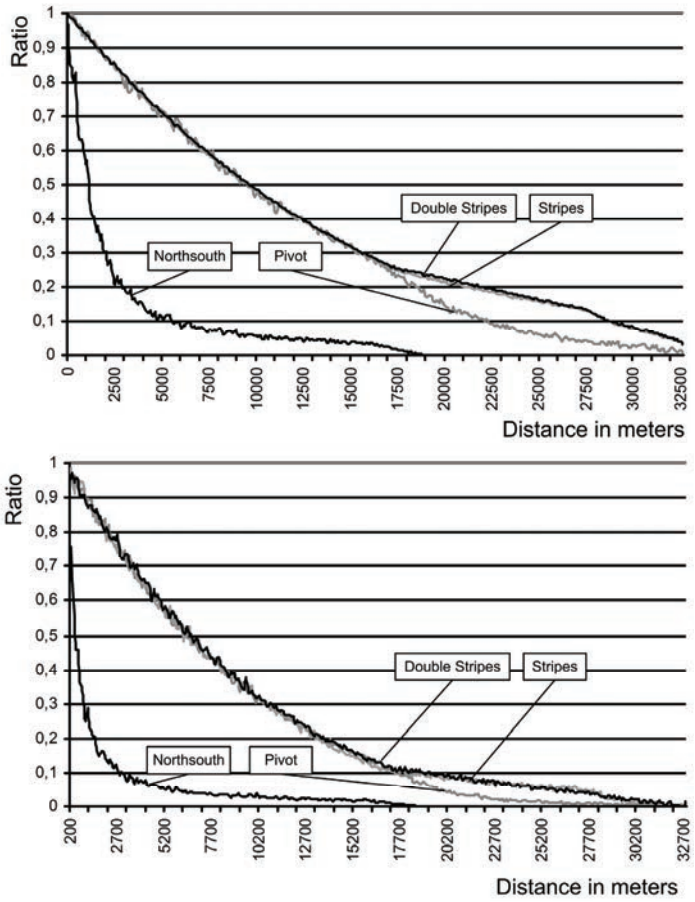


Figure 11: Ratio of two entries in a certain distance that are in the same cluster file
(cluster size 800 kB top, 400 kB bottom)

The main observations:

- The three real localizations are much better than the Northsouth localization.
- Pivot, Stripes and Double Stripes only slightly differ. Beyond a certain distance, Pivot is less effective. Stripes and Double Stripes provide virtually the same results.

Even though this analysis justifies the localization as such, the specific method does not seem to have significant influence on the effectiveness. This is a disappointing result, especially if we consider the complexity of Double Stripes.

4.2 Effectiveness Considering a Certain Application

We could argue that the first evaluation was set in an artificial scenario as it does not consider real application requests. In the second analysis we use a route planning application [Ro12] that accesses a street network of crossings and streets. In the underlying A* algorithm, the basic operation is to follow a street segment from a given crossing to the next crossing. It is important that we have a high probability to remain in the same cluster file. Table 4 represent the 'inverse' case: how many connected crossings are in different cluster files?

Table 4: Ratio of connected crossings that are in different cluster files

	% Ratio (avg. all clusters)		% Ratio (worst cluster)	
Cluster size	400 kB	800 kB	400 kB	800 kB
Pivot	1.09	0.73	1.99	1.46
Stripes	1.10	0.75	2.54	1.37
Double Stripes	1.07	0.73	1.84	1.19
Northsouth	10.43	5.39	16.31	8.44

Again, the difference to Northsouth is significant, whereas the differences between Pivot, Stripes and Double Stripes are low. Double Stripes is slightly better than the others.

If we take into account these analyses and the different properties, we can sum up:

- Localization is a useful tool to store large amounts of geo data.
- The runtime complexity of Double Stripes cannot be justified in comparison to Stripes. Even though Double Stripes constructs more square-like clusters, the benefit does not pay off.
- Pivot localization is easy to implement and provides good results. Its only drawback is non-rectangular cluster regions, which may be difficult for spatial indexing.

We used the Double Stripes localization with 800 kB clusters to implement the *donavio* routing planning [Ro12]. All required routing data for Germany, i.e. the street topology, street geometries, names, and driving properties such speed limits allocate a storage space of approx. 1.3 GB on the smart phone's SD card. The runtime heap in contrast only has a size of max. 64 MB. As the routing algorithm A* usually performs several queries within the same region, our approach is suitable. It takes only some seconds to compute a fastest route from point to point.

5 Conclusions

We presented the *Spatial Hashtable* approach and introduced three mechanisms *Pivot*, *Stripes* and *Double Stripes* to localize geo data. It turned out that localization is an important mechanism to ensure performance even on small systems like smart phones.

Currently, we spatially represent geo data entries by a single point (i.e. its geometric center). Larger objects such as city borders would cut multiple cluster file borders, thus the existing approach is not suitable. In the future we work on an extended version of Spatial Hashtables that also can hold such objects.

References

[ARR+97] Asano, T.; Ranjan, D.; Roos, T.; Welzl, E.; Widmayer, P.: Space-filling curves and their use in the design of geometric data structures, Theoretical Computer Science, Vol. 181, Issue 1, 1997, 3–15

[BB04] Breunig, M.; Baer, W.: Database support for mobile route planning systems, Computers, Environment and Urban Systems, Vol. 28, Issue 6, 2004, 595–610

[Gut84] Guttman A.: R-Trees: A Dynamic Index Structure for Spatial Searching, Proc. of the 1984 ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, June 1984, 47-57

[HS03] Höpfner, H.; Sattler, K. U.: Towards Trie-Based Query Caching in Mobile DBS, in proceedings of the Workshop Scalability, Persistence, Transactions- Database Mechanisms for Mobile Applications, Lecture Notes in Informatics (LNI), 2003, 106-121

[MH97] McCormack, J. E.; Hoog, J.: Virtual Memory Tiling for Spatial Data Handling in GIS, Computer & Geosciences Vol. 23, No. 6, 1997, 659-669

[Ro09] Roth, J.: The Extended Split Index to Efficiently Store and Retrieve Spatial Data With Standard Databases, IADIS International Conference Applied Computing 2009, Rom (Italy), 19.-21. Nov. 2009, Vol. I, 85-92

[Ro10a] Roth, J.: Die HomeRun-Plattform für ortsbezogene Dienste außerhalb des Massenmarktes, in Zipf A., Lanig S., Bauer M. (Hrsg.) 6. GI/ITG KuVS Fachgespräch "Ortsbezogene Anwendungen und Dienste", Heidelberger Geographische Bausteine Heft 18, 2010, 1-9

[Ro10b] Roth, J.: Übernahme von Geodatenbeständen aus Open Street Map und Bereitstellung einer effizienten Zugriffsmöglichkeit für ortsbezogene Dienste, Praxis der Informationsverarbeitung und Kommunikation (PIK), Vol. 13, Heft 4, 2010, 268-277

[Ro11] Roth, J.: Moving Geo Databases to Smart Phones – An Approach for Offline Location-based Applications, Innovative Internet Computing Systems (I2CS), Berlin, 15.-17. Juni 2011, GI Lecture Notes in Informatics, Vol. P-186, 228-238

[Ro12] Roth, J.: Modularisierte Routenplanung mit der donavio-Umgebung, 9. GI/ITG KuVS Fachgespräch "Ortsbezogene Anwendungen und Dienste", 2012

[Vi01] Vitter, J. S.: External memory algorithms and data structures: dealing with massive data, ACM Computing Surveys (CSUR), Vol. 33 Issue 2, June 2001, 209-271

[ZXL02] Zheng, B.; Xu, J.; Lee, D. L.: Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments, IEEE Trans. Comput., Vol. 51 No. 10, 2002, 1141–1153

Integration und Bereitstellung von ortsbezogenen Daten für mobile AR-Anwendungen

Frank Fuchs-Kittowski, Stefan Simroth, Sebastian Himberger, Fabian Fischer

HTW Berlin
Fachbereich 2 - Umweltinformatik
Wilhelminenhofstr. 75a
12459 Berlin
[frank.fuchs-kittowski|simroth|himberger|fischer]@htw-berlin.de

Abstract: In diesem Beitrag wird eine Content-Plattform für die effiziente Realisierung von AR-Anwendungen auf mobilen Endgeräte (Smartphones, Tablets etc.) beschrieben. Ausgehend von Anwendungsfeldern im Hochwasserschutz und der Hydrologie werden Anforderungen an eine solche mAR-Content-Plattform dargestellt. Es wird das Konzept einer den gesamten Wertschöpfungsprozess von AR-Inhalten abdeckenden AR Content-Plattform sowie dessen konsequenter Umsetzung beschrieben. Zudem werden auf dieser Plattform aufsetzende mAR-Apps aus dem Bereich des Hochwasserschutzes und der Hydrologie vorgestellt.

1 Einleitung

Ortsbasierte Dienste haben innerhalb der letzten Jahre vor allem durch die Verfügbarkeit leistungsfähiger und intuitiv benutzbarer Smartphones eine starke Verbreitung erfahren [WR11]. Die massenhafte Verbreitung mobiler Endgeräte bietet Unternehmen aber auch Behörden die Möglichkeit, ortsbezogene Informationen in die Breite zu bringen. D.h. ortsbezogene Dienste und Anwendungen werden nicht nur für wenige Fachexperten entwickelt, sondern stehen einer riesigen Anzahl von Nutzern zur Verfügung. Dies können sowohl die Mitarbeiter eines Unternehmens als auch Bürger sein - Letztere z.B. im Falle der Information über Hochwassergefahren [FWT12].

Während ortsbezogene Informationen oftmals auf Karten oder in Listenform dargestellt werden, besteht eine neuartige, innovative Nutzerschnittstelle in der Darstellung der Information als Erweiterte Realität im Kamerabild des Smartphones. Der Begriff Erweiterte Realität (Augmented Reality, oder kurz AR) bezeichnet die Ergänzung der optischen menschlichen Wahrnehmung der Realität mit digitalen, kontextabhängigen Informationen [Az97]. Bei der mobilen Augmented Reality (mAR) werden mobile Endgeräte dazu genutzt, um die gemeinsame Wahrnehmung von realen und digitalen Informationen möglich zu machen [HFT99].

Mit der neuen Generation an leistungsstarken, handlichen und kostengünstigen mobilen Endgeräten (Smartphones, Tablets etc.) steht die für mAR erforderliche Hardware einer breiten Masse zur Verfügung. Die Hardware wird ergänzt durch plattformübergreifend vorhandene Augmented Reality-Browser für die Darstellung von Inhalten als Erweiterte Realität. Folglich steht nun eine technische Infrastruktur zur Verfügung, die eine kostengünstige Entwicklung mobiler AR-Anwendungen sowie eine massenhafte Nutzung dieser AR-Anwendungen durch Jedermann ermöglicht [FS12].

Mobiler erweiterter Realität wird daher ein großes wirtschaftliches Potenzial zugesprochen [Be11]. Der kommerzielle Erfolg von mAR wird aber stark von der Verfügbarkeit von geeigneten AR-Inhalten abhängen. Bisher ist die erforderliche Bereitstellung der AR-Inhalte eine schwierige, komplexe und aufwendige Aufgabe. Sie erfordert Expertenwissen und Programmieraufwand. Dies behindert die Einführung, Nutzung und (massenhafte) Verbreitung von mAR und hat einen negativen Einfluss auf die Realisierung von nutzenstiftenden mAR-Szenarien.

In diesem Beitrag werden das Konzept einer mAR-Informations-Infrastruktur und dessen Realisierung in Form der holgAR-Content-Plattform beschrieben. Diese soll den Prozess der Erzeugung, Integration, Verwaltung und Bereitstellung von ortsbezogenen Daten für mAR-Anwendungen vereinfachen und beschleunigen. Mit dieser Plattform wird der gesamte Content-Management-Prozess von mAR-Inhalten abgedeckt und somit die Entwicklung von mobilen AR-Anwendungen vereinfacht und effizienter gestaltet. Es können somit die Potenziale der noch jungen Technologie weiter ausgeschöpft werden.

Der Beitrag ist wie folgt strukturiert: Im anschließenden Kapitel 2 wird zunächst eine kurze Einführung in das Thema mAR gegeben und der Stand der Technik von Content-Plattformen für mAR diskutiert. Ausgehend von verschiedenen mAR-Anwendungsfeldern im Hochwasserschutz und der Hydrologie, die in Kapitel 3 anhand von konkreten mAR-Apps beschrieben werden, werden dann in Kapitel 4 Anforderungen an eine solche mAR-Content-Plattform abgeleitet. In Kapitel 5 werden grundlegende Konzepte einer mAR-Infrastruktur dargestellt, die die zuvor genannten Anforderungen erfüllt. Darauf aufbauend wird in Kapitel 6 die Implementierung der holgAR-Content-Plattform skizziert. Abschließend werden in Kapitel 7 aus den Erfahrungen der Umsetzung und des Einsatzes der holgAR-Content-Plattform der Schlussfolgerungen für die Potenziale von mAR gezogen.

2 State of the art - Mobile Augmented Reality (mAR)

Wie zuvor das Internet revolutionieren nun mobile Geräte wie Smartphones den Informationsaustausch und die Kommunikation. Sie ermöglichen die Verschmelzung von realer Welt mit digitalen Informationen zu einer „Erweiterten Realität“ für Jedermann. Der Begriff Erweiterte Realität (Augmented Reality, oder kurz AR) bedeutet die Echtzeit-Überlagerung der menschlichen Wahrnehmung der Realität mit digitalen (kontextabhängigen) Informationen [Az99]. Bei der mobilen Augmented Reality (mAR) werden mobile Endgeräte zur Verschmelzung realer und digitaler Welt genutzt, um die gemeinsame Wahrnehmung von realen und digitalen Informationen im Ortskontext möglich zu machen [HFT99].

Dabei werden digitale Informationen kontextbezogen in Echtzeit im Kamerabild des mobilen Geräts dargestellt. Dadurch werden die digitalen, geokodierten Informationen in ihren räumlichen Kontext gesetzt sowie die Sicht auf die reale Welt durch diese Informationen angereichert. Dies ermöglicht einerseits eine neuartige Wahrnehmung des Ortes durch die Anreichung mit Informationen aus Vergangenheit, Gegenwart oder Zukunft, z.B. die Darstellung eines Bauwerks, das heute nicht mehr oder noch nicht sichtbar ist. Anderseits ermöglicht dies auch ein besseres Verständnis und Analyse von digitalen Daten vor Ort und somit eine bessere Entscheidungsfindung.

Bislang war mobile Erweiterte Realität (mAR) vor allem Grundlagenforschung mit wenigen teuren Spezialanwendungen für wenige Fachexperten. Solche Anwendungen stellen hohe Anforderungen an Hardware und Software, was den breiten Einsatz in der Praxis behindert hat. Typischer Weise handelte es sich dabei um sperrige Spezial-Konfigurationen aus Laptop mit Tragegestell, Datenbrille (Head Mounted Display, HMD), Kamera, einem speziellen Eingabegerät (Handheld) und Trackingsystem [FMH97]. Die Geräte waren komplex, teuer sowie unhandlich (Abb. 1). Zusätzlich war ein hoher Aufwand für die Entwicklung von Software für die Darstellung der Inhalte auf diesen Geräten erforderlich [Tö10].

Abbildung 1: mAR-System zur Visualisierung von Hochwasserereignissen [Co04]

Aufgrund des rasanten Fortschritts in der Entwicklung mobiler Endgeräte verfügen Geräte der neueren Generation über die notwendige Rechenleistung sowie die erforderlichen Sensoren (GPS, Kompass, Beschleunigung) und Komponenten (Display, Videokamera), um mAR realisieren zu können. Eine schnelle Internetverbindung erlaubt es, die Inhalte ortsbezogen und on-demand auszuliefern [Sch11]. Außerdem sind die Geräte kostengünstig, leicht handhabbar und massenhaft verbreitet. Neben dieser neuen Hardware existiert seit kurzem auch spezielle Software - sog. AR-Browser - (z.B. Layar[1], Wikitude[2], Junaio[3], Argon[4]), welche die Darstellung von Informationen kontextbezogen und in Echtzeit im Kamerabild des mobilen Geräts (Smartphone etc.) ermöglicht. Dabei kann es sich um unterschiedliche Arten von Informationen handeln. Möglich sind zum Beispiel Textinhalte, 2D- oder 3D-Objekte sowie Video- und Audiosequenzen. Zudem ist auch das Ergänzen oder Erfassen von Daten durch die Nutzer möglich. Die AR-Browser lassen sich in andere Anwendungen integrieren, sind in der Regel kostenlos sowie auf vielen Smartphones bereits vorinstalliert. Damit steht nun eine technische Infrastruktur zur Verfügung, die eine kostengünstige Entwicklung mobiler AR-Anwendungen sowie eine massenhafte Nutzung dieser AR-Anwendungen durch Jedermann (Mitarbeiter, Bürger etc.) ermöglicht.

Erst vor kurzer Zeit sind die ersten Konzepte und Implementierung von Content-Plattformen (auch AR-CMS) zur Verwaltung und Bereitstellung von Inhalten für mAR-Anwendungen entstanden. Diese wurden hauptsächlich im kommerziellen Kontext bei Internetagenturen (z.B. marWays[5]) und einige wenige im Kontext von Forschungsprojekten (z.B. KHARMA[6]) entwickelt. Einige sind in einem experimentellen Stadium (z.B. KHARMA) oder werden nicht mehr gepflegt (z.B. NakdReality[7], RADAR[8]).

Fast alle Systeme unterstützen nur einen bestimmten AR-Browser; nur das geschlossene, jedoch nicht mehr aktive System Hoppala[9] (und eingeschränkt RADAR) bot Multi-Kanal-Bereitstellung. Andere Ausgabeformate (KML, GeoRSS und ARML) und Standards (z.B. der OGC: WMS, WFS) zur Nutzung der Inhalte in anderen Kontexten werden kaum unterstützt. Einige Systeme bieten nur das aufwendige manuelle Erfassen einzelner AR-Inhalte (POI) – meist textbasiert oder auf einer Karte – (z.B. Hoppala, RADAR), andere den automatischen Import aus einer Datenbank oder Datei (z.B. Poiz[10], buildAR[11], Poistr[12], VISAR[13]), aber nicht aus Webservices oder anderen Quellen. Die meisten Systeme unterstützen die Veröffentlichung von 3D-Objekten und multimedialen Objekten (Audio, Video etc.), aber nicht deren Erstellung. Von keinem System werden komplexere Datenstrukturen und insb. GIS-Daten, wie Polygone, Linien (z.B. Karten aus Shapefiles) und Rasterdaten unterstützt. Die Systeme bieten in der Regel eine Umkreissuche, aber keine Suche anhand des Kontexts des Benutzers (Schlüsselwörter oder Art der Information). Hinzu kommt eine geringe Nutzerfreundlichkeit, wenn die Systeme mit keiner oder nicht mit einer Web-Oberfläche ausgestattet sind, mit dessen Hilfe komfortabel Inhalte erstellt, integriert, gepflegt und veröffentlicht werden können. Keines der bekannten Systeme unterstützt die Rekombination (Mashup) verschiedener Datenquellen (z.B. Stammdaten aus Datenbank und aktuelle Werte aus Webservice), das Erzeugen von 3D-Objekten „on-the-fly“ aus geografischen Flächen und Linien, das für das Marker-basierte AR erforderliche Tracking (z.B. Marker, QR-Code, Bild) oder Rückkanäle (z.B. Rating, Kommentare).

Aufgrund der Mängel vorhandener Systeme wird in der Praxis derzeit meist noch der für die AR-Anwendung erforderliche Webservice individuell implementiert, womit ein großer Aufwand verbunden ist, den sich nur wenige, große Unternehmen leisten können. Dies behindert den Durchbruch der mAR-Technologie massiv. Die existierenden Systeme belegen durchaus deren Notwendigkeit und Potentiale ihrer spezifischen Funktionen, jedoch sind diese bisher nur rudimentär vorhanden.

[1] www.layar.com
[2] www.wikitude.org
[3] www.junaio.com
[4] argon.gatech.edu
[5] www.mcrumbs.com
[6] www.research.cc.gatech.edu/kharma/
[7] www.lightrod.org bzw. www.nakdreality.com
[8] www.dfki.uni-kl.de/radar/
[9] www.hoppala.eu
[10] www.poiz.nl/de
[11] www.buildar.com
[12] www.poistr.com
[13] www.muzar.org

3 Anwendungsfelder im Hochwassermanagement und der Hydrologie

Aufgrund der verfügbaren einfachen und weit verbreiteten technischen Basis sowie einer Vielzahl potenzieller Anwendungsszenarien für unterschiedliche Einsatzbereiche [MRS11], z.B. in Tourismus [LM12], Bildung [Bi11], Werbung [IBM12] oder Unterhaltung, wird mAR ein großes wirtschaftliches Potenzial zugesprochen [IS10]. Auch im Bereich des Hochwasserschutzes und der Hydrologie lassen sich zahlreiche Anwendungsfälle für mAR finden:

3.1 Hochwassergefahrenkarten

Hochwasser-Gefahrenkarten [EG07] informieren darüber, welche Gefahr von Hochwasser grundsätzlich ausgehen kann, und leisten damit einen wichtigen Beitrag zur Hochwasservorsorge [Mü10]. Sie lassen sich vor Ort über das mobile Endgerät auf einer Karte bzw. als Augmented Map [RED05] (Vogelperspektive) oder als Erweiterte Realität (im Kamerabild des Smartphones) darstellen. Durch den Einsatz von mAR kann der durch die Hochwassergefahrenkarten gezeigte, virtuelle Wasserspiegel direkt in der Realität sichtbar gemacht werden. Damit wird es möglich, Hochwasserszenarien realitätsnah zu erzeugen und damit die Wahrnehmung und Analyse von Gefahren zu erweitern.

Abbildung 2: Hochwassergefahrenkarte in Kartenansicht (links) und in AR-Kameraansicht (rechts)

3.2 Hochwasserwarnung

Damit die aktuell von einem Hochwasser bedrohten Bürger rechtzeitig Maßnahmen zu ihrem Schutz ergreifen können, müssen sie schnell aktuelle Informationen über den derzeitigen und erwarteten Wasserstand bzw. die aktuelle Hochwassergefahr erhalten [HR06]. Aktuelle Hochwassermelde-Pegel lassen sich als Liste, Karte und Erweiterte Realität im mobilen Endgerät darstellen. Durch die Anzeige von Pegeln mit Alarmstufen als mAR lässt sich die aktuell bestehende Gefahr realitätsnaher vermitteln.

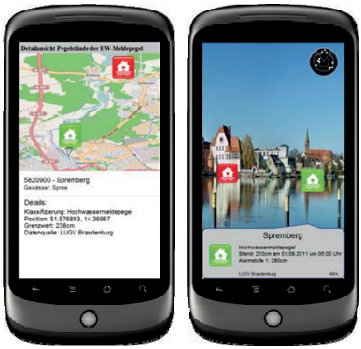


Abbildung 3: Hochwassermeldepegel auf Karte (links) und in Kamera als AR (rechts)

3.3 Gewässerinformationen

Informationen über Gewässer können interessierten Bürgern als Informationsquelle (Name des Gewässers oder des Bauwerks, Einzugsgebietsgröße etc.) dienen. Sie können aber auch hydrologische Fachexperten vor Ort unterstützen; z.B. durch Informationen wie Stationierung, aggregierte, gemessene oder berechnete Wasserstände und Abflüsse (Gewässer-kundliche Hauptzahlen, Bemessungsabflüsse etc.), Ausbauzustand, Pflege-Rhythmus.

Die Abbildung 4 zeigt eine App, die hydrologische Fachdaten für Fachanwender vor Ort (in-situ) visuell zur Verfügung stellt. Dabei handelt es sich um die Darstellung von modellierten Abflussdaten auf einer Karte sowie in der Kamera-Ansicht. Es werden Abflusswerte an mehreren Punkten im Kamerabild angezeigt. Zusätzlich ist es auch möglich, die Karte des betrachteten Flusseinzugsgebiets mit in das Kamerabild einzublenden.

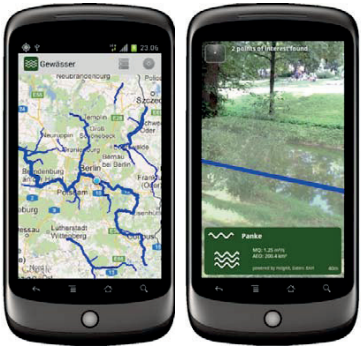


Abbildung 4: Gewässernetz mit Flusseinzugsgebiet als Karte (links) und mit hydrologischen Fachdaten als AR (rechts)

3.4 Bewusstseinsbewahrung (Historische Hochwassermarken)

Historische Hochwassermarken zeigen Überflutungshöhen von historischen Hochwassern an und erinnern damit an diese vergangenen Überschwemmungen [PTK03]. Sie können als Liste, Karte und Erweiterte Realität im mobilen Endgerät dargestellt und ggf. von der interessierten Bevölkerung selbst erfasst (Foto inkl. Metadaten) werden. Dadurch kann die Bevölkerung dauerhaft involviert und das Bewusstsein über die Gefahren von Hochwassern aktiv gehalten werden.

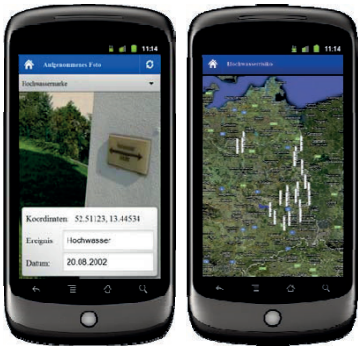


Abbildung 5: Erfassung einer historischen Hochwassermarke (links) und Darstellung auf Karte (rechts)

3.5 Hochwasserlehrpfad

Ein (Natur-) Lehrpfad dient verschiedenen Aufgaben. Dazu gehören u.a. die Umweltbildung, die Förderung der Regionalentwicklung, die Besucherlenkung und die Vermittlung einer spezifischen Thematik, z.B. der Sensibilisierung und Bewusstseinsbildung über Hochwassergefahren [Sz99].

Mit einer mobilen AR-Anwendung können Informationen über die Position der Objekte auf einem Lehrpfad sowie zugehörige Lern-Inhalte im Kamerabild angezeigt werden. Dadurch können die einzelnen Objekte leichter gefunden werden (Orientierung im Gelände, Wegweiser auf dem Pfad) und aktuelle (Zusatz-) Informationen kostengünstig bereitgestellt werden (multimedialer, selbstgesteuerter Lernprozess).

4 Anforderungen aus Hochwassermanagement und Hydrologie

Aus diesen verschiedenen Szenarien aus dem Bereich des Hochwasserschutzes und der Hydrologie lassen sich eine Reihe unterschiedlicher Anforderungen an eine mAR-Informations-Infrastruktur ableiten:

- Es müssen unterschiedliche Informationen aus unterschiedlichen Quellen integriert werden, z.B. GIS für HW-Karten und Web-Services für Pegeldaten, und in verschiedenen Datenformaten beherrscht werden, z.B. 2D-/3D-Objekte für HW-Karten, strukturierte Daten für Pegel oder Bildformate für Hochwassermarken (multi-sourcing).

- Damit diese heterogenen Daten aus verschiedenen Quellen in einem gemeinsamen System genutzt und ggf. wiederverwendet und zu neuen AR-Diensten kombiniert werden können (re-combining), z.B. Hochwassergefahrenkarten zusammen mit Hochwassermeldepegeln, müssen die Input-Daten aufbereitet und - zumindest intern - in ein gemeinsames Format gebracht werden.

- Auch die Datenausgabe ist sehr vielfältig und erfordert unterschiedliche Datenformate für die Bereitstellung (multi-channel). Z.B. sind unterschiedliche Formate für die Darstellung von Daten auf einer Karte und als AR erforderlich, zudem besitzt jeder AR-Browser sein eigenes Datenformat und für die Darstellung spezieller Inhalte sind meist bestimmte Formate erforderlich (z.B. Pegel als strukturierte POI und Gefahrenkarten als 2D-/3D-Objekte etc.). Zusätzlich ist ggf. die Bereitstellung von anwendungsunabhängigen Inhalten in bestimmten (Standard-) Formaten erforderlich.

- Zudem sind unterschiedliche Nutzermengen (wenige bis viele) und Nutzergruppen (Fachleute, Bevölkerung etc.) zu unterstützen.

- Darüber hinaus müssen die Nutzer auch die Möglichkeit haben, mit den virtuellen Objekten zu interagieren, d.h. diese zu ergänzen, zu aktualisieren oder auch eigene neue Objekte zu erstellen (z.B. historische Hochwassermarken erfassen). Es ist also ein Rückkanal zur Informationsbasis bereitzustellen.

Zum gegenwärtigen Zeitpunkt liegt der Fokus der bereits verfügbaren Systeme (siehe oben) nur auf einem einzelnen Teil der Content-Pipeline. Inhalte aus verschiedenen Quellen mit unterschiedlichen Formaten zu integrieren (multi-sourcing), neu zusammenzusetzen (re-combining), plattformübergreifend (cross-platform) für verschiedene Darstellungsformen und AR-Browser bereitzustellen (multi-channel) sowie Interaktion (Rückkanal) mit dem Nutzer zu ermöglichen, ist zurzeit nicht im Fokus. Um die Potenziale der mAR ausschöpfen zu können, muss der Übergang zu ganzheitlichen Systemen gelingen, welche die oben genannten heterogenen Anforderungen umfassend erfüllen.

5 Konzept der mAR-infrastruktur

Zur Realisierung der o.g. Anforderungen (Kapitel 4) bzw. der o.g. mAR-Anwendungen (Kapitel 3) wurde im Projekt MAGUN [Ma12] eine mAR-Informations-Infrastruktur für die Erstellung, Integration, Verwaltung und Bereitstellung von Inhalten für mAR-Anwendungen auf Basis von Open Source-Software (GeoServer etc.) entwickelt. Diese ermöglicht es, vorhandene Daten aus verschiedenen Quellen in unterschiedlichen Formaten (multi-sourcing) zu integrieren und diese wieder für verschiedene AR-Browser und ggf. weitere Dienste über standardisierte Schnittstellen (multi-channel) bereitzustellen. Darüber hinaus ist es möglich, einmal vorhandene Inhalte wiederzuverwenden und zu neuen AR-Diensten zu kombinieren (re-combining) sowie die Interaktion der Nutzer mit den AR-Inhalten zu ermöglichen (Rückkanal). In diesem Abschnitt werden grundlegende Konzepte dieser mAR-Infrastruktur dargestellt.

Kern des Konzepts der Plattform ist die Abbildung aller Datenquellen auf ein einheitliches Modell sowie die Generierung spezieller Output Channel zur Bereitstellung der Daten für die verschiedenen Client-Anwendungen und insbesondere die üblichen AR-Browser-Dienste (siehe Abschnitt 5.1). Damit sich in der mAR-Infrastruktur viele verschiedene mAR-Anwendungen realisieren lassen, werden die anwendungsspezifischen Abbildungen von Quelldaten (bereitgestellt durch Datenquellen) auf ein einheitliches Modell sowie dessen Abbildung auf verschiedene Ausgabeformate in einem sogenannten Projekt erstellt und verwaltet (siehe Abschnitt 5.2).

5.1 Datenintegration und -transformation

Relevante Daten für mAR-Anwendungen, insb. Geodaten, sind häufig bereits verfügbar. Sie liegen aber meist bei unterschiedlichen Datenbereitstellern (Unternehmen, Behörden etc.), in unterschiedlichen Systemen (Datenbanken, Dokumenten-basierten Speichern / NoSQL etc.) in einem eigenen Datenmodell vor und sind dann ggf. über unterschiedliche Schnittstellen (SOAP, RESTful Webservice etc.) zugreifbar. Um diese in verschiedenen Datenquellen verfügbaren Daten erfolgreich in einer neuen mAR-Anwendung zu re-kombinieren, ist ein pragmatischer Ansatz, diese verschiedenen Daten auf ein gemeinsames Modell abzubilden.

Das im Folgenden beschriebene gemeinsame Datenmodell für mAR-Anwendungen wird - inspiriert durch die gängige GIS-Terminologie [Bi10] – im Weiteren „Feature Model“ genannt. Es ermöglicht, die Definition eigener, individueller Typen (ähnlich wie die Definition von Klassen im Objektorientierten Entwurf). Diese

Typen werden Feature Typen genannt. Die Eigenschaften dieser Typen hängen dann vom spezifischen Anwendungsfall ab (wie in jedem Softwareentwicklungsprojekt). Instanzen eines Feature Types werden Features genannt und werden unter Nutzung von Daten aus unterschiedlichen Datenspeichern (data store) erzeugt, die intern durch Datenquellen (data sources) repräsentiert werden.

Eine Datenquelle (data source) ist einen interne Repräsentation und kapselt den (Lese-) Zugriff zu einer der real verfügbaren Datenspeicher (data store), in dem die Daten tatsächlich extern vorgehalten werden. Eine Datenquelle (data source) dient vor allem der einheitlichen, konsistenten Abbildung der von dem Datenspeicher (data store) bereitgestellten Daten in das gemeinsame Datenmodell (Feature Modell). Anschließend können diese Daten genutzt werden, um Instanzen von Feature Types (sog. Features) zu erzeugen. Wie diese Features aus den verfügbaren Daten der Datenquellen erzeugt werden, wird durch eine Abbildung (mapping) beschrieben. Es ist wichtig ist zu beachten, dass die von den Datenquellen angebundenen Daten dabei re-kombiniert werden (können), um Feature-Instanzen zu erzeugen (siehe Abbildung 6).

Ein Attribut einer Datenquelle (data source) kann entweder direkt auf ein Attribut eines Feature Type abgebildet werden oder mehrere verschiedene Attribute können kombiniert werden, um ein einzelnen Ziel-Attribut (eines Feature Type) zu bilden (z.B. können die Attribute „Vorname“ und „Nachname“ zu einem Attribut „Name“ kombiniert werden). Der umgekehrte Fall (Aufspalten eines Attributes einer Datenquelle in mehrere Ziel-Attribute eines Feature Type) ist ebenfalls möglich. Abbildung 6 stellt ebenfalls dar, dass Features aus mehreren verschiedenen Datenquellen kombiniert werden können (z.B. Pegeldaten aus Pegel-Online und Pegeldaten aus WISKI). Die Abbildung wie auch die Konfiguration der Datenquellen müssen über das Projekt erstellt werden.

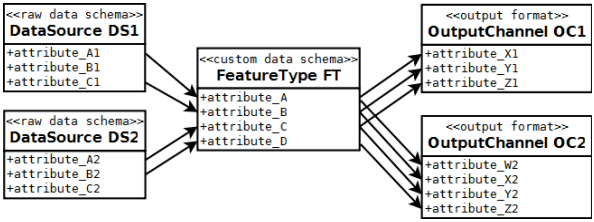


Abbildung 6: Rekombination und Publizierung von Daten aus verschiedenen Datenquellen in verschiedene Ausgabekanäle

Sobald ein Feature (die Instanz eines Feature Type) erzeugt wurde (unter Einbeziehung von Daten verschiedener Datenquellen), kann dieses publiziert werden, wobei unterschiedliche Formate und Protokolle genutzt werden können. Diese werden durch sog. Ausgabekanäle (output channel) gekapselt. D.h. ein Ausgabekanal repräsentiert die Ausgabe eines Features in einem bestimmten Format oder Protokoll. Abhängig von dem gekapselten Format erwartet der Ausgabekanal, dass das Feature über bestimmte Attribute verfügt. Daher ist eine weitere Abbildung (mapping) erforderlich: vom Projekt-spezifischen Feature-Modell (basierend auf dem Feature Type) zum vom Ausgabekanal erwarteten Format. Die Menge (Liste) der Ausgabekanäle wie auch die Abbildungen (mappings) der Attribute des Feature Type auf die Attribute des Ausgabekanals werden ebenfalls innerhalb des Projekts spezifiziert. Auch hier kann es vorkommen, dass ein Attribut des Feature-Type auf mehrere Attribute eines Ausgabekanals abgebildet werden muss oder mehrere Attribute des Feature-Type zu einem Attribut des Ausgabekanals aggregiert werden.

5.2 Projekt zentriertes Mashup-System

Ein weiteres Grundkonzept der mAR-Infrastruktur ist das Projekt-zentrierte Mashup-System. D.h. ein Projekt verkörpert ein einzelnes mAR-Szenario zur Realisierung einer einzelnen mAR-Anwendung auf der Basis der bereits vorhandenen, an das System angebundenen Datenspeicher. Ein Projekt beschreibt, welche Daten und wie diese Daten aus verschiedenen Quellen re-kombiniert (Mashup) sowie unter Nutzung verschiedener Ausgabeformate und -protokolle publiziert werden sollen. Für jede einzelne mAR-Anwendung soll daher ein

eigenes Projekt erzeugt und verwaltet werden. Auf diese Weise lassen sich in der mAR-Infrastruktur viele verschiedene mAR-Anwendungen realisieren.

Ein Projekt beschreibt daher die für eine Anwendung benutzten Datenquellen (data sources), Feature Typen (feature types) und Ausgabekanäle (output channel) sowie auch die Abbildungen (mappings), um die Daten der Datenquellen zu re-kombinieren (Abbildung der Daten der Datenquellen auf die Attribute des Feature Typs) und die erzeugten Feature Typen auf die Ausgabekanal-spezifischen Formate abzubilden (Abbildung des Feature Typs auf die Attribute des Ausgabekanals). Darüber hinaus, damit mehrere unterschiedliche Projekte innerhalb einer mAR-Infrastruktur verwaltet werden können, beinhaltet jedes Projekt eine eindeutige Identifikation sowie weitere beschreibende Metadaten (Name etc.). Diese Projektdefinitionen werden in einem Projekt-Katalog verwaltet. Clients können diesen Katalog durchsuchen, um durch die Infrastruktur bereitgestellte Inhalte zu finden. Diese Daten können von mAR-Anwendungen aber auch von anderen Projekten innerhalb der Infrastruktur als Datenquelle zum Mashup von Daten genutzt werden. Diese Möglichkeit erlaubt somit die Kaskadierung von Projekten.

6. Implementierung

Im Projekt MAGUN wurde zur Umsetzung dieses Konzepts die „HolgAR Content-Plattform“ implementiert.

6.1 holgAR-Architektur

Die Infrastruktur besteht aus vier Server-Anwendungen: Einem auf dem GeoServer basierendem WMS/TMS Server, einem Datenimport-Server, einem Konfigurations- und Administrationsserver sowie dem Publikationsserver.

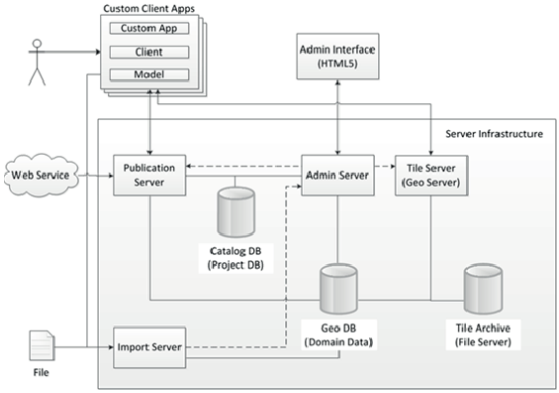


Abbildung 7: holgAR-Architektur

Der Publikationsserver ist die zentrale Komponente der Infrastruktur. Er bindet Daten aus verschiedenen Datenquellen ein, re-kombiniert diese zu den Anwendungsdaten und stellt diese Anwendungsdaten über verschiedene Formate und Protokolle bereit.

Hierzu liest der Publikationsserver den Projektkatalog aus, der vom Konfigurationsserver bereitgestellt wird. Der Projektkatalog beschreibt die Daten aller in der Infrastruktur vorhandenen Projekte und wird genutzt, um die konkreten Datenquellen, Feature-Typen und Ausgabekanäle zu instanziieren. Sobald eine Änderung an den

Projektdaten vorliegt, wird der Publikationsserver vom Konfigurationsserver informiert und kann seine Datenstrukturen neu laden.

Trifft eine vom Client initiierte HTTP-Anfrage beim Publikationsserver ein, wird diese in ein protokollunabhängiges Anfrageobjekt konvertiert und anschließend an die für das Projekt konfigurierten Datenquellen weitergeleitet. Die Datenquellen liefern die passenden Daten zurück. Sie nutzen hierzu entweder persistente Speichermechanismen, Web-Services oder auch in-memory Datenstrukturen. Im Rahmen des Projekts wurden Web-Services, Geo-Datenbanken (PostGIS), Dateischnittstellen zum System WISKI und auch der Pegeldatendienst „Pegel-Online“ als Datenquelle implementiert. Das beschriebene Modell setzt voraus, dass der Server „stateless“ implementiert ist. Die von den Datenquellen zurückgelieferten Daten existieren nur innerhalb einer Anfrage. Eintreffende Anfragen werden zudem asynchron behandelt. Sie werden in eine Queue gelegt und dann von einem Thread-Pool abgearbeitet. Dies bedeutet, dass Datenquellen von mehreren Threads parallel genutzt werden können. Da es sich jedoch um Read-Only-Datenquellen handelt, stellt dies implementierungstechnisch keine größere Herausforderung dar.

Die Datenquellen erzeugen Feature-Instanzen, welche der Client-Anfrage entsprechen. Die Client-Anfrage besteht aus Paginations-Informationen und einer Liste von Prädikaten. Die Prädikate können sich entweder auf Datenattribute (zum Beispiel alle Features vom Typ „See“) oder auf Geometrien (z.B. alle Features innerhalb eines spezifizierten Radius) beziehen. Wie die Datenquellen die Prädikate genau umsetzen, ist implementierungsabhängig. Sobald alle Datenquellen die Features zurückgeliefert haben, werden diese re-kombiniert und an den Client gesendet.

In welcher Form die Daten an den Client gesendet werden, hängt vom gewählten Ausgabekanal ab. Die Auswahl wird zur Anfragezeit vom Client getroffen. Hierzu wird ein spezifischer Pfad innerhalb der URL verwendet. So zeigt z.B. der Pfad /layar/flood auf den Layar-Ausgabekanel des Projekts „flood“.

Jedes Feature-Objekt enthält Geometrie-Informationen. Diese bestehen entweder aus einer einfachen Punktgeometrie oder komplexeren Linien und Polygonen. Der Client benötigt diese Geometrien, um diese entweder auf eine Karte oder in der AR-Ansicht zu visualisieren. Kleine Geometrien, wie zum Beispiel kurze Linien oder Polygone mit wenigen Kanten, können zusammen mit den Sachdaten direkt an den Client gesendet werden. Hierzu können zum Beispiel die Formate WKT oder GeoJSON genutzt werden. Der Client kann diese Geometrien anschließend direkt auf dem Gerät rendern. Diese Möglichkeit kann jedoch nicht immer genutzt werden. Zum einen kann es Limitierungen der Rendering-Engine auf dem Client geben. Dies ist bspw. bei den gängigen AR-Browsern der Fall, da diese in der Regel keine Linien oder Polygone darstellen können. Zum anderen können große Geometrien nicht mit akzeptabler Performance auf mobilen Geräten dargestellt werden. So bestehen einige Geometrien aus dem Bereich der Hochwassergefahrenkarten aus über 2,5 Millionen Kanten und konnten selbst auf aktuellsten Smartphones nicht mehr in akzeptabler Geschwindigkeit dargestellt werden. Das Performanz-Problem kann umgangen werden indem die Polygone in kleinere Teile zerschnitten werden. Dies ist als Tesselation bekannt. Dies behebt jedoch nicht die Problematik der limitierten Rendering-Engines. Da es sich bei den meisten AR-Browsern um Closed-Source-Produkte handelt und der Markt noch immer fragmentiert ist, hat sich als einzige praktikable Lösung die Verwendung einer Serverseitigen Rendering-Engine herausgestellt.

Eine gängige Lösung für dieses Problem ist die Verwendung eines Tile-Servers. Ein Tile-Server stellt vorgenerierte Bilder zur Darstellung auf dem Client bereit. Dies behebt das Performanz-Problem und erlaubt ebenfalls die Darstellung von komplexen Geometrien in der AR-Ansicht, da nahezu alle AR-Browser das Darstellen von Bildern erlauben. Es gibt bereits mehrere Open-Source-Lösungen, die das Generieren von Tiles ermöglichen.

Die implementierte holgAR-Infrastruktur nutzt GeoServer und GeoWebCache zum Generieren und Ausliefern der Tiles. Der GeoServer erlaubt es, Tiles aus verschiedenen Datenquellen zu generieren. Die so erzeugten Tiles werden für die Darstellung in AR nochmals transformiert und über den Publikationsserver ausgeliefert. Die genaue Art der Transformation hängt vom gewählten Ausgabekanal ab.

Der Datenimport-Server erlaubt es mobilen Clients, Geo-Daten über eine REST-Schnittstelle hochzuladen. Diese Daten werden in einer PostGIS-Datenbank gespeichert und über eine Datenquelle im Publikationsserver ausgeliefert.

6.2 holgAR-Datenintegration und -transformation

Abbildung 8: holgAR-Infrastruktur - Schnittstellen für Datenquellen und Datenausgabe

Innerhalb der mAR-Infrastruktur erfolgt die Anbindung konkreter Datenquellen (data sources), deren Abbildung auf Geoobjekte („Features“) und deren Transformation in konkrete Ausgabekanäle (output channels). Implementationen von Datenquellen lassen sich über die Schnittstelle „DataSource“ beitragen. Eine Datenquelle gibt für einen parametrisierten Aufruf eine Menge von DataEntity-Objekten zurück, die einem Datensatz in einer Tabellenzeile (row) entsprechen. Zur Beschreibung der Datenstruktur muss eine Datenquelle ein DataSchema-Objekt zurückgeben. Über ein „Source Mapping“ werden aus den DataEntity-Objekten dann Geoobjekte („Features“), die durch einen Geoobjekttyp („Feature Type“) beschrieben werden. Dieses Quellen-Mapping wird in eine Tabelle zwischengespeichert, die die Struktur des Feature Types aufweist. Somit lässt sich diese Tabelle auch als Feature Type im GeoServer verwenden. Die Ausgabe der Daten erfolgt in verschiedenen Formaten und wird über Ausgabekanäle („Output Channel“) gesteuert. Diese sind ähnlich zu Datenquellen aufgebaut, mit dem Unterschied, dass sie Geoobjekte („Features“) verarbeiten. Die Transformation erfolgt mittels eines Ausgabe-Mappings („Output Mapping“).

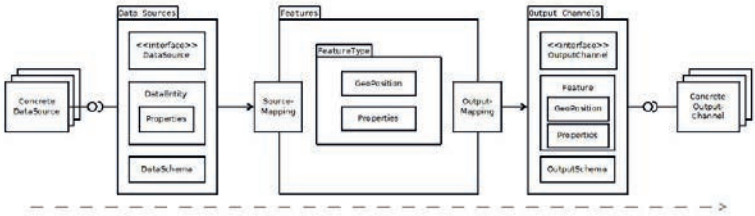


Abbildung 9: holgAR-Infrastruktur – Datenintegration und -transformation

Häufig liegen die in Datenspeichern vorhandenen Daten in Standardformaten vor. Insbesondere Datenspeicher mit georeferenzierten Daten bieten häufig Formate wie GML, KML, GPX, GeoJSON oder GeoRSS. Diesen Formaten ist gemeinsam, dass sie sehr einfach aufgebaut sind. Oftmals enthalten sie nur Titel, Georeferenz und Kurzbeschreibung sowie einen Verweis auf eine speziellere Darstellung des Datensatzes, die dann meist in einem anderen Format vorliegt. Die Abbildung dieser Formate in ein gemeinsames Modell (Feature Type) ist somit relativ einfach. Solche simplen Datenobjekte (POI’s) lassen sich dann auch relativ einfach in diverse Ausgabekanäle transformieren und auf in den Client-Anwendungen (Apps) in typischen Darstellungsformen, d.h. als Liste, als Punktobjekt auf einer Karte oder als POI auf dem Display der Kamera darstellen. Durch Auswahl eines bestimmten Objekts in einer dieser Client-Darstellungen werden mit Hilfe einer vorhandenen Verlinkung die spezielleren bzw. Detail-Informationen aufgerufen und in einer spezielleren Darstellung (meist im Web-Browser) dargestellt.

6.3 holgAR-Apps

Auf Basis dieser Plattform wurden bereits mehrere mAR-Anwendungen für den Hochwasserschutz und hydrologische Fachinformationen entwickelt, z.B. zur Darstellung von Gewässernetzen und Gewässereinzugsgebieten sowie Hochwassergefahrenkarten und Hochwassermeldepegeln, auf einer Karte (Augmented Map) und im Kamerabild (Augmented Reality) von Smartphones bzw. Tablets (siehe Kapitel 3).

7 Zusammenfassung und Ausblick – Grenzen von mAR-Browsern

Mobile Augmented Reality ermöglicht die Verschmelzung und gemeinsame Wahrnehmung von realen und digitalen Informationen im Ortskontext. Mit der Verfügbarkeit von Smartphones und AR-Browser steht auf der Clientseite die erforderliche Hardware und Software zur Verfügung, um eine kostengünstige Entwicklung mobiler AR-Anwendungen und die massenhafte Nutzung dieser AR-Anwendungen zu ermöglichen. Mit der in diesem Beitrag vorgestellten AR Content-Plattform wird auf der Serverseite eine wichtige Lücke geschlossen, da es nun möglich ist, effizient Inhalte für die AR-Anwendungen auch durch Nicht-IT-Experten bereitzustellen. Derzeit wird diese AR Content-Plattform im Rahmen von AR-Anwendungen im Hochwasserschutz erprobt und evaluiert.

mAR besitzt ein großes Potenzial für den Umweltschutz und insbesondere den Hochwasserschutz. Aufgrund der massenhaften Verbreitung AR-fähiger mobiler Geräte wird es nun möglich, Umweltinformationen in die Breite zu bringen. D.h. Informationen sind nicht nur für wenige Fachexperten, sondern für eine riesige Anzahl an Nutzern („normale“ Bürger) verfügbar, z.B. Informationen über Hochwassergefahren. mAR ist eine neuartige Nutzerschnittstelle, die im Ortskontext eine unmittelbare neue mediale Erfahrung schafft und bei der Orientierung in der Realität hilft. Dies ermöglicht einerseits eine neuartige Wahrnehmung des Ortes durch die Anreichung mit Informationen aus Vergangenheit, Gegenwart oder Zukunft (welche Schäden wurden durch ein vergangenes Hochwasser verursacht). Anderseits ermöglicht dies aber auch eine bessere Analyse und Interpretation und Analyse von digitalen Daten vor Ort und somit eine bessere Entscheidungsfindung (welche Gefahr geht von Hochwasser für mein Haus aus).

Die Entwicklung von Hard- und Software für mAR in Endkunden-Segment wird in den kommenden Jahren rasant voranschreiten. Ein Bespiel für zukünftige Entwicklungen ist die von Google angekündigte Datenbrille (Project Glass). Weiterer Forschungsbedarf in Hinblick auf eine Weiterentwicklung der AR Content-Plattform besteht u.a. in der Skalierbarkeit der Plattform hinsichtlich großer Datenvolumina und Nutzerzahlen sowie einer situativen und aktiven Informationsbereitstellung (information push).

8 Danksagung

Das Projekt MAGUN wurde aus Mitteln des Europäischen Fonds für regionale Entwicklung (EFRE) sowie des Instituts für angewandte Forschung (IfaF) Berlin gefördert. Die Autoren danken den Mittelgebern für die Unterstützung sowie allen MAGUN-Projektpartnern für die fruchtbare Zusammenarbeit.

Literaturverzeichnis

[Az97] Azuma, Ronald T. A.: A Survey of Augmented Reality. In: Presence - Teleoperators and Virtual Environments. Vol. 6, No. 4 (August 1997), 1997; S. 355-385.

[Be11] Belic, Dusan: Juniper Research: Revenue from augmented reality apps and services to approach $1.5 billion by 2015, http://www.intomobile.com/2011/02/06/juniper-research-augmented-reality-revenue/, 2011.

[Bi10] Bill, Ralf: Foundations of Geographical Information Systems. 5. Eds., Berlin: Wichmann, 2010.

[Bi11] Bischoff, Andreas: Dienste für Smartphones an Universitäten - ein plattformunabhängiges Augmented Reality Campus-Informationssystem für iPhone und Android-Smartphones. In (Jörg Roth Martin Werner Hrsg.): 8. GI/KuVS-Fachgespräch Ortsbezogene Anwendungen und Dienste, Berlin: Logos Verlag, 2011, 127-135.

[Co04] Coelho, Alexandre Hering: Erweiterte Realität zur Visualisierung simulierter Hochwasserereignisse. Dissertation, Universität Fridericiana zu Karlsruhe, 2004.

[EG07] Europäische Gemeinschaft: Richtlinie 2007/60/EG des Europäischen Parlaments und des Rates vom 23. Oktober 2007 über die Bewertung und das Management von Hochwasserrisiken. ABl. L 288 vom 06.11.2007, 2007.
[FMH97] Feiner, S.; MacIntyre, B.; Höllerer, T.; Webster, T. (1997): A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. In Proc. of the ISWC '97 (First IEEE Int. Symp. on Wearable Computers), S. 208–217.
[FS12] Fuchs-Kittowski, Frank; Simroth, Stefan: Mobile Erweiterte Realität für Jedermann. In: arcAKTUELL: Erde 2.0 - GIS und Natur, S. 46-47.
[FWT12] Fuchs-Kittowski, Frank; Wilske, Fabian; Trosien, Frank: Mobile erweiterte Realität im Hochwasserschutz. In: Umweltinformationssysteme - Informationsgewinnung und Datenaufbereitung für maritime Informationssysteme, Dessau: Umweltbundesamt, 2012, S. 45-54.
[HFT99] Höllerer, T.; Feiner, S.; Terauchi, T.; Rashid, G.; Hallaway, D.: Exploring MARS - Developing In-door and Outdoor User Interfaces to a Mobile Augmented Reality System, Computers and Graphics, 23(6), Elsevier Publishers, 1999; S. 779-785.
[HR06] Hornemann, Corinna; Rechenberg, Jörg: Was Sie über den vorsorgenden Hochwasserschutz wissen sollten. Dessau: Umweltbundesamt, 2006.
[IBM12] IBM Research: Augmented reality makes shopping more personal - New mobile application from IBM Research helps both consumers and retailers. http://www.research.ibm.com/articles/augmented-reality.shtml, 2012.
[IS10] Inoue, K.; Sato, R.: Mobile Augmented Reality Business Models. In: Mobile Augmented Reality Summit, 17.2.2010 in Barcelona, S. 1-2, www.perey.com/MobileARSummit/Tonchidot-MobileARBusiness-Models.pdf.
[LM12] Linaza, M.T.; Marimon, D.: Evaluation of Mobile Augmented Reality Applications for Tourism. In (Fuchs, M. et al. Hrsg.): Information and Communication Technologies in Tourism, Wien: Springer-Verlag, 2012, S. 260-271.
[Ma12] MAGUN: Mobile Anwendungen auf Basis von Geoinformationen in einer In-situ Informationsinfrastruktur im Umwelt- und Navigationsbereich, URL: http://www.magun-projekt.de/, 2012.
[MRS11] Mehler-Bicher, A.; Reiß, M.; Steiger, L.: Augmented Reality — Theorie und Praxis. München, 2011.
[Mü10] Müller, Uwe: Hochwasserrisikomanagement - Theorie und Praxis. Vieweg & Teubner, 2010.
[PTK03] Petrow, Theresia; Thieken, Annegret; Kreibich, Heidi; Merz, Bruno: Vorsorgende Maßnahmen zur Schadensminderung. In: Hochwasservorsorge in Deutschland - Lernen aus der Katastrophe 2002 im Elbegebiet. Schriftenreihe des DKKV 29, Bonn: Deutsches Komitee für Katastrophenvorsorge e.V., 2003.
[RED05] Reitmayr, Gerhard; Eade, Ethan; Drummond, Tom: Localisation and Interaction for Augmented Maps. In: Proc. IEEE ISMAR'05, October 5-8, 2005, Vienna, Austria. URL: http://mi.eng.cam.ac.uk/~gr281/augmentedmaps.html, 2005.
[Sch11] Schall, G.: Mobile Augmented Reality for Human Scale Interaction with Geospatial Models. Dissertation, TU Graz. February 2011.
[Sz99] Szekeres, P.: Naturlehrpfade. 3. Aufl., Marburg, Institut für Ökologie, URL: http://www.projektwerkstatt.de/download/texte_cd/reader/lehrpfade.pdf, 1999.
[Tö10] Tönnis, M.: Augmented Reality — Einblicke in die Erweiterte Realität. Berlin/Heidelberg, 2010.
[WR11] Werner, Martin; Roth, Jörg (Hrsg.): 8. GI/KuVS-Fachgespräch - Ortsbezogene Anwendungen und Dienste, Berlin: Logos-Verlag, 2011.

A Tool for Visualizing and Editing Multiple Parallel Tracks of Time Series Data from Sensor Logs

Marco Maier, Florian Dorfmeister, Mirco Schönfeld, Moritz Kessel

Mobile and Distributed Systems Group
Ludwig-Maximilians-Universität München
Oettingenstraße 67
80538 München
{forename}.{surname}@ifi.lmu.de

Abstract: In the last couple of years, the number of smartphone users has proliferated. As smartphones contain a lot of different sensors, research interest in algorithms combining data of several sensors to derive meaningful information about a user's context, e.g. her location, is increasing. Ideas for reasonable algorithms often can be found by looking at examples of raw (or somehow pre-processed) sensor data. Still, the research community lacks a simple means to intuitively work with several parallel tracks of time series data. We present a tool which can be used to visualize and edit time dependent data from various sensors along with reference data like audio or video recordings. It is complemented by an optional application for logging all kinds of sensor data on Android devices. The tool has already been used to analyze a large data set containing hundreds of walking traces, intended to improve pedestrian dead reckoning algorithms.

1 Introduction

With the increasing spread of smartphones, using their wide range of integrated sensors to derive context information such as a user's location has become a major research topic. A typical smartphone contains sensors like accelerometers, magnetic field sensors, gyroscopes, GPS receivers, or light sensors. Additionally, various other data sources can be used, such as a list of surrounding WiFi access points (and the respective received signal strengths) or a list of nearby bluetooth devices. While most of these data sources can be of use even when regarded independently, a lot of interesting applications require fusioning several types of sensor information.

For example, research in indoor positioning techniques has resulted in pedestrian dead reckoning techniques based on step detection (using accelerometers), heading information (using gyroscopes and compasses) and activity-based map matching (using accelerometers). [GIK10] Step detection and pedestrian dead reckoning in general are prime examples of research areas in which (ideas for) algorithms can often be developed or improved by simply looking at visualizations of sensor data. For example, a plot of accelerometer values of a smartphone carried by a walking person shows a very characteristic pattern. Although it is not advisable to purely rely on intuition when looking for new algorithms, visualizations of raw or pre-processed sensor data can be a great help to examine outliers and edge cases.

During evaluation of various step detection and dead reckoning algorithms on a new data set we collected, the need arose to visualize data logs of several sensors all at once and all of them synchronized in order to further examine spots where the algorithms did not perform as expected. We often ended up with a multitude of windows showing various plots of various sensors as depicted in figure 1.

We found this to be overly complicated and confusing. Due to the lack of existing software solving this problem, we developed a tool which allows to display and edit multiple parallel tracks of time-dependent sensor data. The concept for the user interface was inspired by multi-track audio editing software like Audacity [aud].

In our tool, there are several pre-defined track types to appropriately visualize different kinds of data, e.g. the *static plot track*, the *dynamic plot track*, the *map track* or the *audio track* and *video track*. Static and dynamic plot tracks can be used to represent data in various charts. They have several parameters to define how the input data format should be interpreted and what kind of plot should be used. The map track can be used to show a map as well as

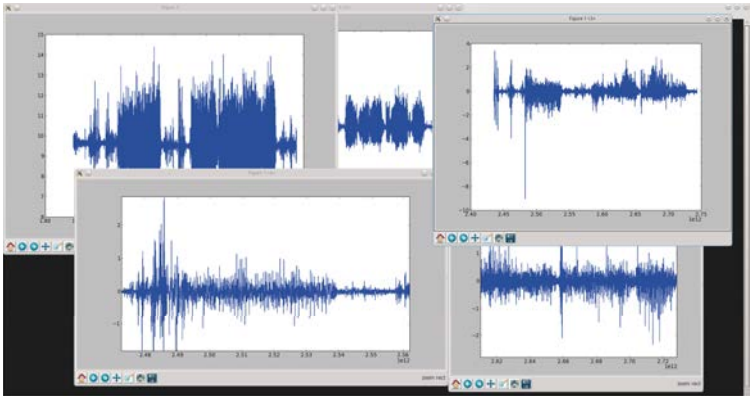


Figure 1: Screenshot of the typical desktop contents when working with many plots and charts, which led to the creation of the MTDT.

markers for positions of entities which change over time. The map track also is suitable as a reference track for the development of positioning algorithms. Whenever other types of context should be recognized, audio or video files can serve as convenient reference tracks, therefore our tool includes the audio track and video track types. It is easily possible to add one's own track types in case special visualizations are required.

We used our tool to analyze a newly collected data set consisting of several hundreds of walking traces. To create the traces, we asked people to take a walk with us on a pre-defined route and gave them three Android smartphones which were carried in different positions. Afterwards, we analyzed the behaviour of different dead reckoning algorithms. Our tool made it easy to synchronize the data of the three smartphones and to compare the different sensor data traces. Our logging application also recorded an audio track from which we could infer some reference information.

For situations when there are several data sets originating from different devices but observing the same situation (like the three smartphones in the example above) our tool provides utility features such as an automatic synchronization of the different data sets.

Although our tool is quite flexible with regard to input data sources, we additionally provide an Android application which can be used to log all kinds of sensor data and the output of which is perfectly suited to be used with our visualization tool.

In the following, we first list the requirements we identified for the tool (section 2). In section 3, we present the concept of our tool. After that, we give a brief overview of the corresponding Android application (section 4). In section 5, we explain the synchronization algorithm we use in the tool and finally, we conclude by looking at future work (section 6).

2 Requirements

We identified several requirements which a tool intended for the above mentioned purposes has to meet.

Appropriate visualizations As explained in section 1, there is a multitude of types of sensors and therefore types of sensor data which might be interesting to visualize and analyze within the tool. A lot of types of data

can often be represented by plots like line charts or scatter plots. For example, accelerometer logs can easily be visualized by three line charts, each showing either the x, y or z component of the recorded data.

In contrast, magnetometer logs are not as easy to interpret when shown in a line chart but might be better represented by a virtual compass showing the current heading which can be inferred from the recorded data.

Another example is information about entity positions. Location most often is best visualized on a map with markers for mobile devices/users and points of interest.

Further ideas include graphical representations like histograms or even simple tables to show precise values of the recorded or pre-processed data.

Time-dependent visualizations In most cases, the recorded data is time-dependent. A visualization tool should provide means to advance and go back in time and skip parts of the recorded time span. Visualizations should either adapt to the currently selected point in time or (in case the representation already shows the data in relation to time) at least indicate which part of the visualization shows the values of the current point in time.

Simultaneous visualization of several streams of sensor data In order to analyze several streams of sensor data in parallel, some kind of simultaneous visualization is required. The typical work flow to create plots sequentially and then display them in multiple desktop windows is cumbersome and does not allow to conveniently get an overview of the data. An easier and especially more compact way to visualize several streams in parallel is necessary.

Pre-processing / filtering of input data We developed an additional sensor data logging application for Android smartphones (see section 4) which is perfectly suited to work in conjunction with our tool. However, a data analyzation/visualization tool should provide a reasonably generic way to import data, so that other tools and data sources can be used. This might make it necessary to pre-process or filter the input data.

For example, one might need to adjust the timestamp format (e.g. converting from microseconds to seconds) or to filter out specific entries of the data set which should not be included in the visualization. The tool should provide flexible means to perform these operations.

Automatic synchronization of several data sets Oftentimes, the sensor data logs of an experiment are not created on a single device but on several devices which observe the situation simultaneously. As a consequence, several data sets might exist which probably are not perfectly in sync (due to inaccurate device clocks, etc.). Therefore, the tool should not only provide the possibility to include several data sets in the same visualization, but also to synchronize them. Ideally, the latter is done automatically.

Comparison with reference data Keeping a detailed record of what has happenend during an experiment is a laborous task and often, many things are omitted which in retrospect are found to be of value but then are not available. Therefore, creating an audio or video recording can lead to a valuable basic truth data source. A useful tool should allow to view such reference recordings in parallel and synchronized to all the other visualizations.

Edit and export functionality In addition to visualization, it would be useful to be able to edit and then export the altered data into a new file. A very common task is to extract only a part of the data set into a new file for further processing. Furthermore, it is desirable to be able to tag specific parts of the data. These tags can not only be helpful to the user, but could also be used to create training and test data files which can be used with machine learning techniques.

Extensibility A non-functional but very important requirement is the ability to easily extend the tool in order to add new visualization types which fit one's own specific scenario. Adding new visualizations and integration of various data sources should be as easy as possible.

3 Multi Track Data Tool

In order to meet the aforementioned requirements, we developed a new tool which we call *Multi Track Data Tool (MTDT)*. The general idea is to allow for simultaneous visualization of several streams of sensor data in parallel tracks, similar to audio editing software like Audacity.

There is a global *transport control* which is used to select a point in time or play back the sensor recordings. Again, the concept is similar to audio editing software.

3.1 Main elements

There are two main elements which are used in the tool: *Data sets* and *tracks*. In accordance with similar tools from the audio or video editing area, we call a specific arrangement of data sets and tracks a *project*.

A data set is a single file containing sensor data logs in a *comma-separated values (CSV)* format. Optionally, an audio or video file can be named as a reference file. It is possible to include more than one data set in a project. The format of a data set is defined by a *data set definition (DD)* and *line format definitions (LD)*. See section 3.3 for a more detailed explanation.

A track is a specific visualization of the data set or parts of it. There are several pre-defined track types (see section 3.4). However, a core concept of the MTDT is to allow for easily extending the range of available track types. Track implementations only have to follow a simple interface (see section 3.3).

In general, tracks can fall into two categories, namely *time-dependent* and *time-independent*. Of course, any kind of track usually has some kind of time dependence. However, there are some visualizations which are generated once and after that only show a bar or marker to indicate the current point in time, whilst other visualizations change completely in correlation to the current point in time.

In a project, one can add as many tracks of any kind as desired. In combination, the tracks provide a compact and perfectly customized view onto the data sets.

3.2 Technical details

The MTDT is written in the Python programming language [pyt]. It uses the following libraries for its main features:

- wxPython [wxp] as a cross-plattform GUI toolkit
- matplotlib [mat] for generating the various charts
- pygame [pyg] for audio and video support
- mapnik [map] for map generation

Although Python is an interpreted, high level language, we found it to be perfectly suited as the main language for our tool. Due to its simple syntax and its "There should be one - and preferably only one - obvious way to do it" concept[1], it is very easy for any mildly advanced programmer to add functionality to the tool. Nevertheless, it also allows for more complicated calculations because Python has two great scientific libraries, namely NumPy [num] and SciPy [sci]. Other performance-critical features can either be prepared in an external pre-processing step or could even be included as a Python extension in C.

[1] http://www.python.org/dev/peps/pep-0020/

3.3 Interfaces

Due to the diversity of studies and experiments in which our tool might be of use, it is important that it can be easily adapted to the given scenario. This is achieved by a flexible configuration system for data set formats and simple interfaces which can be used to add new data set and track types.

Data sets A data set typically is a CSV file in which each line represents a sensor data reading. Data sets can contain records of different sensors, each identified by a unique id. Each record must at least contain a *timestamp* value. To allow for different formats of data set files, one has to specify a data set definition. Besides common parameters for CSV files like column delimiter and quote characters, it describes

- which column contains the timestamps
- whether there is more than one sensor type and if this is the case, which column contains the identifiers
- the different line format definitions for each sensor type

Line format definitions contain

- the name of each column of the record
- the data type (e.g. integer or floating point number) of each column

Although CSV files are the only supported data set type at the moment, one could easily add other data set types which are backed by other data sources, e.g. a database. Data sets have to provide the following main methods:

- `getTimestamps` to get a list of all the timestamps in the data set
- `getMinTimestamp` to get the smallest timestamp in the data set
- `getMaxTimestamp` to get the highest timestamp in the data set
- `normalizeTimestamps` to normalize the timestamps so that the smallest one is 0; optionally timestamps can be multiplied by a given factor (e.g. to convert from microseconds to milliseconds)
- `getData` to load and return the records of the complete data set or of a specific sensor type

Tracks There a several pre-defined track types (see section 3.4). However, it is easy to add new track types. A track has to be a subclass of wxPython's `wx.Panel`. Thereby, it can automatically be integrated in the main window's tracks panel while being able to draw any kind of visualization within its own canvas.

To be in sync with the globally selected point in time (and thus all other tracks and reference tracks), a track has to provide one method: `visualizeForTime(timestamp)`. When invoked, the track has to update its visualization to represent the given point in time. As explained in section 3.1, some tracks might only display a marker or progress bar to indicate the current time whilst others might change the drawn image completely whenever the time changes.

3.4 Track types

While analyzing various data sets from our studies and experiments, we created several track types which are now included in the MTDT by default. These track types should be a good starting point for data analyses in the area of (indoor) positioning and context/activity recognition.

Static plot track The static plot track is a time-independent visualization. It offers various kinds of plots and charts (backend by matplotlib) and usually visualizes all the records of a specific sensor at once. The current point in time is only indicated by a vertical bar which traverses the x axis.

As an example, the static plot track can be used to show the values of an accelerometer over time (i.e. drawing $< timestamp, accelerationValue >$ tuples in a line chart or a scatter plot).

Dynamic plot track Similar to the static plot track, the dynamic plot track offers various plot and chart types. However, the dynamic plot track is time-dependent, i.e. its contents changes completely when the current point in time changes. This is due to the fact that it only visualizes a sub-interval of the recorded time span. One can define the length of the interval by two variables δ_1, δ_2. The plot then would be generated for the interval

$$[currentTime - \delta_1, currentTime + \delta_2]$$

Oftentimes, one sets $\delta_1 = \delta_2$. An example use case is to generate a histogram of the data which surrounds the current point in time.

Map track Especially in the area of positioning algorithms and location based services, visualizing positions of entities on a map is required. This can be achieved with the help of the map track. It displays a map together with an arbitrary number of markers. The markers' positions change with respect to the current point in time.

An example use case would be to visualize the position of a pedestrian as tracked by a reliable positioning system, and in a second map track, show the position of the pedestrian as calculated by a dead reckoning algorithm based on data of inertial measurement units. To complete the use case, one might add a static plot track, showing the distance of calculated and actual position over time.

Compass track The compass track is a time-dependent track, which displays a virtual compass. For the current point in time, it simply shows the direction a device is facing as indicated by the recorded magnetometer values.

Audio and video track The audio and video track types are primarily used as reference tracks. Often, it is more convenient to create an audio or video recording of an experiment in contrast to writing everything down by hand. In case an audio or video recording is available, it can be included in the project and played synchronously to the other tracks, thus providing very good reference information.

3.5 Example project

In figure 2, one can see a simple example of a MTDT project. It was used while analyzing pedestrian traces we collected to examine different pedestrian dead reckoning algorithms.

In this case, three data sets (originating from three different smartphones used simultaneously) are imported. As a reference, a video track is included, showing what happened in the experiment. On a map track, the current position of the pedestrian is shown on a floor plan of our university building. Furthermore, there are three static plot tracks, showing the acceleration values in each axis x, y and z.

4 Android Sensor Data Logger

To simplify our workflow of creating and analyzing sensor data logs, we additionally created an Android [and] application which logs sensor readings from the device. Currently, the following data is logged:

- accelerometer

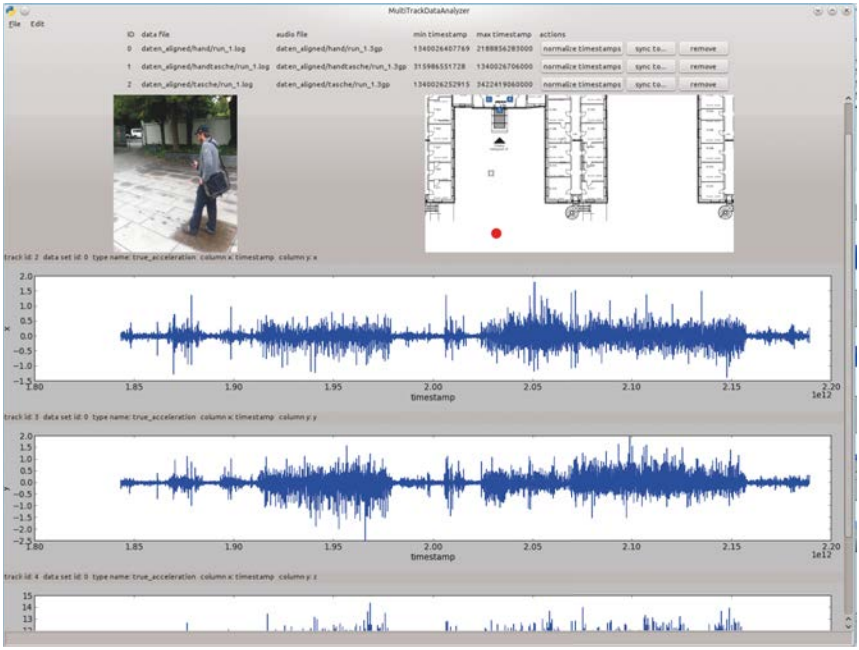


Figure 2: Screenshot of the MTDT. At the top, there is a list of three data sets which have been imported. Below, there is a video track, a map track, and three static plot tracks.

- magnetometer
- gyroscope
- orientation
- rotation matrix
- light
- wifi (available access points and signal strengths)
- gps
- audio

The data is written to a single CSV file (except for the audio data which is written to its own file), individual sensors are identified by a unique ID. The format of the data set is exactly the format used in the MTDT and thus, the data set can be used without further configuration or pre-processing tasks.

5 Automatic synchronization

When using several devices to observe an experiment and create sensor logs, one ends up with several data sets. As described in section 3.1, the MTDT allows to import more than one data set. However, oftentimes the created logs are not perfectly in sync.

In case GPS data has been logged, one can use the timestamps of these records because GPS timestamps should be sufficiently accurate to align different data sets. Otherwise, usually the internal device clocks are used to create the timestamps, which probably are not accurate enough to align several data sets.

In the latter case, the MTDT includes an automatic synchronization mechanism. The only requirement is the availability of an audio recording in each of the data sets which usually can be achieved when using smartphones. Synchronization then is performed by calculating the offset of the audio files.

5.1 Algorithm

We use a simple algorithm which was inferred from what a human user usually would do when told to manually align two audio tracks in a audio editing software like Audacity. In figure 3 one can see an example of two audio tracks recorded on different devices but at the same time.

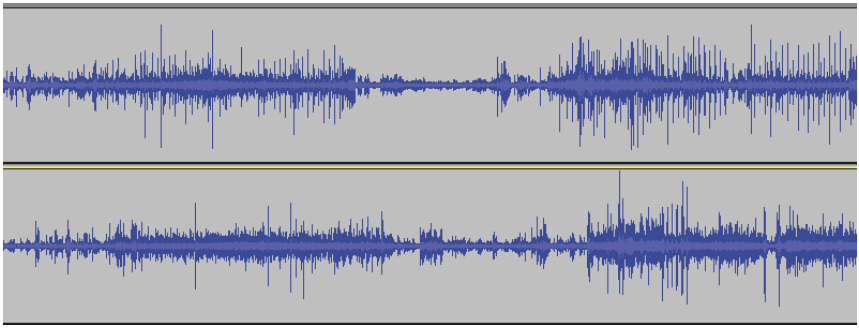


Figure 3: Example of two audio tracks recorded on different devices during the same experiment. The wave forms look quite similar except for the second one being shifted to the right by some amount of time.

One can see that both wave forms look similar except for the second one being shifted to the right. Manually, one now would try to move the second wave form to the left until the amplitudes match better. We automated this process by the following algorithm.

Each audio track essentially is a list of sample values. These lists are now divided into buckets of size M, i.e. each containing M sample values. For each bucket, the sum of the absolute values of the samples is calculated. These sums can be regarded as a representation of the amount of energy contained in the bucket. Buckets with more energy (i.e. many high amplitudes) correspond to the “bubbles” which can be seen in the audio track visualization.

The described process leads to two lists of bucket values. These two lists are now aligned in every possible position (i.e. every possible offset) and every time, the sum of the differences of each pair of aligned buckets is calculated. This sum can be seen as some kind of distance between both lists in the given constellation.

The offset with the smallest calculated distance is selected and the whole process is repeated with a smaller M. However, in those subsequent steps, not all possible offsets have to be examined but only those which overlap with the previously calculated position.

5.2 Evaluation

We conducted a small evaluation of the automatic synchronization algorithm. For a study concerning pedestrian dead reckoning algorithms, we collected about 100 traces of pedestrians, each carrying three smartphones in different positions (hand, trouser pocket, bag). The smartphones ran our logging application (see section 4). Though not intended for evaluation of our synchronization algorithm, the collected traces were a good example of data sets we could test the algorithm with.

For each trace, we tried to synchronize the three corresponding data sets with the described algorithm. Synchronization of all three data sets worked in approximately 50 percent of the cases. Synchronizing at least two of three data sets yielded a success rate of about 65 percent.

We looked for the reasons why the algorithm failed in some cases and found that in these cases, even manual alignment of the tracks would have been nearly impossible due to no clearly visible similarity between the wave forms of the tracks. However, being able to manually align the tracks basically was the prime assumption the algorithm is based on. It is no suprise that the algorithm does not work when this assumption does not hold true.

This result means that the algorithm is not as widely usable as we hoped it to be, but on the other hand, if the basic assumption holds true, does work very well. However, one can improve the performance of the algorithm by simply providing hints in the audio recording such as clearly identifiable claps of a slateboard like it is done in movie productions.

6 Conclusion and future work

In this work, we presented the concept of the Multi Track Data Tool intended to improve working with time series data from sensor logs, especially when examining data of several sensors at once. We are using a prototype in our work in the field of context recognition and intend to provide the MTDT as an open source tool for the scientific community. Collaboratively working on the MTDT hopefully will lead to a broad range of available track types.

Most of the requirements presented in section 2 are basically met by the current prototype, except for the edit and export features which the tool still lacks. We intend to improve the tool in this area as well.

The presented algorithm for automatic synchronization of several data sets will be examined further on its own. We will also look into other algorithms for matching audio tracks because the currently used algorithm can be regarded as a “works for now” solution but probably can be replaced by a better algorithm in the future.

References

[and] Android. http://www.android.com/. Accessed: 2012-10-13.

[aud] Audacity. http://audacity.sourceforge.net/. Accessed: 2012-10-13.

[GIK10] D. Gusenbauer, C. Isert, and J. Krosche. Self-contained indoor positioning on off-the-shelf mobile devices. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, pages 1–9. IEEE, 2010.

[map] mapnik. http://mapnik.org/. Accessed: 2012-10-13.

[mat] matplotlib. http://matplotlib.org/. Accessed: 2012-10-13.

[num] NumPy. http://numpy.scipy.org/. Accessed: 2012-10-13.

[pyg] pygame. http://www.pygame.org/. Accessed: 2012-10-13.

[pyt] Python Programming Language. http://www.python.org/. Accessed: 2012-10-13.

[sci] SciPy. http://www.scipy.org/. Accessed: 2012-10-13.

[wxp] wxPython. http://www.wxpython.org/. Accessed: 2012-10-13.

Adaptive Objektlokalisierung durch Tiefenbildanalyse mittels einer Kinect-Kamera

Mario Haustein, Andreas Löscher, Matthias Werner

Professur Betriebssysteme
Technische Universität Chemnitz
Straße der Nationen 62
09111 Chemnitz
{hamari,loesa,werne}@informatik.tu-chemnitz.de

Abstract: Lokalisierungssysteme, die geometrische Positionen bereitstellen sollen, erfordern in der Regel eine vorherige Kalibrierung an ihre Umgebung. Dieser Aufwand ist für einfache Versuchszwecke meist nicht gerechtfertigt oder im Falle einer im Vorfeld unbekannten oder unzugänglichen Versuchsumgebung gar unmöglich. Der hier präsentierte Ansatz nutzt ein primär zur Gestenerkennung entwickeltes Kamerasystem, um die Positionen sich bewegender Objekte anhand von Tiefenbildern zu bestimmen. Das Lokalisierungssystem passt sich dabei adaptiv an die Einsatzumgebung an.

1 Einleitung

Lokalisierungssysteme erfordern stets eine gewisse Infrastruktur. Diese Infrastruktur muss an die Umgebung, in der die Lokalisierung durchgeführt werden soll, angepasst werden. So sind beispielsweise für RFID-gestützte Lokalisierungssysteme Markierungen durch spezielle Tags notwendig [6], für die Feldstärkenlokalisierung wird eine durch Messung gewonnene Fingerprint-Datenbank benötigt und laterationsbasierte Verfahren erfordern eine genaue Vermessung der Positionen ihrer Basisstationen. Besonders für Erprobungs- und Demonstrationszwecke stellen diese Voraussetzungen aber einen erheblichen Nachteil dar, da die Lokalisierungssysteme in diesem Fall selbst portabel sein müssen und die Einsatzumgebung unbekannt und zumeist im Vorfeld unzugänglich ist.

Das hier beschriebene Lokalisierungssystem entstand aus der Notwendigkeit heraus, die Positionsbestimmung von Robotern in kleinräumigen Versuchsumgebungen durchführen zu können [1]. Dabei galt es folgende Anforderung zu erfüllen.

- Das Lokalisierungssystem muss in der Lage sein bewegliche Objekte zu lokalisieren.
- Die Update-Rate soll 10 Hz nicht unterschreiten.
- Das Lokalisierungssystem soll geometrische Koordinaten bereitstellen.
- Das Lokalisierungssystem soll adaptiv sein, d.h. kein Vorwissen über seine Einsatzumgebung benötigen.
- Eine spezielle Markierung der zu lokalisierenden Objekte im Vorfeld soll nicht notwendig sein.

Grundlage für das Lokalisierungssystem bildet ein optisches Lokalisierungsverfahren. Im Gegensatz zu weit verbreiteten Ansätzen wie der Erkennung bestimmter Markierungen [7, 3] oder der Berechnung von optischen Flüssen [2] aus zweidimensionalen Bilddaten [4], stellt die hier eingesetzte Hardware bereits Tiefeninformationen bereit. Durch Auswertung dieser Tiefeninformationen kann auf zeitintensive Bilderkennungsverfahren verzichtet werden.

Die Arbeit ist im weiteren wie folgt gegliedert. Zunächst werden in Abschnitt 2 Funktionsweise und Kenndaten des Tiefensensors vorgestellt. Abschnitt 3 erläutert die prinzipielle Funktionsweise der Tiefenbildlokalisierung und

zeigt die einzelnen Verarbeitungsstufen des Lokalisierungsprozesses auf. Anschließend werden unter 4 die wesentlichen Kenndaten des Lokalisierungssystems zusammengefasst, Einsatzgrenzen aufgezeigt und Ansatzpunkte für Verbesserungen diskutiert.

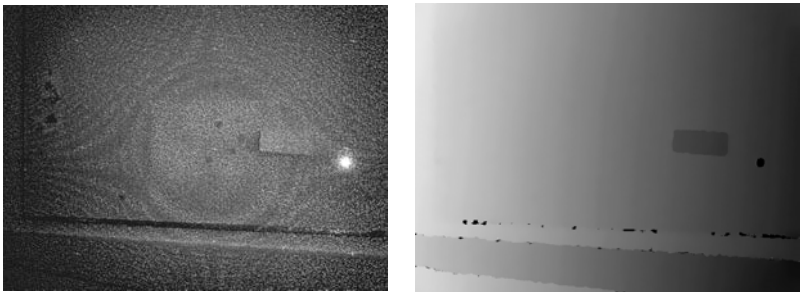
2 Tiefenbildsensoren

2.1 Funktionsweise

Tiefenkamera	$58{,}8° \times 45{,}6°$, 640×480, $30\,\mathrm{Hz}$
Arbeitsbereich	$[1{,}2\,\mathrm{m}, 3{,}5\,\mathrm{m}]$[1]
	$[50\,\mathrm{cm}, 8\,\mathrm{m}]$[2]
Auflösung	$1\,\mathrm{mm}$
Farbkamera	$62{,}0° \times 48{,}6°$, 640×480, $30\,\mathrm{Hz}$

Abbildung 1: Kinect-Kamera

Als Tiefenbildsensor dient die Kincect-Kamera von Microsoft (Abb. 1). Dieses Kamara-System besteht aus zwei Einzelkameras (Infrarot + RGB) und einem Infrarotlaser, der ein statisches, punktförmiges Muster in den Raum wirft (siehe Abb. 2a). Die Projektion dieses Muster auf die Umgebung wird von der Infrarotkamera aufgenommen. Aufgrund der aus der Parallaxe resultierenden Verschiebung des aufgenommenen Punktmusters gegenüber dem Referenzmuster, lässt sich die Entfernung der Punktprojektion von der Kameraebene bestimmen.



(a) Infrarotbild

(b) Tiefenbild

Abbildung 2: Tiefendaten

Label

Hauptsächlich wird diese Kamera innerhalb der Computerspieleindustrie als Eingabegerät für Gesten eingesetzt, was sich auch in den meisten Programmierschnittstellen für diesen Sensor niederschlägt. [11, 9] Dennoch besteht durch die Treiberbibliothek „libfreenect“ [8] auch die Möglichkeit, direkt auf das Tiefenbild zugreifen zu können. Dabei kann dieses Tiefenbild sogar auf die Bildkoordinaten des RGB-Kanals umprojiziert werden, wodurch sich zu jedem Bildpixel der Farbkamera ein Tiefenwert angeben lässt.

[1] laut Hersteller (FIXME: Quelle)
[2] eigene Versuche

2.2 Genauigkeit

Bedingt durch das Funktionsprinzip des Sensors unterliegen die Tiefendaten im Wesentlichen drei Fehlerquellen, die bei der Lokalisierung berücksichtigt werden müssen. Befindet sich ein Objekt an einer herausgehobenen Stelle der Szene (wie z.B. im rechten Teil von Abb. 2), ist der Rand zwischen Objekt und dem Hintergrund unstetig. Aufgrund kleiner Abweichungen im Messprozessprozess kann es in der Folge passieren, dass ein Pixel auf diesem Rand zu einem Zeitpunkt dem Hintergrund (großer Tiefenwert) zugeordnet wird und zu einem anderen Zeitpunkt dem Vordergrund (kleiner Tiefernwert). Hieraus ergibt sich eine starke szenenbedingte Streuung einzelner Tiefenwerte.

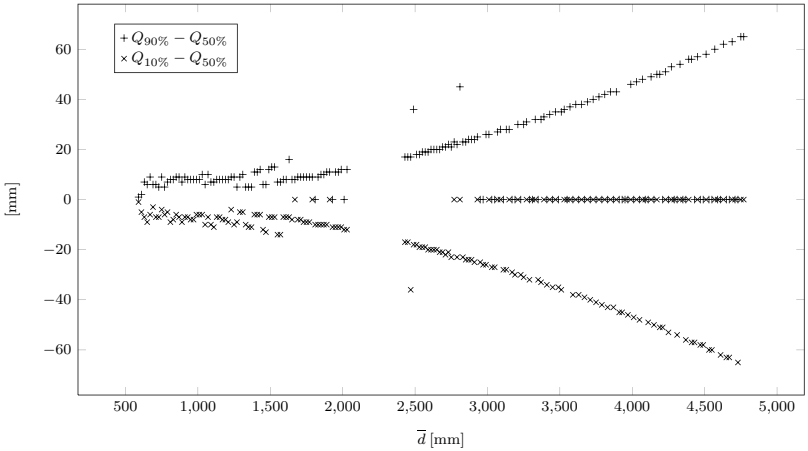


Abbildung 3: Streuung der Tiefenwerte

Obwohl der Treiber die Tiefendaten mit einer Auflösung von $1\,\mathrm{mm}$ bereitstellt, unterliegen die Werte z.T. erheblichen Schwankungen. Abbildung 3 zeigt die Schwankungen der Tiefenwerte anhand der Differenz des 90%-Quantils und des 10%-Quantils zum Median in Abhängigkeit zum mittleren Entfernungswert $\overline{d}$. Bei dieser Messung wurden bereits die oben beschriebenen szenenbedingten Streuungen herausgefilter, um die Messwerte nicht zu verfälschen. Es ist zu erkennen, dass sich 80% aller Tiefenwerte symmetrisch um den Median herum verteilen. Zudem nimmt der Messfehler quadratisch mit steigender Entfernung zu. Grund dafür ist das mit zunehmender Entfernung größer werdende Verhältnis von Entfernungsänderung zu parallaktischer Verschiebung. Im Rahmen der Messungen hatten die Extremwerte z.T. mehr als die doppelte Abweichung zum Median im Vergleich zu den oben dargestellten Quantilen.

Der dritte Fehler ist technischer Natur. Durch Spekularreflexion oder Abschattung, kann in einigen Bildbereichen das IR-Punktmuster nicht wahrgenommen werden. An diesen Stellen kann damit auch keine Aussage über die Bildtiefe getroffen werden, es liegen folglich Datenlücken vor.

3 Tiefenbildlokalisierung

3.1 Funktionsweise

Die adaptive Tiefenbildlokalisierung basiert auf der Trennung von beweglichen Vordergrundobjekten vom Hintergrund der Szene. Sie fußt dabei im Wesentlichen auf zwei Grundannahmen:

1. Es gibt einen statischen Hintergrund. Die Bewegungen der Objekte spielen sich vor dem Hintergrund ab.
2. Verändert sich der Tiefenwert eines Bildpixels, ist davon auszugehen, dass der größte aller Werte der Tiefe des Szenenhintergrunds entspricht.

Basierend auf den obigen Annahmen lässt sich folgendens Lokalisierungsverfahren ableiten. Während des gesamten Lokalisierungszeitraums wird ein Hintergrundbild erstellt. Zum Initialisierungszeitpunkt des Systems, wird die gesamte Szene mangels besseren Wissens als Hintergrund betrachtet. Sobald sich eines der zu lokalisierenden Objekte zu bewegen beginnt, ändern sich die Tiefenwerte. Werden die Tiefenwerte für ein Pixel geringer, ist davon auszugehen, dass sich ein Objekt vor den Hintergrund geschoben hat. Als Hintergrundswert wird weiterhin der alte (d.h. größere) Tiefenwert herangezogen. Steigen die Tiefenwerte jedoch, muss zuvor der Hintergrund durch ein unerkanntes Objekt verdeckt gewesen worden sein. Der neue (d.h. größere) Tiefenwert wir zukünftig als Hintergrundwert herangezogen. Weicht der Tiefenwert eines Pixel irgendwann um einen bestimmten Schwellwert vom Hintergrundwert ab, kann diese Pixel als zum Vordergrund gehörig betrachtet werden. Durch dieses Verfahren entsteht somit eine Bitmaske, die für jedes Pixel entweder dem Vordergrund oder dem Hintergrund zuordnet.

3.2 Objekterkennung

Um den in Abschnitt 2.2 beschriebenen Fehlern Rechnung zu tragen, kann das oben dargelegte Lokalisierungsverfahren nicht eins zu eins umgesetzt werden. Abbildung 4 zeigt die einzelnen Schritte zur Objekterkennung schematisch auf.

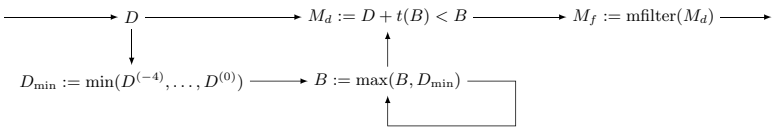


Abbildung 4: Schema zur Objekterkennung

Das Hintergrundbild B wird dabei nicht direkt aus den bereitgestellen Tiefenbildern D berechnet. Da die Werte in diesen Tiefenbildern extreme Schwankungen aufweisen können, würden die Maxima aufs Hintergrundbild durchschlagen. Der Erkennungsschwellwert müsste in diesem Fall sehr groß gewählt werden, damit es nicht zu Falsch-Positiv-Fällen kommt. Aus diesem Grund wird zunächst das Minimum über die vergangenen fünf Tiefenbilder $D_{\min}$ berechnet. Dieses Verfahren erlaubt bis zu vier Ausreiser in Folge, kann aber auch zu Falsch-Negativ-Fällen führen. Dies ist genau dann der Fall, wenn der Hintergrund hinter einem beweglichen Pbjekt für höchsten fünf Frames sichtbar ist. Bei einer Framerate von $30\,\mathrm{Hz}$ müsste das Objekt also innerhalb von max. $167\,\mathrm{ms}$ seine Ausgangsposition wieder einnehmen, was als vertretbar hingenommen werden kann.

Die Berechnung des Hintergrundbildes erfolgt schließlich durch fortwährende Maximumbildung des gleitenden Minimas. Durch Vergleich von Tiefen- und Hintergrundbild kann eine Bitmaske M_d erstellt werden, die das Tiefenbild in Vordergrund (1-Bit) und Hintergrund (0-Bit) klassifiziert. Der Schwellwert $t(\cdot)$ zur Trennung von Vorder- und Hintergrund ist dabei abhängig von der Entfernung, um dem in Abbildung 3 dargestellten Fehler Rechnung zu

tragen. Für $t(\cdot)$ kommt der quadratische Ansatz

$$t(x) = a \cdot x^2 + b \cdot x + c$$

mit den folgenden empirisch ermittelten Parametern zur Anwendung:

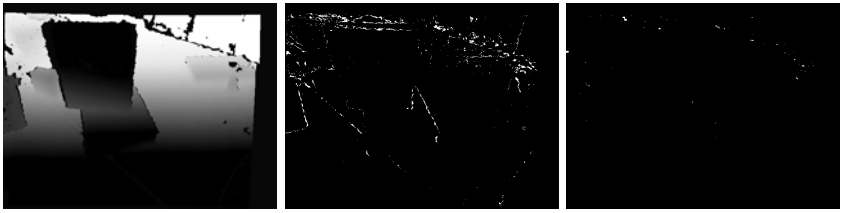
$$\begin{aligned} a &= 150000\,\mathrm{mm}^{-1} \\ b &= 0 \\ c &= 5\,\mathrm{mm} \end{aligned}$$

Tabelle 1 zeigt die Größe des Schwellwerts für ausgewählte Entfernungen.

x	[mm]	500	1000	2000	3000	4000	5000	6500	8000
$t(x)$	[mm]	7	12	31	65	112	172	287	432

Tabelle 1: Schwellwerte

Aufgrund der szenenbedingten Streuung werden durch den Schwellwert-Ansatz nicht nur bewegte Objekte sondern auch die Kanten stationärer Objekte erkannt. An solchen Kanten setzt sich auf lange Zeit der Hintergrundwert durch, und sobald der Sensor den Rand dem näheren Objekt zuordnet, wird dieser als Vordergrundobjekt klassifiziert. Die Abbildungen 5a und 5b verdeutlichen diesen Effekt.



(a) Szene (b) Maske (ungefilter) (c) Maske (gefilter)

Abbildung 5: Objekterkennung

Die Maske M_d muss folglich noch auf geeignete Weise gefiltert werden. Hierzu kommen die Operationen $\mathrm{dilate}(\cdot)$ (anlagern) und $\mathrm{erode}(\cdot)$ (abtragen). Diese Operationen können auf Bitmasken angewendet werden. Im Rahmen der Operation $\mathrm{dilate}(\cdot)$ sind dabei die 1-Bits dominant, d.h. im Ergebnis der Operation enthält jedes Pixel genau dann ein 1-Bit, wenn sich in der Eingabe innerhalb seiner 8-Nachbarschaft mindestens ein 1-Bit befindet. Für die Operation $\mathrm{erode}(\cdot)$ gilt die gleiche Aussage analog für 0-Bits. Die Operation $\mathrm{mfilter}(\cdot)$ setzt sich nun wie folgt zusammen:

$$\mathrm{mfilter}(\cdot) = (\mathrm{erode} \circ \mathrm{erode} \circ \mathrm{dilate} \circ \mathrm{dilate} \circ \mathrm{erode})(\cdot)$$

In einem ersten Schritt werden die durch szenenbedingte Streuung entstandene Ränder abgetragen. Dadurch werden aber evt. bestehende Lücken vergrößert. Diese müssen im Anschluss durch zwei Anlagerungsschritte gefüllt werden. Weil feine Ränder bereits komplett entfernt wurden, können Sie durch Anlagerung nicht anwachsen. Da die Maske nun eine größere Ausdehnung besetzt als M_d, muss wieder abgetragen. In diesem Fall wird zweifach abgetragen, womit die Maske insgesamt etwas kleiner als M_d ist. Dies ist notwendig, um bei einer späteren Kantenerkennung sicherzustellen, dass der Kantenzug mit Sicherheit über Vordergrundpixeln verläuft.

3.3 Objekttrennung

Nach dem Erkennungsschritt liegt eine gefilterte Objektmaske vor. Dennoch entspricht nicht jede Zusammenhangskomponente innerhalb der Maske genau einem Objekt. Durch Verdeckung mehrerer Objekte ist es möglich, dass diese zu einem Bereich in der Objektmaske verschmelzen (siehe Abb. 7a). Aus diesem Grund müssen die Objekte vor der Positionsbestimmung getrennt werden. Das grundlegende Schema des Trennungsverfahren ist in Abbildung 6 dargestellt.

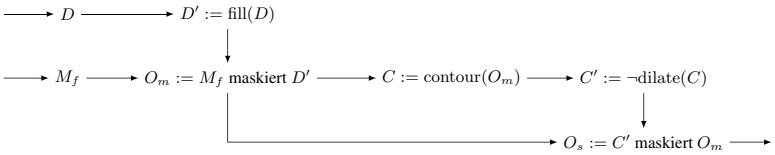


Abbildung 6: Schema der Objekttrennung

Eine Überlappung von Objekten zeichnet sich dadurch aus, dass im Tiefenbild sehr starke Sprünge der Tiefenwerte stattfinden. Diese Sprungstellen, können durch Kantenerkennungsalgorithmen, wie dem Canny-Algorithmus [5] berechnet werden. Für diese Algorithmen ist es jedoch wichtig, dass keine Datenlücken innerhalb des Tiefenbilds vorliegen. Die Fülloperation $\text{fill}(\cdot)$ schließt Datenlücken, durch Interpolation aus Nachbarpixeln (D'). Jedes Fehlpixel nimmt im Rahmen der Fülloperation den arithmetischen Mittelwert aller definierten Pixel in seiner 4-Nachbarschaft an. Mit jeder interativen Anwendung der Fülloperation auf das Tiefenbild werden Fehlstellen um eine Pixelbreite verkleinert. Durch dieses Verfahren erfolgt keine lineare Interpolation, sondern Objekte wachsen mit ihrer Randtiefe in die Breite, bis sie durch andere Objekte begrenzt werden. Diese scharfen Kanten sind für die anschließende Kantenerkennung notwendig. Das Füllbild D' wird schließlich mit M_d maskiert und ergibt das Tiefenbild aller Vordergrundpobjekte O_m (siehe Abb. 7c).

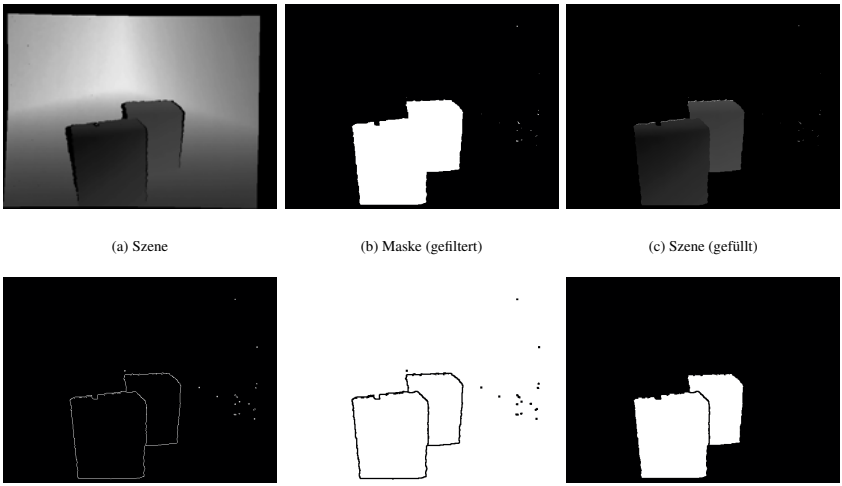
Auf O_m wird eine nun eine Kantenerkennung angewendet, deren Ergebnis mit C bezeichnet werde. Zum einen werden dadurch die Ränder der Objektmaske erkannt. Zum anderen entstehen auch an besagten Sprungstellen Kanten (siehe Abb. 7d).

Die Trennungskanten verlaufen nun genau entlang der Grenze, die sich zwischen zwei Objekten ausbildet. Um allerdings Randkonturen für jedes der Objekte zu erhalten, kann eine weitere Maske erstellt werden (siehe Abb. 7f). Sie entsteht, indem man zunächst die Operation $\text{dilate}(\cdot)$ auf C anwendet und anschließend invertiert. Das Zwischenprodukt C' wird letzlich mit M_f durch ein logisches UND verknüpft. Alternativ kann auch das Zwischenergebnis C' direkt auf O_m anstatt M_f anwenden, wie in Abbildung 6 gezeigt, wodurch ein Tiefenbild der einzelnen Objekte entsteht. Die Aufweitung der Trennungskanten durch die dilate-Operation stellt sicher, dass die Kanten, die sich später um die getrennten Vordergrundobjekte herum ergeben, stets über den Pixeln im ursprünglichen Tiefenbild liegen, die auch zum entsprechenden Objekt gehören.

3.4 Vektorisierung und Tracking

Bisher fanden alle Bearbeitungsschritte auf Rasterbildern statt. Da nun alle Objekte im Rasterbild identifiziert wurden, kann eine Vektorisierung erfolgen. Jedes Objekt wird dabei durch seine Randkontur – genauer, einer Folge von Eckkordinaten und ihrer Entfernung von der Kameraebene – beschrieben. Die Konturenerkennung erfolgt dabei auf der getrennten Objektmaske (Abb. 7f). Konturen unterhalb einer gewissen Mindestfläche werden generell verworfen, um letzte Artefakte, die aus Messfehlern resultieren, zu filtern. Zusätzlich werden alle Polygone verworfen, die durch andere Polygone eingeschlossen werden. Damit werden Löcher innerhalb der zu lokalisierenden Objekte ignoriert.

Die Randpolygone der einzelnen Frames müssen nun zueinander in Beziehung gesetzt werden, da nur so Posi-



(a) Szene (b) Maske (gefiltert) (c) Szene (gefüllt)

(d) Kanten (e) Trennungsmaske (f) Maske (getrennt)

Abbildung 7: Objekttrennung

tionsangaben zu einem Objekt zugeordnet werden können. Ein einfacher Ansatz besteht in der Defintion einer Ähnlichkeitsnorm zwischen zwei Konturen C_1 und C_2, die es erlaubt Objekte aufeinanderfolgender Frames miteinander zu korrelieren. Diese Norm, sollte weder skalen- noch translationsinvariant sein. Ein naiver Ansatz ist durch folgende Gleichung gegeben.

$$\langle C_1, C_2 \rangle = \sqrt{(\overline{x}(C_1) - \overline{x}(C_2))^2 + (\overline{y}(C_1) - \overline{y}(C_2))^2} + \sqrt{|\Omega(C_1) - \Omega(C_2)|}$$

Die Größen $\overline{x}$ und $\overline{y}$ bezeichnen die Position des Flächenschwerpunkts in Pixelkoordinaten und Ω gibt den Raumwinkel eines Objekts an, der dem Flächeninhalt der Objektmaske entspricht. Der Wert der Norm wird um so kleiner, je ähnlicher die Objekte sind. Es handelt sich genau genommen also um eine Unähnlichkeitsnorm. Die Form eines Objekts wird von der hier beschriebenen Norm jedoch nicht berücksichtigt.

Zur Korrelation zweier Objekte zwischen aufeinanderfolgender Frames werden die Konturen des alten Frames paarweise mit denen des neuen Frames verglichen. Das Matching mit geringstem Normwert wird alles zusammengehörig betrachtet. Beide Konturen scheiden aus und der Vorgang wird für die restlichen Konturen wiederholt, bis eine der Mengen leer ist. Sollten noch Elemente in der Menge der neuen Konturen enthalten sein, werden diese als neue Objekte betrachtet.

3.5 Positionsbestimmung

Von einem Lokalisierungssystem wird in der Regel erwartet, dass es einen numerischen oder symbolischen Positionswert bereitstellt. Bisher stehen für die lokalisierten Objekte aber lediglich Randkonturen und ein „Höhenprofil“ seiner Vorderseite bereit. Da eine Reduzierung dieser Daten ist in vielen Fällen jedoch anwendungsabhängig ist, macht eine allgemeine Reduktion zu einem Koordinatentripel wenig Sinn. Aus diesem Grund werden Randkontur und Höhenprofil als Position betrachtet.

Für die Lokalisierung konvexer Roboter – dem Anlass für die Entwicklung dieses Lokalisierungsverfahrens – könnte eine Reduktionsverfahren wie folgt aussehen. Der Schwerpunkt der Projektionsfläche stellt einen gut geeigneten Richtungswert dar, da er in jedem Fall innerhalb der entsprechenden Kontur liegt, eindeutig ist und einen guten Mittelwert darstellt. Für eine dreidimensionale Positionsangabe wird allerdings noch ein Tiefenwert benötigt. Die Wahl dieses Tiefenwerts ist auch für das vereinfachte Beispiel der Roboterlokalisierung nicht unbedingt intuitiv. Zum einen kann man als Tiefenangabe den Tiefenwert des Schwerpunkts wählen. Die Entfernungsangabe wird hierbei jedoch i.d.R. unterschätzt, da lediglich die Positions der zur Kamera zeigenden Grenzfläche des Objekts gemessen wird. Eine andere Möglichkeit besteht in der Mittelwertbildung aller Tiefenwerte der Randpunkte. Dieser Wert erfüllt eher die Erfordernisse eines Mittelwerts, ist aber u.U. mit einer stärkeren Streuung behaftet, da die Randpunkte von Frame zu Frame stark schwanken können, obwohl die Randkontur nahezu identisch ist.

4 Zusammenfassung

4.1 Auswertung

Das beschriebene Lokalisierungsverfahren arbeitet für den angedachten Einsatzzweck sehr zufriedenstellend. Die Implementierung des Algorithmus erfolgte auf einem gewöhnlichen Desktop-PC[3] unter Verwendung der OpenCV-Bibliothek [10]. Pro Frame ergibt sich auf dem Testsystem eine Gesamtberechnungszeit von $36\,\mathrm{ms}$, was ca. $27\,\mathrm{Hz}$ entspricht. Die $\mathrm{fill}(\cdot)$-Operation – die einzige Operation, die nicht durch OpenCV bereitgestellt wird – macht dabei bereits $15{,}4\,\mathrm{ms}$, also über 40% der Berechnungszeit, aus.

Objekte müssen zwei geometrische Mindestanforderungen erfüllen, um lokalisiert werden zu können. Zum einen müssen sie den in Tabelle 1 angegebenen Mindestabstand vom Hintergrund aufweisen. Zum anderen müssen sie einen gewissen Mindestraumwinkel besitzen, um in dem in Abschnitt 3.4 beschriebenen Lokalisierungsverfahren nicht gefiltert zu werden. Der Schwellwert wurde empirisch auf einen Wert festgesetzt, der bei kreisförmigen Objekten etwa einem Öffnungswinkel von $1{,}4°$ entspricht. Damit lassen sich Roboter mit nur $5\,\mathrm{cm}$ Durchmesser in einem Abstand von $2\,\mathrm{m}$ lokalisieren.

Dennoch unterliegt das Lokalisierungssystem einigen Einschränkungen, die technischer bzw. prinzipieller Natur bedingt sind.

4.1.1 Prinzipielle Einschränkungen

Zu den prinzipiellen Einschränkungen zählen Überlappung, Verdeckung und die Berührung von Körpern. Eine Überlappungssituation ist in Abbildung 7 dargestellt. Die Szene besteht aus zwei Quadern, wobei ein Quader den anderen teilweise überlappt. Der zweite Quader wird als einzelnes Objekt erkannt, jedoch entspricht seine Randkontur nicht der eines Quaders. Analyseverfahren zur Formerkennung sind folglich nur sehr beschränkt einsetzbar.

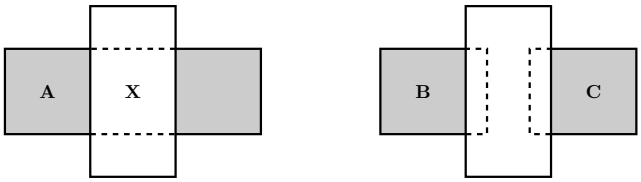


Abbildung 8: Verdeckungsfehler

[3] Intel Core 2 Duo 2,1 GHz (nur ein Kern wird genutzt)

Wohingegen beim Auftreten von Überlappungsfehlern noch alle Objekte zuverlässig erkannt werden können, treten bei Verdeckungsfehlern unter Umständen echte Lokalisierungsfehler auf. Die in Abbildung 8 gezeigten Fälle erzeugen identische Tiefenbilder und sind somit ununterscheidbar. Das Lokalisierungsverfahren wird in jedem Fall die Situation (b) erkennen, obwohl auch Situation (a) vorliegen kann.

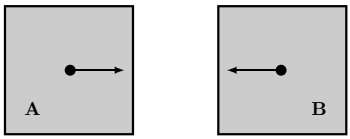


Abbildung 9: Berührungsfehler

Die Berührung zweier Objekte in etwa gleicher Entfernung zur Kamera erzeugt grundsätzlich einen Lokalisierungsfehler. In diesem Fall ist die Unstetigkeit zwischen den Tiefenwerten so gering, dass beide Objekt zu einem Metaobjekt verschmelzen. Die Unähnlichkeitsnorm des Metaobjekts zu den Ausgangsobjekten kann dabei u.U. so groß werden, dass keines der Ausgangsobjekte mehr mit dem Metaobjekt korreliert wird. Eine Trennung der Objekte hätte einen ähnlichen Effekt zur Folge, sodass nun die beiden Objekte im Tracking-Schritt als neu erkannt werden und die Referenz zur ihrer ursprünglichen Tracking-Information verloren ist.

4.1.2 Technische Einschränkungen

Technische Einschränkungen liegen im Gegensatz zu den oben beschriebenen Schwachstellen nicht im Verfahren begründet, sondern resultieren aus der Beschaffenheit der Hardware. Im Rahmen der Implementierung des beschriebenen Lokalisierungsverfahren wurde beobachtet, dass die Rohdaten des Tiefensensors nach einiger Zeit mit einer gradientenhaften Störung überlagert werden, deren Ursache bis jetzt noch nicht ermittelt werden konnte. Dabei steigen die Tiefenwerte im linken Bildteil an und lassen im rechten Bildteil nach. Dieser Übergang findet abrupt wenige Minuten nach Öffnen der Verbindung zur Kamera statt. Die Überlagerung dieses Grandienten führt zu einer pixelabhängigen Veränderung des Schwellwertpuffers, wodurch im linken Bildteil zu Falsch-Negativ-Fehlern (Nichtlokalisierung) und im rechten Bildteil zu Falsch-Positiv-Fehlern (Scheinlokalisierung) kommen kann.

4.2 Ausblick

Im Rahmen der Arbeit konnte gezeigt werden, dass es möglich ist, mittels Tiefensensoren eine Lokalisierung von Objekten vorzunehmen, ohne dabei auf Wissen über die Struktur der Objekte oder die Umgebungsbedingungen zurückzugreifen.

Nachteilig an diesem Verfahren ist allerdings die beschränkte Reichweite, die sich aus dem Bildwinkel der Kamera und der maximal messbaren Bildtiefe ergibt. Eine Möglichkeit dieses Problem zu lösen, ist der Einsatz mehrerer Tiefensensoren. Bei unkoordinierter Nutzung mehrerer Tiefensensoren kommt es jedoch zu einer Überlagerung der Infrarotpunktmuster, sodass eine Tiefenbestimmung unmöglich wird. Lösbar wäre dieses Problem durch Zeitmultiplexing der Infrarotlaser, was jedoch wiederum die Update-Rate des Gesamtsystems verringert. Außerdem erfordert dieser Ansatz die genaue Vermessung von Lage und Ausrichtung der Sensoren untereinander um die Koordinaten der einzelnen Sensoren in ein globales Koordinatensystem umrechnen zu können. Dennoch könnte sich eine solche Erweitung ebenfalls als praktisches und kostengünstiges Lokalisierungsverfahren für Laborumgebungen erweisen.

Literatur

[1] Mario Haustein, Andreas Löscher und Matthias Werner. Constraintbasierte Roboterformationen. Poster, Juli 2012.

[2] Berthold K. P. Horn und Brian G. Schunck. Determining Optical Flow. *ARTIFICAL INTELLIGENCE*, 17:185–203, 1981.

[3] J. J. Leonard und H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, August 2002.

[4] Sooyong Lee und Jae-Bok Song. Robust mobile robot localization using optical flow sensors and encoders. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04.*, Jgg. 1, Seiten 1039–1044, April 2004.

[5] J Canny. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, Juni 1986.

[6] T. Sanpechuda und L. Kovavisaruch. A review of RFID localization: Applications and techniques. In *5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008.*, Jgg. 2, Seiten 769–772, Mai 2008.

[7] A.P. Sample, C. Macomber, Liang-Ting Jiang und J.R. Smith. Optical localization of passive UHF RFID tags with integrated LEDs. In *2012 IEEE International Conference on RFID*, Seiten 116–123, April 2012.

[8] OpenKinect, libfreenect API, `http://openkinect.org/wiki/Main_Page`

[9] Microsoft Kinect API, `http://msdn.microsoft.com/en-us/library/hh855352.aspx`

[10] Open Source Computer Vision Bibliothek, `http://opencv.org/`

[11] OpenNI Framework, `http://openni.org/Documentation/ProgrammerGuide.html`

Modularisierte Routenplanung mit der donavio-Umgebung

Jörg Roth

Fakultät Informatik
Ohm-Hochschule Nürnberg
Kesslerplatz 12
90489 Nürnberg
Joerg.Roth@Ohm-Hochschule.de

Abstract: Dieser Beitrag stellt die *donavio*-Umgebung vor, die ein vollständiges Gerüst für alle Funktionen von Anwendungen zur Routenplanung anbietet. Die Umgebung unterstützt den Einsatz verschiedener Verteilungsarchitekturen, z.B. der Zugriff auf einen Routing-Service über eine Web-Service-Schnittstelle oder die Offline-Navigation auf einem Smart-Phone. Insbesondere die Algorithmen der Routenplanung können modifiziert werden, so sind verschiedene Heuristiken einsetzbar. Die Konfigurierbarkeit geht hinunter bis auf niedrige Ebenen des Datenzugriffs – so können die Laufzeitstrukturen des eingesetzten A*-Verfahrens bezüglich Zeit- und Speicherbedarf abgestimmt werden. Auch für den Zugriff auf die umfangreichen Straßennetz-Daten können verschiedene Varianten eingesetzt werden, optimiert beispielsweise entweder für die Speicherung auf dem Smart-Phone oder auf einem Server.

1 Einleitung

Die Suche nach optimalen Wegen in Straßennetzen ist ein gut erforschtes Gebiet. Die meisten Arbeiten behandeln jedoch im Schwerpunkt die Graphenalgorithmen, meist Variationen des A*-Algorithmus, um möglichst effizient einen Weg auf einem topologischen Straßennetz zu finden. Für den realen Einsatz sind jedoch weitere Funktionen notwendig, z.B. die Entgegennahme von Routing-Aufträgen oder die Generierung von Abbiegekommandos. Bezüglich dieser Funktionen gibt es eine Reihe von Freiheitsgraden. Als Beispiele:

- Wo wird welche Funktion ausgeführt, insbesondere in einer Client-Server-Umgebung. Soll z.B. eine Offline-Navigation auf dem Smart-Phone ausgeführt werden, oder soll die Routenplanung außerhalb des Endgerätes über eine Web-Service-Schnittstelle aufgerufen werden.

- Wie werden die umfangreichen Daten des Straßennetzes gespeichert z.B. als SQL-Datenbank, im Laufzeitspeicher oder auf dem Flash-Speicher eines mobilen Gerätes. Insbesondere ist zu klären, wie man geometrische Abfragen durch einen räumlichen Index beschleunigt.

- Für Algorithmen der Routenplanung (insbesondere im Zusammenhang mit A*) gibt es verschiedene Varianten der Laufzeitoptimierung. Beispiele: Schätzung mit overdo, vorberechnete ALT-Potenziale, ignorieren langsamer Straßen in der Routenmitte.

- Auch für die Generierung der Abbiegekommandos sind verschiedene Ansätze denkbar. So gibt es unterschiedliche Verfahren zur Erkennung, ob man eine Straße verlässt (Namen, Kurvengeometrien, Vorfahrtregeln). Abbiegevorgänge können unter Einbeziehung der exakten Straßengeometrien oder Straßennamen genauer formuliert werden.

In der aktuellen Situation sind die eigentlichen Graphenalgorithmen wissenschaftlich zwar gut untersucht, die genauen Ausgestaltungen verschiedener Varianten, die insbesondere in kommerziellen Routenplanern zum Einsatz kommen, sind aber oft ein Geschäftsgeheimnis. Ziel unserer Arbeit ist daher, eine Umgebung bereitzustellen, die neben der Routenplanung alle Aspekte einer Navigationsanwendung abdeckt. Unterschiedliche Ansätze zu bestimmten Teilbereichen sollen leicht integrierbar sein und getestet werden können.

Es gibt einer Reihe von frei verfügbaren Routenplanungsdiensten, z.B. Open Route Service [NZ08] oder die Routenplanung von Google Maps – hierbei sind allerdings die Interna der Dienste nicht modular austauschbar. Außerdem sind verschiedene Architekturvarianten wie z.B. die Offline-Navigation auf Smart-Phones, nicht möglich.

2 Die donavio-Umgebung

2.1 Einleitung

Vielfach entscheidet man sich bei einer speziellen Softwareumgebung für eine bestimmte Variante einer Teillösung und legt damit den gesamten Entwurf fest. Veränderungen sind dann nur noch mit viel Aufwand nachträglich integrierbar. Ziel der *donavio*-Umgebung ist es, eine modulare Plattform anzubieten, mit der verschiedene Varianten integriert und ausprobiert werden können. Insbesondere sollen unterschiedliche Architekturen (z.B. Client-Server, Offline-Navigation auf Smart-Phones) getestet werden können. Erreicht wird die Modularität über eine Komponentenarchitektur mit festen Schnittstellen:

- Alle auswechselbaren Module werden über objektorientierte Schnittstellen aufgerufen. Diese Vorgehensweise entspricht dem Standardmuster, wenn man unterschiedliche algorithmische Varianten in einem Rahmenwerk kapseln möchte.

- Größere Komponenten (z.B. die Suche nach Straßen über den Namen) haben *zusätzlich* eine Web-Service-Schnittstelle. Hierdurch ist eine flexible Verteilung zwischen Client und Server möglich.

- Einige Zugriffsfunktionen sind so zeitkritisch, dass eine Kapselung über eine Objektschnittstelle zu viel Aufwand zur Laufzeit erzeugen würde. Ein Beispiel ist die Organisation der Knotenlisten innerhalb von A*. Diese kann über feste Arrays signifikant beschleunigt werden. Prinzipiell könnte der Array-Zugriff zwar über Objektschnittstellen gekapselt werden, hiermit würde man aber den Laufzeitvorteil aufgeben. Als Lösung werden zeitkritische Module über Macro-Preprocessing integriert.

Die donavio-Umgebung ist in Java realisiert und läuft auf Desktop-Systemen (Stand-alone oder über Web-Service-Schnittstelle) und als Android-App (Offline-Navigation oder als Web-Service-Client). Zur Beschreibung der Umgebung kann man zwei Anwendungsfälle unterscheiden:

1. *Bereitstellung*:

   Navigationsdaten und Code werden für eine bestimmte Export-Konfiguration oder ein bestimmtes Szenario generiert und für die Benutzung bereitgestellt. Die Daten können beispielsweise auf die Art der Fortbewegung angepasst werden. Der Code kann beispielsweise für eine bestimmte Einsatzumgebung exportiert werden. Da die Daten für die eingesetzten Algorithmen optimiert exportiert werden, müssen Code und Daten zusammenpassen.

2. *Benutzung zur Laufzeit*:

   Der generierte Code wird mit den exportierten Daten verwendet. Die Ausführung kann in bestimmten Grenzen durch eine Laufzeit-Konfiguration beeinflusst werden. Beispielsweise kann ein bestimmtes Verkehrsmittel aus der Liste von denjenigen Verkehrsmitteln ausgewählt werden, die von den exportierten Daten unterstützt werden. Die Grenzen der Laufzeit-Konfiguration ergeben sich daraus, wie Daten und Code unter Punkt 1 bereitgestellt wurden.

Der Rest des Kapitels 2 befasst sich mit der Bereitstellung. Kapitel 3 wird die Laufzeitfunktionen darstellen. Kapitel 4 schließlich zeigt, wie die Modularisierung softwareseitig unterstützt wird.

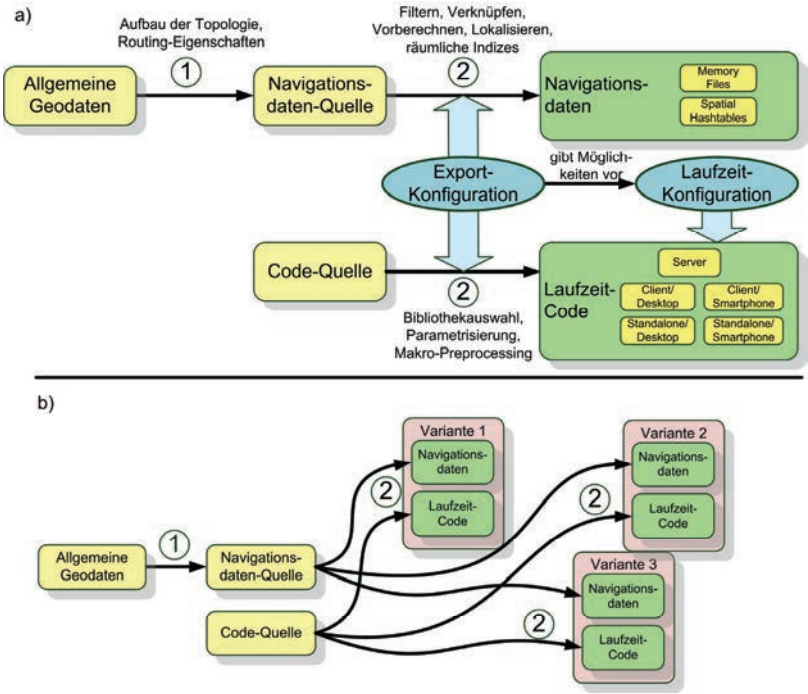


Abbildung 1: a) Phasen der Bereitstellung; b) Erzeugen von Varianten

Abb. 1a) zeigt die Bereitstellung von Code und Daten. Es gibt zwei Phasen:

(1) Aus einem allgemeinen Datenbestand wird eine Navigationsdaten-Quelle abgeleitet. Der Navigationsdatenbestand ist allgemein, beinhaltet beispielsweise alle möglichen Objekte, nicht nur Straßen. Als wichtige Festlegung beim Export wird eine regionale Einschränkung der Daten gemacht, indem z.B. nur alle Einträge aus Bayern zugrunde gelegt werden. Details dieser Übernahme sind in [Ro10b] zu finden.

(2) Aus der Navigationsdaten-Quelle werden für eine bestimmte *Export-Konfiguration* Navigationsdaten abgeleitet. Eine bestimme Export-Konfiguration kann dabei die Einschränkung auf bestimmte Straßen sein (z.B. nur mit dem Fahrrad befahrbar). Zusätzlich werden diverse Vorarbeiten auf den Daten durchgeführt. Parallel zu den Daten wird Laufzeit-Code abgeleitet.

Als Export-Format für die Navigationsdaten werden im Moment *Memory Files* und *Spatial Hashtables* unterstützt (siehe später).

Abb. 1b) illustriert, wie verschiedene Varianten erzeugt werden können. Die Navigationsdaten- und Code-Quelle sind so allgemein, dass verschiedene Laufzeit-Varianten erzeugt werden können. Durch unterschiedliche Export-Konfigurationen von Schritt (2) entstehen so unterschiedliche Paare von Laufzeit-Code und Navigationsdaten. Wichtig ist hierbei, dass diese zueinander passen müssen.

2.1 Übernahme aus dem allgemeinen Geodatenbestand

Die Navigationsdaten werden aus dem allgemeinen Datenbestand von Open Street Map (OSM) [OSM, Ro12a] abgeleitet. OSM-Daten unterstützen nicht vordergründig die Routenplanung, d.h. entsprechende Informationen liegen verstreut im Datenbestand vor. In einem ersten Schritt werden daher alle relevanten Daten herausgerechnet:

- Anhand der Straßengeometrien wird das Wegenetz, also ein Graph aus Knoten und Kanten berechnet. Kreuzende Straßen werden in OSM nur dadurch ausgezeichnet, dass sie einen gemeinsamen Geometriepunkt besitzen. Daher kann die Topologie nur über aufwändige Überkreuzungstests abgeleitet werden.

- Die resultierende Topologie enthält noch keine Sackgassen, da die Sackgassen-Enden formal keine Kreuzung sind. Topologisch sind Sackgassen zwar nicht interessant, später möchte man aber Start oder Ziel in eine Sackgasse legen können. Daher werden entsprechende Knoten und Kanten künstlich integriert.

- Straßen werden in diesem Schritt klassifiziert. Diese Klassifikation wird später verwendet, um festzustellen, von welchem Verkehrsmittel eine Straße verwendet werden kann und wenn ja, mit welcher Durchschnittsgeschwindigkeit. Darüber hinaus werden Geschwindigkeitsbeschränkungen und Einbahnstraßenregelungen erfasst.

- Auch Informationen über die Benennungen von Straßen werden in diesem Schritt ermittelt. Das ist nicht trivial, da OSM viele Varianten für die Namensgebung unterstützt.

- Die Länge von Straßenabschnitten von Kreuzung zu Kreuzung wird berechnet. Die Durchschnittsgeschwindigkeit oder Fahrzeit können an dieser Stelle noch nicht ermittelt werden, da diese von der Export-Konfiguration (insb. dem Fortbewegungsmittel) abhängen.

Die Ausgabe dieses Schritts wird in einer SQL-Datenbank [Ro10a, Ro12c] gespeichert. Tabelle 1 zeigt ein typisches Mengengerüst für Straßendaten in Deutschland.

Tabelle 1: Mengengerüst der Straßendaten für Deutschland (Stand Juni 2012)

	Alle Straßen in Mio.	**Befahrbare Straßen in Mio.**
Kreuzungen	17,7	4,7
Straßenabschnitte (Verbindungen zwischen Kreuzungen)	23,3	10,9
Komplette Straßen (inkl. Namen)	6,1	2,3

2.2 Generieren von Laufzeit-Daten und -Code

Gemäß der Export-Konfiguration wird aus den Navigationsdaten für ein spezielles Anwendungsszenario eine Datensammlung erzeugt. Die Export-Konfiguration kann folgendes festlegen:

- Mit welchem Fahrzeug (oder mit welchen Fahrzeugen) soll die Routenplanung später erfolgen? Hiermit wird gefiltert, welche Straßentypen überhaupt exportiert werden müssen. Es werden auch nur noch Kreuzungsobjekte zwischen relevanten Straßen exportiert. Zusätzlich werden Geschwindigkeitsprofile eingetragen. Für jeden Straßenabschnitt von Kreuzung zu Kreuzung wird insbesondere berechnet, welche Zeit zum Durchfahren benötigt wird.

- Sollen Abbiegekosten berücksichtigt werden? Ist das der Fall, müssen Kreuzungskosten vorberechnet werden, indem z.B. alle möglichen Abbiegewinkel berechnet werden.

- Bestimmte Heuristiken der Wegeplanung mit A* benötigen eine Vorberechnung, z.B. bei der Verwendung so genannter ALT-Potenziale (siehe später).

- Zum schnellen späteren Zugriff müssen zahlreiche Indizes, insb. räumliche Indizes berechnet werden [Ro09, Ro11b]. Beim Export in eine Smart-Phone-Umgebung kommen beispielsweise so genannte Spatial Hashtables zum Einsatz [Ro12b]. Damit diese effizient arbeiten, müssen alle Einträge zunächst so sortiert werden, dass räumlich nahe Einträge auch im Speicher nah beieinander liegen (Lokalisierung).

Passend zu den Daten wird Laufzeit-Code exportiert. Hierzu liegen die entsprechenden Komponenten schon in verschiedenen Export-Bibliotheken (z.B. Android, Server) vor. Zusätzlich steuert die Export-Konfiguration die Einbindung bestimmter Klassen, z.B. für den Zugriffsmechanismus auf die Laufzeit-Daten. Beim Start des Laufzeit-Codes wird überprüft, ob der Code zu den Laufzeitdaten passt. Gegebenfalls wird der Hochlauf mit einem Fehlercode beendet.

3 Laufzeitfunktionen von donavio

3.1 Die Basiskomponenten von donavio

Neben der reinen Wegeplanung gibt es im Rahmen einer Navigationsanwendung viele Funktionen. Eine Übersicht über alle relevanten Funktionen zeigt Abb. 2.

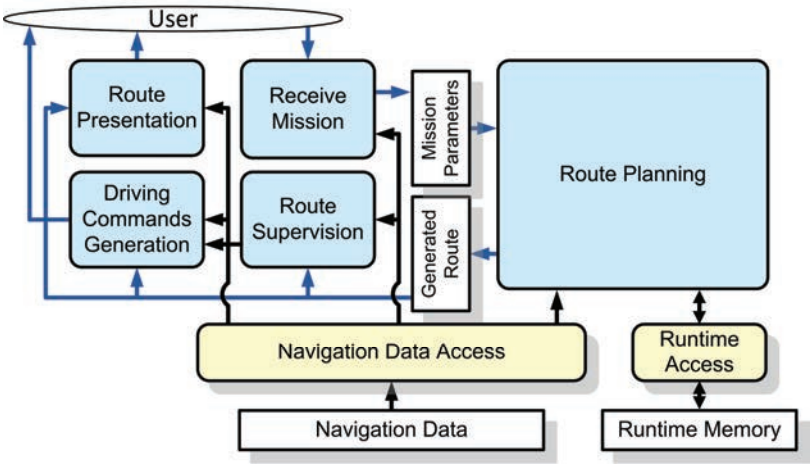


Abbildung 2: Übersicht über donavio-Laufzeitfunktionen

Der Zugriff auf die Navigationsdaten kann über verschiedene Verfahren erfolgen, die in den Access-Schichten gekapselt werden. Die funktionalen Komponenten sind:

- *Receive Mission*: Ein Navigationsauftrag muss entgegengenommen werden. Das erfordert die Suche nach Start- und Endpunkt über symbolische Namen, Adressen, Karten oder geographische Positionen.
- *Route Planning*: Ein Weg soll gemäß verschiedener Optimalitätskriterien und Fortbewegungsprofilen berechnet werden. Die Berechnung muss angemessen schnell erfolgen.
- *Route Supervision*: Schließlich muss die Fahrt überwacht werden. Es muss ermittelt werden, wann Abbiegekommandos präsentiert werden und wann eine Neuberechnung der Route veranlasst werden muss.

- *Driving Commands Generation*: Der gefundene Weg muss auf Abbiegekommandos (z.B. "In 100 Metern rechts abbiegen") abgebildet werden (Abb. 3 links oben). Das ist keineswegs trivial, da die Wahrnehmung markanter Punkte durch Menschen sich signifikant vom topologischen Straßennetz für die Routenplanung unterscheidet.

- *Route Presentation*: Zusätzlich soll der Weg meistens auf einer Karte dargestellt werden (Abb. 3 rechts unten). Das umfasst das Darstellen eines Kartenhintergrundes (u.U. gedreht) und die übersichtliche Darstellung der abzufahrenden Route. Für die aktuelle Realisierung dieser Komponenten verwenden wir die selbstentwickelte *dorenda*-Umgebung [Ro11a], die gegenüber Google Maps einige Vorteile hat.

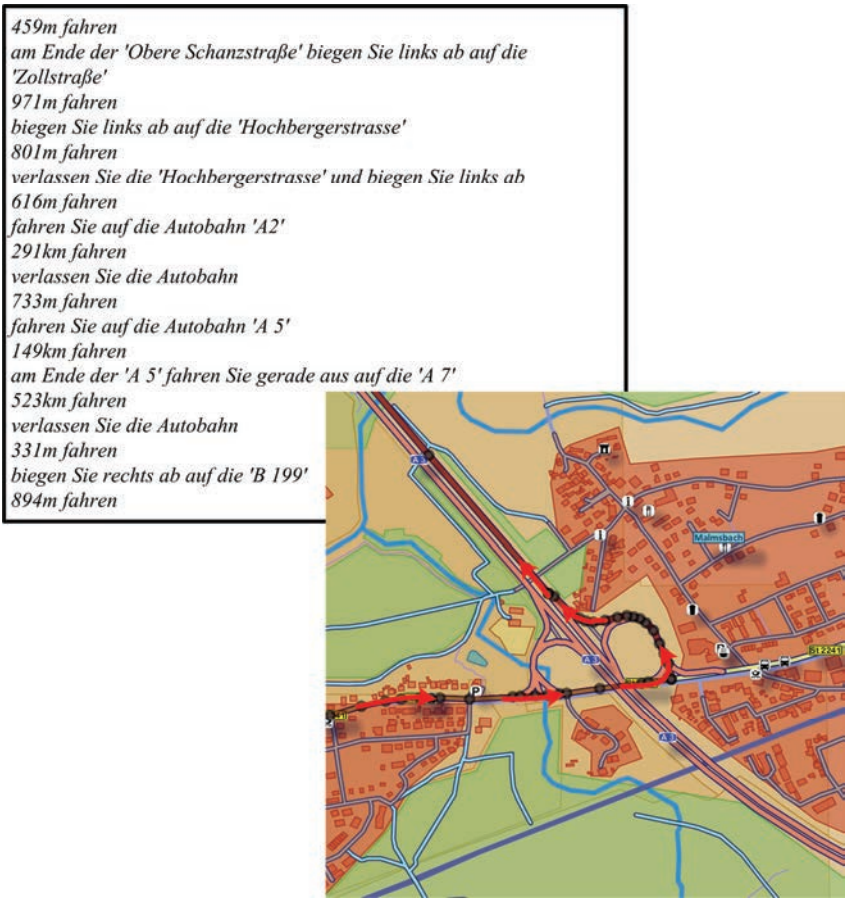


Abbildung 3: Beispiel einer donavio-Ausgabe

Zu den fünf Komponenten aus Abb. 2 stellt donavio jeweils eine standardisierte Schnittstelle zur Verfügung. Damit sind diese Komponenten modular auswechselbar. neben einer Aufrufschnittstelle über ein Class Interface gibt es eine Web-Service-Schnittstelle. Damit können einzelne Komponenten in einer Client-Server-Umgebung ausgelagert werden. Die Ausnahme hierzu bildet die Komponente *Route Supervision*. Da diese relativ zeitkritisch ist, muss sie auf dem jeweiligen Endgerät ausgeführt werden.

3.2 Die donavio-Zugriffsmodule

Neben der algorithmischen Realisierung der fünf Basiskomponenten hat der Zugriff auf die Daten und die Verwaltung der Laufzeitdaten einen großen Einfluss auf das Laufzeitverhalten. Eine Design-Entscheidung von donavio war, die Zugriffsmodule vollständig von den Basiskomponenten zu trennen. Es gibt zwei relevante Module: *Navigation Data Access* und *Runtime Access*.

Navigation Data Access

Dieses Modul bietet Zugriffsfunktionen auf Kreuzungen, Straßenabschnitte und komplette Straßen. Kreuzungen und Straßenabschnitte beschreiben dabei die Topologie für die Routenplanung. Die kompletten Straßen beinhalten die Straßengeometrie für die Darstellung auf der Karte und symbolische Informationen für die Suche. Das Zugriffsmodul unterstützt den Zugriff per Objekt-ID sowie die räumliche und symbolische Suche. Zurzeit werden zwei Realisierungen für dieses Modul angeboten:

- *Memory Files*: Beim Programmstart werden alle Daten in den Objekt-Speicher geladen. Hierzu wurde vorher eine Datei generiert, die alle Objektstrukturen enthält. Einmal geladen, kann der Zugriff auf alle Daten mit maximaler Geschwindigkeit erfolgen, allerdings ist der Speicherbedarf zur Laufzeit sehr groß. Diese Variante eignet sich für Server, die mit entsprechendem Hauptspeicher ausgestattet sind.

- *Spatial Hashtables*: Hier wird nur der jeweils benötigte Bereich der Tabellen aus einem Sekundärspeicher in den Hauptspeicher eingelagert. Das Verfahren ähnelt dem virtuellen Speicher, allerdings wird die räumliche Lage der Einträge berücksichtigt. Damit beim Einlagern direkt viele Einträge in der näheren Umgebung vorliegen, werden die Tabellen vorher *lokalisiert* [Ro12b]. Dieses Zugriffsmodul ist für Smart-Phones geeignet, die über verhältnismäßig wenig Hauptspeicher verfügen.

Ein drittes Modul, das den Zugriff über SQL auf eine Datenbank unterstützt, wurde zwar implementiert, wird aber aufgrund der sehr schlechten Performance nicht eingesetzt.

Runtime Access

Einen großen Einfluss auf die Laufzeit und den Speicherbedarf hat die Verwaltung der Strukturen, die A* während des Ablaufs benötigt. Hierbei handelt es sich um den Knotenstatus (*OPEN* oder *CLOSED*) sowie um die Knotenwerte *f* und *g* [HNR68]. Prinzipiell könnten diese Informationen in dem topologischen Netzwerk gespeichert werden. Beim Zugriff über Spatial Hashtables liegt das topologische Netzwerk jedoch auf dem externen Speicher, kommt damit als Speicherort für die Verwaltungsstrukturen von A* aus Zeitgründen nicht in Frage. Daher werden die Verwaltungsstrukturen separate abgelegt und zu jedem Eintrag wird eine Referenz auf den Knoteneintrag in der Topologie verwaltet.

Bei einem typischen Suchlauf werden einige 10 000 bis mehrere Million Knoten besucht, dabei hat die Liste der offenen Knoten meist nicht mehr als 10 000 Einträge. Die donavio-Umgebung unterstützt derzeit folgende Zugriffsverfahren:

- *Arrays*: Für alle A*-Eigenschaften werden Arrays von der Größe der gesamten Knotenliste angelegt. Diese Arrays sind in der Regel viel zu groß, da nur ein Teil der Knoten durch A* besucht wird. Es wird aber dadurch ein sehr schneller Zugriff ermöglicht. Darüber hinaus bleibt der Speicherbedarf zur Laufzeit konstant, d.h. es geht keine Zeit für die Garbage Collection verloren. Diese Variante ist vor allem für Server geeignet.

- *DynArray*: Es wird ein einzelnes Objekt-Array von der Größe der gesamten Knotenliste angelegt. Die Referenzen in diesem Array sind zuerst alle unbelegt. Erst wenn ein Knoten durch A* besucht wurde, wird

ein Objekt-Eintrag angelegt. Der Speicherbedarf ist deutlich niedriger als bei *Arrays*, allerdings entsteht ein zusätzlicher Aufwand durch das Anlegen und Freigeben der Objekte.

- *Hashtable*: Hier werden Hashtable-Einträge mit der Knotenreferenz als *key* und einem Objekt aller Verwaltungsinformationen als *value* gespeichert. Es entsteht ein Suchaufwand zur Laufzeit, da ein Knoten erst in der Hashtable gesucht werden muss. Der Speicherbedarf kann jedoch nie größer als die Liste der besuchten Knoten werden. Diese Variante ist selbst auf Endgeräten mit wenig Hauptspeicher lauffähig.

Abb. 4 illustriert noch einmal die verschiedenen Varianten.

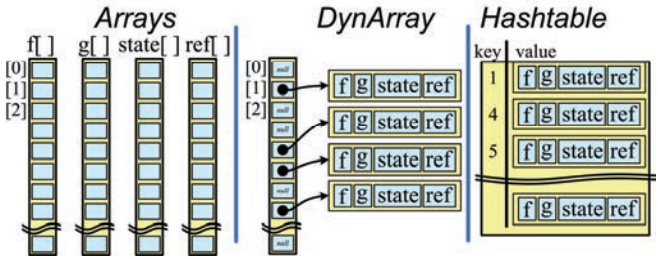


Abbildung 4: Zugriffsverfahren für die A*-Laufzeitstrukturen

3.3 Die donavio-Routenplanung

Die meisten Möglichkeiten für verschiedene Konfigurationen bietet die Komponente *Route Planning*. Daher ist diese in Abb. 5 weiter aufgeschlüsselt. Die donavio-Umgebung unterstützt derzeit verschiedene Module, die den A*-Algorithmus modifizieren oder erweitern. Insbesondere die Heuristic-Module haben im Zusammenhang mit der Laufzeit eine besondere Rolle. Ziel ist, durch geeignete Verfahren den Suchaufwand so zu reduzieren, dass die Routenplanung selbst auf leistungsschwachen Plattformen nur einige Sekunden benötigt. Dabei nimmt man suboptimale Routen in Kauf.

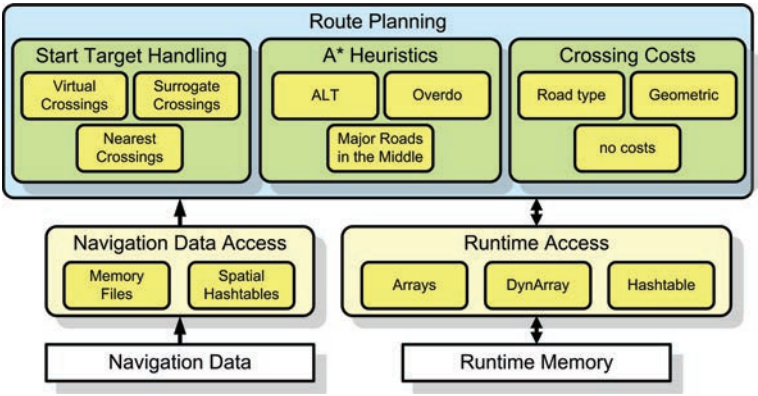


Abbildung 5: Übersicht über Funktionen der Routenplanung

Für die inneren Komponenten gibt es jeweils mehrere Möglichkeiten. Im Rahmen der Code-Bereitstellung müssen, je nach Export-Konfiguration, nicht alle Komponenten bereitgestellt werden. So kann eine konkrete Variante nur eine einzelne Sub-Komponente von A* Heuristics enthalten. Wird in der Laufzeit-Konfiguration (Abb. 1a) eine nicht existente Variante gewählt, gibt es zur Laufzeit einen Fehler.

A* Heuristics

In der Literatur werden verschiedene Verfahren zur Optimierung von A* im Kontext der Wegeplanung vorgeschlagen. Zurzeit sind in donavio die folgenden Verfahren integriert:

- *overdo*: es wird eine nicht-optimistische Schätzung verwendet, indem die optimistische (Luftlinien-) Schätzung mit einem konstanten Faktor multipliziert wird. Hiermit wird die Schätzung in der Regel realistischer, im Grenzfall aber pessimistisch. Daher sind suboptimale Resultate möglich.

- *ALT* (*A* search, landmarks, and triangle inequality*): Zu einer kleinen Anzahl ausgewählter Landmarks werden die exakten Wege-Kosten für allen Knoten von und zu diesen Landmarks berechnet, die so genannten ALT-Potenziale [GH05]. Für die Gesamtkosten eines Knotens zu einem bestimmten Ziel ist die maximale Differenz der Potenzialwerte eine sehr gute optimistische Schätzung. Diese Variante erfordert, dass bei der Bereitstellung der Daten die ALT-Potenziale vorberechnet und exportiert werden.

- *Major Roads in the Middle*: A* verfolgt bei Überschreitung einer Entfernung vom Start oder Ziel nur noch große Straßen [SP05]. Das entspricht der typischen Erfahrung bei langen Fahrten: im "mittleren" Segment einer Route verwendet man typischerweise nur Autobahnen oder Landstraßen. Optimale Routen, die in der Routenmitte kleine Straßen verwenden, werden so aber nicht erkannt.

Weitere Module sind denkbar und können bei Bedarf integriert werden.

Start Target Handling

Oft wird ein Problem vernachlässigt: A* kann zwar eine optimale Route von Knoten zu Knoten (also von Kreuzung zu Kreuzung) berechnen – typischerweise liegen Start und Ziel aber nicht auf einer Kreuzung. Es werden derzeit folgende Varianten zur Behandlung des Problems unterstützt:

- *Nearest Crossings*: Start und Ziel werden auf die nächste *erreichbare* Kreuzung abgebildet. Dieses Verfahren löst das Problem nicht eigentlich, wird daher nur als letzte Möglichkeit angeboten.

- *Virtual Crossings*: Start und Ziel werden als zusätzliche Knoten in das Straßennetz eingefügt. Zusätzlich müssen diese Knoten über neue Kanten mit den anteiligen Kosten mit dem Straßennetz verbunden werden. Diese Variante kann nur eingesetzt werden, wenn das Zugriffsmodul auf die Navigationsdaten eine Erweiterung der Topologie erlaubt. Spatial Hashtables erlauben dies nicht, da die Daten optimiert exportiert wurden und nicht verändert werden können.

- *Surrogate Crossings*: Start und Ziel werden zunächst jeweils auf die nächste *erreichbare* Kreuzung abgebildet (genannt *Surrogate* Crossing). A* plant dann von Surrogate zu Surrogate. Am Ende wird eine Gesamtroute durch *Start*→*Surrogate*, *Surrogate*→*Surrogate*, *Surrogate*→*Ziel* zusammengesetzt. Hierbei müssen ggfs. doppelte Wege abgezogen werden. Es entstehen zahlreiche Sonderfälle, wenn beispielsweise Start und Ziel auf Einbahnstraßen liegen oder gar auf derselben Straße. Darüber hinaus können suboptimale Wege entstehen, wenn eine theoretisch optimale Route nicht beide Surrogate Crossings enthält.

Das Verfahren Surrogate Crossings ist noch einmal in Abb. 6 illustriert.

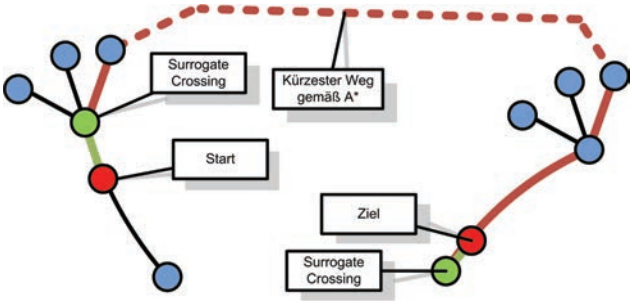


Abbildung 6: Das Surrogate-Crossing-Verfahren

Crossing Costs

In der Realität kosten Abbiegevorgänge Zeit. Eine Wegesuche in Graphen berücksichtigt aber erst einmal nur Kantenkosten. Wünschenswert sind daher Verfahren, die auch die Kosten von Abbiegungen berücksichtigt. Es werden derzeit folgende Varianten unterstützt:

- *no costs*: Die Zeit für das Abbiegen wird nicht berücksichtigt.
- *Road Type*: Es werden Kosten geschätzt, die daraus resultieren, dass man eine Straße verlässt und auf eine andere Straße fährt. Die Kosten berücksichtigen dabei nur den Übergang der Straßentypen (z.B. Autobahn→Ausfahrt).
- *Geometric*: Zusätzlich zu den geschätzten Kosten durch den Übergang zwischen Straßen wird der Abbiegewinkel berücksichtigt. Für scharfes Abbiegen wird daher eine längere Fahrzeit erwartet. Dieses Modul erfordert, dass bei der Bereitstellung der Navigationsdaten die Winkel berechnet werden, unter der eine Straße auf eine Kreuzung trifft.

Auch hier sind weitere Module denkbar. So könnte ermittelt werden, ob eine Kreuzung eine Ampel besitzt, oder ob man über eine Vorfahrt-Achten-Straße einmündet.

4 Die softwareseitige Unterstützung der Modularisierung

Ziel der Modularisierung ist, dass verschiedene Varianten ausprobiert werden können und dass verschiedene Zerlegungen zwischen Client und Server eingesetzt werden können. Daher erfüllen alle Basismodule (Abb. 2) eine objektorientierte Aufrufschnittstelle und haben (bis auf das Modul *Route Supervision*) zusätzlich eine Web-Service-Schnittstelle.

Die inneren Module von *Route Planning* sowie die Zugriffsmodule werden nur lokal aufgerufen, deshalb wurden keine Web-Service-Schnittstellen vorgesehen. Man kann hier zwei Arten von Modulen unterschieden:

1. *Start Target Handling* und *Crossing Costs*: Diese binden sich über eine klassische objektorientierte Aufrufschnittstelle in die Routenplanung ein. In den Laufzeit-Code werden alle Klassen integriert, die gemäß der Export-Konfiguration später notwendig sind. Wurde in der Export-Konfiguration beispielsweise die Vorberechnung von Kreuzungswinkeln deaktiviert, wird für das Modul *Geometric Crossing Costs* kein Code exportiert. Die eigentliche Auswahl zur Laufzeit wird über den Aufruf der Routenplanung definiert.
2. *Zugriffmodule* und *A* Heuristics*: Diese Module zeichnen sich durch folgende Eigenschaften aus:

- Sie werden extrem häufig aufgerufen (viele Millionen Mal), der eigentlich ausgeführte Code ist demgegenüber aber sehr klein.

- Eine Modellierung über eine Objektschnittstelle würde einen erheblichen Laufzeitaufwand produzieren. Letztlich kann man zwar jede Funktionalität über ein System von Klassen darstellen, der Code muss aber entsprechend zerlegt werden. Darüber hinaus produzieren die Parameterübergabe, Rückgabewerte und der Objektaufruf selbst einen gewissen Aufwand.

Da als Zielplattformen auch solche mit geringerer Prozessorleistung in Frage kommen sollen, werden diese Module über *Macro-Preprocessing* eingebunden, d.h. konkret:

- Es gibt Code, der außerhalb einer Klasse als Code-Fragment textuell vorliegt.

- In der Routenplanung werden diese Code-Fragmente je nach Modul eingeblendet. Die Parameterübergabe findet zur Compile-Zeit statt. Da Code kopiert wird, gibt es formal auch keinen Aufruf zur Laufzeit.

Als Beispiel: Die Variante *Arrays* des Moduls *Runtime Access* greift auf die Knoten-Eigenschaften über Felder zu, z.B. mit **`g[index]`** auf die bisherigen Kosten vom Start zum Knoten. Selbstverständlich könnte dieser Aufruf über eine Objektschnittstelle als Methode

```
public float getG(int index) { return g[index]; }
```

modelliert werden. Damit würde aber für jeden Aufruf zusätzlicher Code ausgeführt werden. Über Macro-Preprocessing wird der Array-Zugriff zur Compile-Zeit an die entsprechenden Stellen kopiert. Noch größer ist der Vorteil bei der Runtime Access-Variante *Hashtable*. Hier zerfällt der Zugriff auf den Knotenstatus in zwei Teile:

- Über **`astar_state=astar_hash.get(index)`** beschafft man sich die Referenz auf das Knotenobjekt.

- Über **`astar_state.g`**, **`astar_state.f`** etc. greift man auf die Eigenschaften zu.

Bei einer klassischen objektorientierten Zerlegung würde man innerhalb der Methoden **`getG`**, **`getF`** das Knotenobjekt **`astar_state`** immer wieder aus der Hashtable lesen. Wenn man die Reihenfolge der Zugriffe nicht kennt, könnte das Zugriffsobjekt sonst nicht gewährleisten, dass man schon vorher das richtige Statusobjekt ausgelesen hat. Stehen hinreichende Berechnungsressourcen zur Verfügung, nimmt man solche Mehrfachzugriffe in Kauf, wenn man dadurch eine robuste Verwendung gewährleisten kann. In zeitkritischen Umgebungen kann man sich unnütze Zugriffe nicht erlauben. Daher wird über Macro-Preprocessing die richtige und redundanzfreie Verwendung in den Code eingebettet.

Die donavio-Plattform ist komplett in Java entwickelt und Java kennt von Hause aus kein Macro-Preprocessing. Es kommt im Java-Design bewusst nicht vor und die Verwendung von Macros wird kontrovers diskutiert. Das Argument dagegen ist, dass man praktisch alles, was als Macros denkbar ist, auch über ein Klassensystem modellieren kann. Darüber hinaus kann man durch falschen Gebrauch von Macros die Wartung von Code signifikant erschweren. Die Entscheidung fiel dennoch zugunsten der Einbindung durch Macros aus, da der resultierende Laufzeit-Code nur noch den Code der gewählten Variante enthält, damit sehr schlank und effizient ist.

Zurzeit kann Code für die Java Standard Edition sowie für die Android-Umgebung generiert werden. Abb. 7 zeigt den Prototyp einer Android-Navigationsanwendung.

Abbildung 7: Android-Prototyp basierend auf donavio

5 Zusammenfassung

Die donavio-Umgebung stellt eine vollständige und leistungsfähige Plattform für die Generierung und Ausführung von Routen-Planungsanwendungen dar. Hierbei sind verschiedene Architekturen und Einsatzumgebungen denkbar.

Durch den modularen Aufbau können auf einfache Weise neue Verfahren getestet werden. Auch der Einsatz in Lehrveranstaltungen bietet sich an. Bisher gab es das Problem, dass man z.B. Varianten von A* in Lehrveranstaltungen realisieren konnte, diese waren aber nicht isoliert lauffähig, da die Zugriff auf Topologiedaten und die Präsentation der Routen eine Lehrveranstaltung überfrachtet hätten. Mit donavio können bestimmte Komponenten in Praktika realisiert und in der gesamten Navigationsanwendung zum Einsatz gebracht werden.

Literaturverzeichnis

[GH05] Goldberg, A. V.; Harrelson, C.: 2005, Computing the Shortest Path: A* Search Meets Graph Theory. In Proc. 16th ACM-SIAM Symposium on Discrete Algorithms, 2005, 156–165

[HNR68] Hart P. E., Nilsson N. J., Raphael B. 1968, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics SSC4 (2), 100–107

[NZ08] Neis, P.; Zipf, A.: LBS_2.0 - Realisierung von Location Based Services mit user-generated, collaborative erhobenen freien Geodata, in: J. Roth (Hrsg.): 5. GI/ITG KuVS Fachgespräch Ortsbezogene Anwendungen und Dienste, 4.-5. September 2008, Nürnberg. Schriftenreihe der Georg-Simon-Ohm-Hochschule Nürnberg Nr. 42, 111-115

[OSM] Open Street Map, http://www.openstreetmap.org/

[Ro09] Roth, J.: The Extended Split Index to Efficiently Store and Retrieve Spatial Data With Standard Databases, IADIS International Conference Applied Computing 2009, Rom (Italien), 19.-21. Nov. 2009, Vol. I, 2009, 85-92

[Ro10a] Roth, J.: Die HomeRun-Plattform für ortsbezogene Dienste außerhalb des Massenmarktes, in Zipf A., Lanig S., Bauer M. (Hrsg.) 6. GI/ITG KuVS Fachgespräch "Ortsbezogene Anwendungen und Dienste", Heidelberger Geographische Bausteine Heft 18, 2010, 1-9

[Ro10b] Roth, J.: Übernahme von Geodatenbeständen aus Open Street Map und Bereitstellung einer effizienten Zugriffsmöglichkeit für ortsbezogene Dienste, Praxis der Informationsverarbeitung und Kommunikation (PIK), Vol. 13, Heft 4, 2010, 268-277

[Ro11a] Roth, J.: Der Einsatz von OpenStreetMap-Daten in der akademischen Informatik-Ausbildung, in Küpper A., Roth J. (Hrsg.): 7. GI/ITG KuVS Fachgespräch "Ortsbezogene Anwendungen und Dienste", Logos-Verlag, 2011, 65-72

[Ro11b] Roth, J.: Moving Geo Databases to Smart Phones – An Approach for Offline Location-based Applications, Innovative Internet Computing Systems (I2CS), Berlin, 15.-17. Juni 2011, GI Lecture Notes in Informatics, Vol. P-186, 2011, 228-238

[Ro12a] Roth J. 2012, Managing Geo Data – Usage of Open Street Map for Own Services and Applications, Tutorial on the TPMC 2012, Lisbon, available at http://www.wireless-earth.org/homerun.html

[Ro12b] Roth, J.: A Spatial Hashtable Optimized for Mobile Storage on Smart Phones, 9. GI/ITG KuVS Fachgespräch "Ortsbezogene Anwendungen und Dienste", 2012

[Ro12c] Roth, J.: Geo-Tabellen im HomeRun-Projekt, Internes technisches Papier, Ohm-Hochschule Nürnberg, 2012

[SP05] Sanders, P.; Schultes, D.: Highway hierarchies hasten exact shortest path queries, in Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS 3669, Springer, Heidelberg, 2005, 568–579

Testing Sensor Fusion Algorithms in Indoor Positioning Scenarios

Moritz Kessel, Marco Maier, Mirco Schönfeld and Florian Dorfmeister

Mobile and Distributed Systems Group
Ludwig-Maximilians-University Munich
Oettingenstr. 67
80538 München
vorname.nachname@ifi.lmu.de

Abstract: Location information is the foundation for location based services. However, a cheap and global indoor positioning solution offering a sufficiently high accuracy and precision for most applications is not yet available and detains location based services from indoor areas. This is also due to missing comparability and standards in the field of indoor positioning, resulting in a large number of proprietary research prototypes of varying capabilities. In this paper we present a software environment for testing and comparing sensor fusion algorithms in indoor positioning scenarios. For this purpose an extendable tool for evaluating accuracy, precision, robustness, complexity, and storage is designed and the results from a first implementation presented and discussed.

1 Introduction

Indoor positioning is of increasing importance for the application area of location-based services since it is essential for the provisioning of indoor location-based services (ILBS). While a large number of outdoor services is available today to interested users, ILBS are only beginning to leave the research labs and only few have arrived at the market. This is mostly due to the lack of a cheap and accurate positioning technology in indoor environments. While GPS offers cheap and global positioning with an accuracy of few meters, it is only available in outdoor areas due to the strong attenuation and multipath effects inside buildings. Furthermore, most ILBS have even stronger accuracy requirements that can only be fulfilled at reasonable expenses by the combination of multiple sensors integrated in user devices such as smartphones. A vertical accuracy of less than 3 meters is needed to provide reliable distinction between several floors of a building. The required horizontal accuracy might even be higher to allow the differentiation between neighboring rooms.

In the past, many different approaches for indoor positioning have been developed. Some systems based on dedicated hardware such as Ubisense [SG05] allow for submeter accuracy in three dimensions. While their accuracy is sufficiently high for ILBS, those systems are often only available in a limited area due to the large cost induced by the positioning infrastructure. Single technology approaches with existing hardware and infrastructure, mostly based on WLAN [BP00] or inertial sensors [WH08] have to face intrinsic limitations induced by the choice of technology. While WLAN based systems have the advantage of often existent infrastructure, they also require a tremendous amount of calibration time to operate with sufficient accuracy and are prone to jumps in consecutive position estimations. Inertial sensors avoid the jumping and need no time consuming calibration, but their absolute positioning accuracy decreases over time, since only relative position changes are measured and the error thus accumulates over time. Sensor fusion mechanisms provide a solution here, since they allow for the combination of several sensors in a way that can minimize the disadvantages and maximize the strengths.

However, most algorithms for the combination of sensor data, also called sensor fusion, lack comparability. Algorithms are implemented and tested at different environments, with different data sets, and different hardware. We present a tool for the evaluation of different algorithms with various data sets from several environments with respect to accuracy, precision, robustness, calculation speed, and complexity. Thus it becomes possible to compare several algorithms and identify their strength and weakness concerning varying environments, hardware, and data sets.

The tool consists of three components. The first component runs on a mobile phone and offers capabilities for

logging of sensor data and manually inserting a special timestamp. The second component is a model provider offering environmental information such as floorplans or fingerprint databases. The last and most essential component is the evaluation tool. The tool can import logfiles from the first component and import environmental information from the model provider. Furthermore, ground truth can be included for comparison with estimated positions and calculation of the positioning error. The tool offers visual output in form of a map on which a step by step simulation of positioning can be carried out. Step by step means that the tool allows the sequential processing of each single sensor measurement in the logfile and simulation means that the result of processing a measurement can be displayed in real time. A researcher willing to compare or analyze his sensor fusion algorithm has only extend the algorithm to implement an interface and can then access the full functionality from processing measurements, displaying immediate results, and calculating the positioning error. Since the sensor fusion in decoupled from the data, results for data sets from various hardware and different environments can be used to compare and evaluate multiple fusion algorithms.

The rest of the paper is structured as follows: In the next section requirements for indoor positioning systems for location based services are defined and characterized. Then follows an overview on existing technologies and positioning systems. Section 4 introduces the tool and its components, while Section 5 shows the feasibility of the tool by comparing the capabilities of a Kalman and a particle filter concerning the requirements with the help of the tool. The paper is then concluded and finalized with hints on future research.

2 Requirements

Indoor location based services have special requirements for the underlying indoor positioning technology. One of the more crucial requirements is the capability to calculate the user's position in real time. Since the location of a user is used to generate value added content and filter relevant information, it is desirable that the changes in position information are discovered within few seconds. No-one would like a navigation system which lags some seconds behind. The **update time** of position information depends on many factors such as the technology, the sensing capabilities and the processing speed of the positioning device as well as the **complexity** of the positioning algorithm. The latter can be described by the quotient calculation time/measurement time. While the quotient is smaller than 1, the system is able to calculate positions faster than the time between measurements and thus has real time capabilities. In this case, the update rate depends only on the sensor data rate.

Another requirement is a high **accuracy** and **precision** of the positioning system. The accuracy is often given in for of the mean deviation of the position estimation from the real position, the ground truth, while the precision is the standard deviation of the real position and an assessment of the stability of the positioning system. For many indoor location based services an accuracy of one to two meters with a high precision of less than a meter is desirable. A coarse estimation of the **robustness** of a system can be given in form of the maximal position error. All these errors can not be given for all environments, but the errors of different positioning systems can be compared in a single environment.

In addition, the **memory requirements** of a positioning system influence the choice of mobile terminals and the component where the calculations are executed. This could either be terminal-based on the mobile terminal alone or network-based in an infrastructure.

Of course are there other factors such as the cost, scalability, energy consumption, market maturity, or availability, but they are hard to measure and subject to many influences. However, for indoor location based services it is crucial to obtain the position information of pedestrians and thus we propose to utilize smartphones and their available sensing technologies for position estimation.

3 Smartphone Positioning and Sensor Fusion

In the following paragraphs, related work is grouped by positioning methods, technologies and systems. The specification of a test environment for sensor fusion algorithms similar to our approach has not been presented in

literature so far, even if most researchers obviously rely on some evaluation tool for testing their algorithms. Those tools however are far from standardization, not publicly available, and probably not easily extendable for other algorithms or data sets.

3.1 WLAN Positioning

WLAN positioning offers some advantages for positioning on smartphones: the infrastructure is often already deployed in form of a communication network or can be added without to much financial effort. Existing approaches [KW11, MVFB10, CBM10] offer real time localization on smartphones without any calculational effort in some infrastructure. The best results concerning robustness, accuracy, and precision are achieved by fingerprinting approaches [BP00, YA05]. These are based on measuring reference data, i.e. signal strength and MAC addresses from surrounding access points, in a deployment phase at certain positions. When the reference database is built, position estimation can be carried out by pattern matching the current measurement with the stored information. The more similar the current measurement to a reference data, the more probable is the device situated at the position the data was recorded. The best performance within fingerprinting approaches is obtained by probabilistic methods working with conditional probabilities and Bayes rule. But even those only offer the required accuracy and precision under optimal circumstances and are prone to jump between consecutive measurements. Furthermore, the calibration of such a system is time consuming and needs to be carried out in case of environmental changes.

3.2 Camera-based Positioning

Optical methods for position estimation on a smartphone can offer a high accuracy and precision [WKM11] well below a meter. The expenses are small and the positioning stable. There are some approaches to carry out camera-based positioning on smartphones [HB07, MWSB09, WKM11]. Some positioning systems rely on pattern matching similar to WLAN [WKM11]. In this case, scale and rotation invariant and distinctive features [BTVG08, Low99] are extracted from a camera image and compared to a database of previously recorded images which position is known. Instead of natural features, also artificial markers can be utilized for encoding position information in the visual environment. Depending on size and camera resolution the calculational effort is extremely high and the position estimation still has to be carried out in a powerful infrastructure. Even then response times of several seconds can occur which might be to slow for many indoor location based services.

3.3 Inertial Sensors for Positioning

Most modern smartphones have a number of sensors built in which can be utilized for position estimation without external infrastructure. For movement detection an accelerometer can offer acceleration measurements which can be further processed for velocity estimation or step detection. The direction of movement can be measured by a compass or deduced from gyroscope readings. In addition, barometers can provide hints on the vertical position. However, these inertial measurement units do not provide an absolute position but more the relative change in position and are therefore only suitable for dead reckoning. When an initial position is available consecutive changes and thus the current position can be estimated. Unfortunately, small errors in relative positioning due to inaccurate and cheap sensors accumulate over time and detain dead reckoning from fulfilling the requirements on a positioning system over longer periods. Possible compensation can be achieved through map matching [WH08] or sensor fusion techniques [AMGC02]. Existing approaches often utilize step detection [LSVW11] and estimate the steplength from step frequency and body height [GIK10]. To compensate for the accumulating error, most positioning systems based on inertial sensors utilize map matching techniques. These allow for a plausibility check of estimated positions by checking the walkability and connectivity between consecutive positions [SKC11]. Especially probabilistic multi-hypothesis approaches such as particle filters are well suited for map matching.

3.4 Sensor Fusion for Combining Several Sensors

Every affordable and on the smartphone available technology has its disadvantages and weaknesses when used for position estimation. So there is at the moment no single technology system fulfilling all the mentioned requirements for indoor location based services. Many approaches therefore rely on the combination of different technologies which is also known under the term sensor fusion. There are two major algorithms utilized for this task: the Kalman filter and the particle filter [AMGC02]. While the Kalman filter generally has a lower computational complexity it is restricted to a single hypothesis, Gaussian distributed measurements, and linear systems. The particle filter on contrary estimates the position with a discretization of an arbitrary probability distribution, possibly with multiple modes or clusters. In [EM06] a combination of WLAN position estimation and inertial sensors is presented and the performance of a Kalman filter and a particle filter compared. The evaluation shows that particle filters tend to have a better performance concerning accuracy, precision and robustness, but has a higher computational complexity.

4 Designing an Evaluation Toolkit

In the previous sections, requirements for positioning systems were presented followed by several systems grouped by technology. The overview is coarse and every month new positioning systems are developed to reach the goal of a global indoor positioning solution bringing location based services to indoor environments. Nevertheless, there is no general agreement on the term which positioning system is better than another one. No widely available benchmarks exists, not even a standard metric for measuring the performance. To bridge these gaps and to ease the development and evaluation of new indoor positioning algorithms without tiresome gathering of data, provision of realistic test environments, and designing an evaluation software, we propose a framework for easily testing sensor fusion algorithms.

The framework should provide means for easily gathering synchronized test data from multiple smartphone sensors, evaluating several algorithms on the same data set, visually presenting the results, and changing parameters of single algorithms. The tasks are divided between three software components that are also displayed in Figure 1.

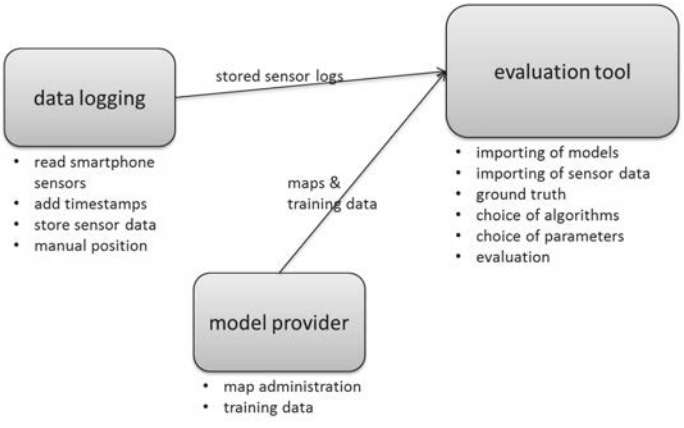


Figure 1: The three software components of the evaluation toolkit and their tasks.

4.1 Data Logging

The task of reading sensor data from a smartphone is accomplished by the data logging component which runs on the smartphone. Depending on the type and equipment of the smartphone, different sensor information can be gathered at varying resolution concerning measurement quantity and quality. A modern phone is often equipped with GPS module, GSM module, WLAN card, Bluetooth chip, accelerometer, gyroscope, magnetometer, barometer, microphone, and two video cameras (front and back camera). The data logging component combines the readings most of these sensors, but has not the capabilities of handling video-streams or microphone data. This is due to storage and especially privacy concerns, since it should be easy to share gathered data without violating the rights of any person in a video or audio record. The component listens to all other sensor sources for new values and stores these values enriched with an identifier and the system time in a file whenever available.

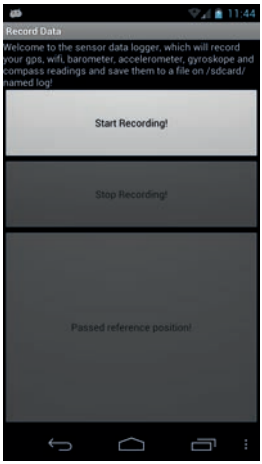


Figure 2: A screenshot of the data logging component currently only available for Android.

Furthermore, the data logging component enables a test person to generate high quality ground truth at recording time with minimal effort. It is crucial for this task that the path the test person follows is known and consists of a sequence of linear line segments. At each junction of two segments a reference position is stored and when the test person crosses such a reference position, a button is pressed to generate a timestamp for crossing the reference position (see Figure 2). Of course is it not possible to always stay exactly at the preset path and some error is induced to the system by manually pressing the button, but without an accurate positioning system as a reference, it is a highly efficient and accurate enough for evaluating the performance of positioning systems when handled with care.

4.2 Model Provider

The model provider is a component with two important tasks: It is responsible for managing map and for managing training data for pattern matching or fingerprinting approaches. Maps are for sake of simplicity and generalization given as a simple bitmap with a pixel grid as reference system. Additionally, each map has certain properties such as the scaling factor given in the pixel-to-meter ratio, a rotation in degrees giving the deviation to plan aligned along the north-south axis, the floor number of the image, and a unique identifier for the building. The bitmap must be conform to some requirements such as a white representation of walkable space and a certain color for

floor transition regions. A region, e.g. a stair or elevator allowing movement to the next floor up is colored green, the possibility of going down is displayed in red, and if both possibilities exist as in the case of stairways the region is colored yellow.

In addition to floorplans, the model provider is responsible for the provision of training data. WLAN fingerprint data can be offered as text documents in a fixed format which is suitable for both Bayesian and k-Nearest-Neighbor fingerprinting. The format allows the specification of a fingerprint with an identifier, a two-dimensional coordinate, an orientation value giving the phone's orientation while recording the fingerprint, and level and building identifier referencing the corresponding map. Each fingerprint has also a number of measurement entries giving the mean received signal strength identificator (RSSI), its standard deviation, and the MAC-address for each access point. The following listing shows the specification and a small example fingerprint:

```
fingerprint: <ID> <x> <y> <level> <map id> <orientation>
measure: <ID> <MAC> <mean RSSI> <standard deviation> <fingerprint ID>
measure: <ID> <MAC> <mean RSSI> <standard deviation> <fingerprint ID>
...

example:

fingerprint: 1 144 234 0 oett 90
measure: 1 00:03:52:ab:82:c4 -92.51 2.71 1
measure: 2 00:03:52:ab:65:a6 -91.63 1.53 1
measure: 3 00:03:52:ab:88:3f -91.45 1.15 1
fingerprint: 2 126 246 0 oett 0
measure: 4 00:03:52:ab:82:c4 -86.32 2.19 2
measure: 5 00:03:52:ab:65:a6 -92.37 1.98 2
measure: 6 00:03:52:ab:88:3f -93.69 1.04 2
```

4.3 Evaluation Tool

The evaluation tool, written in Java, is the heart and the main contribution of this paper. It can import logfiles from the data logging component and import environmental information from the model provider. Furthermore, ground truth can be included for comparison with estimated positions and calculation of the positioning error. This way the mean error, the standard deviation, and the maximal error of a positioning system can be determined. The tool also offers visual output in form of a map on which the output of position estimation is displayed according to the sequence of sensor measurements in real time. Own positioning algorithms can be integrated and their parameters configured. In the following paragraphs, the functionality is described in detail.

For the import, the data needs to be in a specific textual format as generated by the previously described data logging component. This also ensures the synchronization of the data of multiple sensors and enables the manually supported generation of a ground truth track. When map information and data is loaded successfully, the generation of ground truth data can be started. The user needs to click sequentially on the reference positions on the map view to combine the timestamps from the data logger with position information. Once a ground truth is generated it can also be exported in a simple text-based format for later usage.

The tool has several visualization capabilities: Most basically it can display maps, fingerprint data, and ground truth tracks. Furthermore, position estimates, quantiles of multivariate Gaussian distributions, particle clouds, or estimated tracks can be displayed, depending on the positioning algorithm used and the desired visual output. Error vectors can show the deviation from ground truth and evaluation results are displayed in form of a textual output. The user interface also supports the setting of parameters, the choice of algorithms, as well as import and export functionalities. See Figure 3 for an example visualization of a particle filter at runtime.

The calculation of errors of the estimated track with respect to ground truth is achieved by comparing the calculated position to the position on ground truth at the same time. In between reference positions on ground truth, a linear interpolation scheme is applied. This way a small error is accepted in exchange for a higher resolution of the

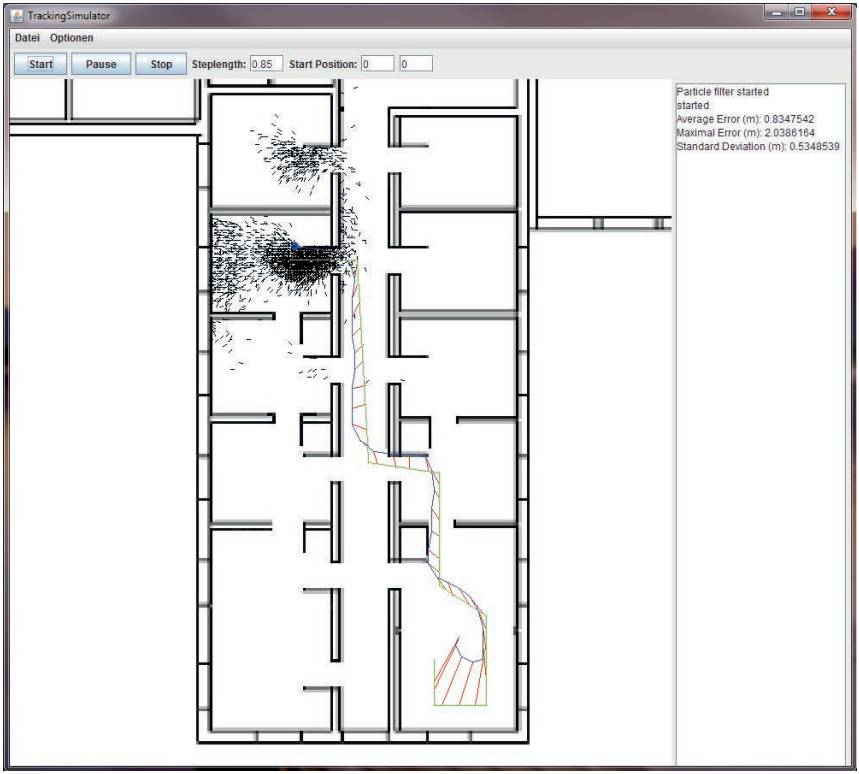


Figure 3: Ground truth (green lines), estimated track (blue lines), error vectors (red lines), and the particle cloud (black dots) displayed on top of the building plan.

position error.

When replaying sensor data of a recorded trace, it is desirable to be able to watch the positioning result in real time and to pause whenever some anomaly or interesting event occurs. So the tool allows to set the replay speed similar to a video and also to pause at arbitrary times.

A new algorithm can easily be added to the tool by extending a class called **AbstractSimulationThread** which handles the communication with the UI and the provision of sensor data and parameters. The researcher only needs to define the parameters and the processing steps of sensor data, returning stepwise results to the tool in one of the predefined forms. This can be either a position, a track segment, a Gaussian distribution or a particle cloud, which is then rendered on screen. One issue remaining at this time is the automatic coupling of parameters to the user interface. At the moment, an additional input needs to be defined for every parameter, causing changes to the UI in case of a new algorithm. This is far from perfect and will be fixed in future versions.

The tool offers a range of new possibilities concerning the evaluation and comparison of different indoor positioning algorithms. One advantage is the independence of the algorithmic design from test data and visualization, allowing the development of new approaches without the need of gathering data or designing one's own evaluation

environment. Another plus is the possibility to provide datasets of sensor data for evaluating algorithms in varying environments and comparing algorithms concerning exactly the same data and environment, providing real comparability between different approaches.

5 Experimental Evaluation

To show the capabilities of the tool, we implemented a Kalman and a particle filter extending the given interface. We then gathered test data in a wing at our university building, where a bitmap with the required meta data exists and a fingerprint database was created. We then tested the algorithms concerning six test traces with respect to the proposed metric except the memory requirements, which is unfortunately not yet measurable from our tool. The test traces and the WLAN fingerprint database are depicted in Figure 4.

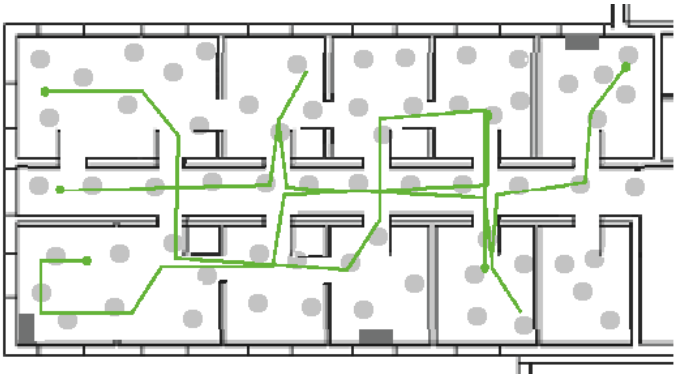


Figure 4: Reference positions (gray dots) and access points (gray rectangles), as well as ground truth tracks (green) for testing. Each displayed track was recorded starting at the left and at the right side.

Both algorithms start with an initial WLAN position calculated with a modified kNN algorithm calculating the variance of position estimation from the variance of the nearest neighbors to the estimated position. Furthermore, both algorithms utilize the same step detection algorithm with low-pass filtered accelerometer data and a cut-off threshold of 2ms^{-2} in a sliding window of one second. The step length was set manually and in both algorithms equal to 85cm with a relative large standard deviation of 20cm. The direction of movement was deduced from the magnetometer readings at the time of step detection, also disturbed by Gaussian noise with a standard deviation of $15°$. The variance of step detection was utilized to update the covariance matrix in the case of the Kalman filter and to move randomly disturbed particles in the case of the particle filter. Consecutive WLAN position estimates are either used to update the position (Kalman filter) according to the variances or to calculate a new weight for each particle conform to the probability of being at the specific position (particle filter). Moreover, the particle filter applied a simple map matching technique deleting all particles trying to move through obstacles. As a measure for complexity, the particle filter worked with 10^4 particles, using a sequential importance resampling technique to compensate for deleted particles and avoid degeneration.

The results are the mean values of results for all test traces and show the superiority of the particle filter to the Kalman filter at the test environment at our site. The particle filter is superior concerning accuracy, precision, and robustness, but suffers from a higher computational load resulting in a higher complexity. The results are summed up in Table 1.

Table 1: Evaluation results

Trait	Particle Filter	Kalman Filter
Accuracy	1.04 m	1.72 m
Precision	0.89 m	0.95 m
Robustness	4.01 m	4.52 m
Complexity	0.06	0.01

6 Conclusion and Future Work

In this paper, a tool for evaluation sensor fusion algorithms in indoor positioning scenarios has been presented. The tool consists of three components allowing for the recordings of sensor information, the provision of maps or training databases, and the evaluation and comparison of indoor positioning algorithms with a single dataset. In the paper, five different traits of a positioning system are deduced from requirements from location based services. The tool is able to evaluate an positioning system concerning four of these traits, i.e., accuracy, precision, robustness, and complexity.

As an application example, the comparison of a simple Kalman filter and a particle filter both based on WLAN fingerprinting and step detection is given concerning data from a test environment at our site. The test confirmed that in an office environment, the particle filter outperforms the Kalman filter with respect to accuracy and precision, but suffers from a higher complexity and slower calculation speed. This might be no news for indoor positioning, nevertheless a comparison of most existing sensor fusion approaches with carefully defined test sets regarding different scenarios such as positioning in large hallways, e.g., in a museum, shopping malls, office buildings, or private houses, should offer a new insight in sensor fusion mechanisms for indoor positioning and provide a standard benchmark for future research.

The next steps are the finalization of the tool and its online publication in addition with test traces to make it widely available in the community. This might be a first step towards standardization of benchmarks in indoor positioning scenarios and ease the development of new algorithms.

References

[AMGC02] M. S. Arulampalam, S. Maskell, Neil Gordon, and T. Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[BP00] P. Bahl and V. N. Padmanabhan. RADAR: an In-Building RF-based User Location and Tracking System. In *19th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2 of *INFOCOM*, pages 775–784, 2000.

[BTVG08] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 110:346–359, 2008.

[CBM10] Eddie C. L. Chan, G. Baciu, and S. C. Mak. Orientation-Based Wi-Fi Positioning on the Google Nexus One. In *6th International Conference on Wireless and Mobile Computing, Networking and Communications*, WiMob, pages 392–397, 2010.

[EM06] Frédéric Evennou and François Marx. Advanced Integration of WIFI and Inertial Navigation Systems for Indoor Mobile Positioning. *EURASIP Journal on Advances in Signal Processing*, 2006:1–11, 2006.

[GIK10] Dominik Gusenbauer, Carsten Isert, and Jens Krösche. Self-Contained Indoor Positioning on Off-The-Shelf Mobile Devices. In *Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN'10)*, 2010.

[HB07] H. Hile and G. Borriello. Information Overlay for Camera Phones in Indoor Environments. In *Location-and Context-Awareness: Third International Symposium*, LoCA, pages 68–84, 2007.

[KW11] Moritz Kessel and Martin Werner. SMARTPOS: Accurate and Precise Indoor Positioning on Mobile Phones. In *Proceedings of the International Conference on Mobile Services, Resources, and Users (MOBILITY'11)*, pages 158–163, 2011.

[Low99] David G. Lowe. Object Recognition from Local Scale-Invariant Features. In *International Conference on Computer Vision*, ICCV, pages 1150–1157, 1999.

[LSVW11] J. A. B. Link, Paul Smith, Nicolai Viol, and Klaus Wehrle. FootPath: Accurate Map-based Indoor Navigation Using Smartphones. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[MVFB10] Eladio Martin, Oriol Vinyals, Gerald Friedland, and Ruzena Bajcsy. Precise Indoor Localization Using Smart Phones. In *International Conference on Multimedia*, MM, pages 787–790, 2010.

[MWSB09] Alessandro Mulloni, Daniel Wagner, Dieter Schmalstieg, and Istvan Barakonyi. Indoor Positioning and Navigation with Camera Phones. *IEEE Pervasive Computing*, 8:22 – 31, 2009.

[SG05] P. Steggles and S. Gschwind. The Ubisense Smart Space Platform. In *Proceedings of the Third International Conference on Pervasive Computing (PERVASIVE'05)*, pages 73–76, 2005.

[SKC11] Martin Schäfer, Christian Knapp, and Samarjit Chakraborty. Automatic Generation of Topological Indoor Maps for Real-Time Map-Based Localization and Tracking. In *Proceedings of the 2nd International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[WH08] Oliver Woodman and Robert Harle. Pedestrian Localisation for Indoor Environments. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp'08)*, pages 114–123, 2008.

[WKM11] Martin Werner, Moritz Kessel, and Chadly Marouane. Indoor Positioning Using Smartphone Camera. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[YA05] Moustafa Youssef and Ashok Agrawala. The Horus WLAN Location Determination System. In *3rd International Conference on Mobile Systems, Applications, and Services*, MobiSys, pages 205–218, 2005.

Vergleich plattformübergreifender und nativer mobiler Anwendungen für ortsbasierte Web Services

Stefan Siegl, Carsten Kleiner

Hochschule Hannover
Fakultät IV, Abteilung Hannover
Ricklinger Stadtweg 120
30459 Hannover
carsten.kleiner@hs-hannover.de
siegl.stefan@googlemail.com

Abstract: Ortsbasierte Dienste werden heutzutage häufig mit mobilen Endgeräten genutzt, da diese sich stets am Ort des Nutzers befinden. Mobile Endgeräte nutzen dabei eine Vielzahl von verschiedenen Betriebssystemen. Jedes bringt sein eigenes Software Development Kit (SDK) mit sich, um Anwendungen dafür entwickeln zu können. Anwendungen, die mit einem speziellen SDK entwickelt wurden und nur auf den dafür vorgesehenen Smartphones genutzt werden können, werden auch native oder plattformabhängige Anwendungen genannt.

Plattformübergreifende Anwendungen können eine Alternative zu den nativen Anwendungen darstellen. Ein Groß eil der Anwendungslogik wird durch eine Webanwendung bereitgestellt und die Benutzeroberfläche wird mit Hilfe von Webtechnologien, wie HTML, CSS und JavaScript, gestaltet und somit plattformübergreifend definiert. Dadurch ist es möglich, eine Anwendung auf vielen Smartphones zur Verfügung zu stellen, ohne die Anwendung für jedes Smartphone neu entwickeln zu müssen. Als hybride Anwendung werden diejenigen Anwendung bezeichnet, bei denen nur ein geringer Teil der Anwendungslogik als plattformabhängige Komponente realisiert wird. Die restliche Anwendung wird ebenfalls mit Webtechnologien realisiert. Ein Zugriff auf die genannte Gerätefunktion kann somit möglicherweise einfacher realisiert werden.

Dieser Artikel befasst sich mit den verschiedenen Zugriffsmöglichkeiten auf die Position des mobilen Endgerätes und der Nutzung von ortsbasierten Web Services mit den betrachteten Anwendungsvarianten: nativ, webbasiert und hybrid. Bei der Webanwendung und der PhoneGap-Anwendung ist der Zugriff auf die Position über die Spezifikation *Geolocation API* möglich. Bei der Anwendung mit MonoTouch bzw. Mono For Android ist der Zugriff auf die Position mit den generierten Klassen des Frameworks möglich, die sich jedoch untereinander unterscheiden. Mit einer nativen Anwendung stehen die vom SDK bereitgestellten Möglichkeiten zur Verfügung. Für die Nutzung des LBS gibt es für jede Anwendungsvarianten verschiedene Werkzeuge um den Zugriff zu erleichtern.

Für eine sinnvolle Nutzung von LBS ist häufig eine lückenlose Aufzeichnung der Positionen unerlässlich. Da es für eine Webanwendung nicht möglich ist Positionsaktualisierung zu empfangen und zu verarbeiten, wenn der Browser im Hintergrund ist und eine andere Anwendung den Fokus hat, ist diese Form für viele, insbesondere kontinuierliche Dienste, weniger geeignet. Der Einsatz einer plattformübergreifender oder nativer Anwendung hängt von verschiedenen Faktoren wie Funktionsumfang, Know-How der Entwickler, der Anzahl der Plattformen, der Möglichkeit zur Bearbeitung im Hintergrund sowie der Performance ab.

1 Einleitung

Ortsbasierte Dienste werden heutzutage häufig mit mobilen Endgeräten genutzt, da diese sich stets am Ort des Nutzers befinden. Mobile Endgeräte nutzen dabei eine Vielzahl von verschiedenen Betriebssystemen. Jedes bringt sein eigenes Software Development Kit (SDK) mit sich, um Anwendungen dafür entwickeln zu können. Anwendungen, die mit einem speziellen SDK entwickelt wurden und nur auf den dafür vorgesehenen Smartphones genutzt werden können, werden auch native oder plattformabhängige Anwendungen genannt. Es ist mit sehr hohen Entwicklungskosten verbunden, wenn eine Anwendung für eine häufig genutzte Teilmenge aller Smartphones entwickelt werden soll, da der Quellcode wegen der Bindung an ein spezielles SDK selten wieder verwendet werden kann.

Als Alternative zu den plattformabhängigen Anwendungen hat sich das mobile Web erwiesen. Ein Großteil der Anwendungslogik wird durch eine Webanwendung bereitgestellt und die Benutzeroberfläche wird mit Hilfe von Webtechnologien, wie HTML, CSS und JavaScript, gestaltet und somit plattformübergreifend definiert. Dadurch

ist es möglich, eine Anwendung auf vielen Smartphones zur Verfügung zu stellen, ohne die Anwendung für jedes Smartphone neu entwickeln zu müssen. In [WIM12] werden die Architekturen von mobilen Anwendungen sowie die Frameworks bzw. Bibliotheken zur Realisierung behandelt.

Für mobile Webanwendungen wird meist eine von zwei Framework-Varianten eingesetzt. Bei der ersten Variante, den so genannten UI-Frameworks, welche im Rahmen einer reinen Webanwendung zum Einsatz kommen, wird der Zugang zu der Anwendung durch eine URL, die mit Hilfe eines Browser aufgerufen wird, bereitgestellt. Bei dieser Variante befindet sich selbst keine Anwendung auf dem Smartphone, wodurch sich ein Zugriff auf bestimmte Gerätefunktionen, wie z.B. die für LBS wichtige aktuelle Position des Gerätes, als schwierig erweisen kann. Als Hybrid-Frameworks wird die zweite Variante bezeichnet, bei der nur ein geringer Teil der Anwendungslogik als plattformabhängige Komponente realisiert wird. Ein Zugriff auf die genannte Gerätefunktion kann somit möglicherweise einfacher realisiert werden.

Mit Hilfe von *MonoTouch* und *Mono For Android* besteht ebenfalls die Möglichkeit einen Großteil der Anwendung für eine andere Plattform wieder zu verwenden. Bei den beiden Frameworks wird der Anwendungscode mit Hilfe von C# entwickelt und, vor dem Installieren der Anwendung auf dem jeweiligen Smartphone, in eine plattformabhängige Anwendung transformiert.

In diesem Artikel werden die Möglichkeiten, die eine plattformabhängige Anwendung bietet und die, die Frameworks des mobilen Web bereitstellen, um ein Web Service zur Nutzung eines LBS zu konsumieren untersucht und gegenübergestellt. In [OT12] wurden bereits Frameworks zur Entwicklung plattformübergreifender Anwendungen miteinander verglichen. Dieser Artikel befasst sich daher schwerpunktmäßig mit dem Vergleich der unterschiedlichen Anwendungsvarianten. Dabei wurde eine Webanwendung mit dem UI-Framework *jQuery Mobile*, eine Anwendung mit dem Hybrid-Framework *PhoneGap* und eine Anwendung mit *MonoTouch* bzw. *Mono For Android* für das iPhone und ein Android-Smartphone entwickelt. Für den Vergleich mit rein nativen Anwendungen wurde ferner eine iPhone Anwendung realisiert. Für Android wurde dies bereits in [Ber11] dargestellt.

Alle diese Anwendungen wurden für ein exemplarisches Anwendungsszenario entwickelt, das im folgenden Abschnitt kurz eingeführt wird. Eine ausführliche Darstellung des Szenarios findet sich in [BKZ10].

2 Anwendungsszenario

Im Folgenden wird die Architektur vorgestellt und im Anschluss die Kriterien ermittelt, welche für die verschiedenen Implementierungen beachtet werden sollen. Die Architektur wird in dem Artikel „Enhancing Customer Privacy for Commercial Continuous Location-based Services“ aus [BKZ10] beschrieben und hier kurz vorgestellt.

Bei Pay-as-you-Drive (PAYD) handelt es sich um ein Tarifmodell für die KFZ-Versicherung, bei dem die Versicherungsprämie basierend auf dem konkreten Fahrverhalten des Versicherungsnehmers berechnet wird. Um Datenschutzaspekten Rechnung zu tragen, ist eine einfache Übertragung des kompletten Bewegungsprofils an die Versicherung nicht akzeptabel.

Eine der möglichen Architekturen für PAYD wird in der Abbildung 1 vorgestellt. Dabei befindet sich das mobile Endgerät im Fahrzeug des Versicherungsnehmers, um so die für die Versicherung interessanten Positionsdaten zu ermitteln. Sobald sich die Position des Fahrzeugs verändert, wird dies durch das mobile Endgerät registriert und die Position auf dem mobilen Endgerät gespeichert. Um die gespeicherten Positionen zur Weiterverarbeitung an den LBS-Server (Location Based Service) versenden zu können, wird ein Pseudonym von der Versicherung (Insurance) benötigt. Die Authentisierung des Versicherungsnehmers gegenüber der Versicherung kann über die Versicherungsnummer und ein Passwort erfolgen. Das Erfragen des Pseudonyms wird über einen Web Service der Versicherung bereitgestellt, der entweder auf REST oder SOAP basiert.

Sobald die benötigte Anzahl an Positionen auf dem mobilen Endgerät vorhanden ist, können die Positionen, zusammen mit dem Pseudonym an den LBS-Server zur Weiterverarbeitung versendet werden. Anhand des Pseudonyms kann der LBS-Server kein Bewegungsprofil eines einzelnen Versicherungsnehmers erstellen, da die Pseudonyme sich regelmäßig ändern. Der LBS-Server bestimmt für die Positionen die von der Versicherung benötigten Metadaten, z.B. Art der Straßen, Geschwindigkeitsbegrenzungen und Unfallwahrscheinlichkeiten. Anhand der erhaltenen Daten wird nun überprüft, ob sich der Fahrer des Fahrzeugs den Verkehrsregeln entsprechend verhalten hat und wie

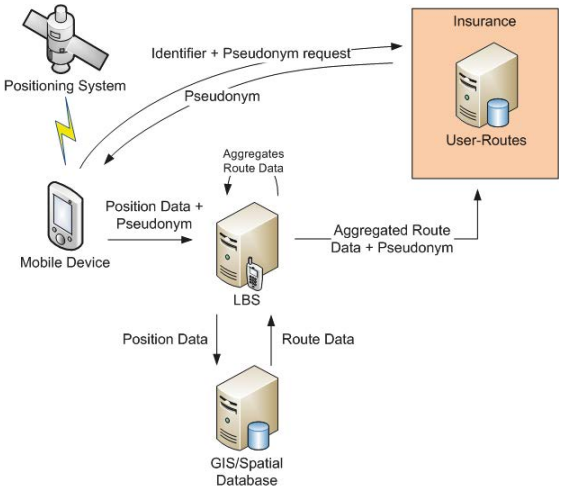


Abbildung 1: Eine der möglichen Architekturen für Pay-as-you-Drive

viele Kilometer auf welcher Straße zurückgelegt worden sind. Die gesammelten Daten werden in einem Bericht der Versicherung zur Verfügung gestellt. Die Versicherung erhält in diesem Szenario keine Liste mit Positionsdaten sondern eine Liste mit Informationen, die das Fahrverhalten des Fahrers wiedergeben, um so einen speziell für den Versicherungsnehmer angepassten Versicherungsbeitrag ermitteln zu können.

Neben der Bestimmung der Positionen, welche im Kapitel 4 vorgestellt wird, müssen die möglichen Implementierungen mit den Frameworks einen REST- und einen SOAP-basierten Web Service aufrufen können. Letzteres wird in Kapitel 5 dargestellt. Die Übertragung der Daten kann im Klartext erfolgen, sollte jedoch zur Wahrung der Vertraulichkeit und Integrität der Daten auch verschlüsselt möglich sein.

3 Anwendungsvarianten

Die Bestrebung, plattformübergreifende Anwendungen zu entwickeln, bestand schon vor einiger Zeit mit der Einführung von Java ME (Java Micro Edition) im Jahre 1999, wo die Aufteilung von Java in die drei Editionen JavaEE, JavaSE und JavaME vorgenommen wurde. Eine Unterstützung von Java ME auf den heutigen mobilen Endgeräten ist nicht mehr vorhanden. Die Herausforderung für die Entwicklung von plattformübergreifenden Anwendungen besteht jedoch weiterhin, da die Entwicklung verschiedener Anwendungen für die unterschiedlichen Plattformen in nativer Sprache sehr kostenintensiv sein kann, denn für jede Anwendung ist ein eigener Software-Entwicklungsprozess notwendig. Ebenso müssen die Entwickler die nötige Erfahrung für die Programmiersprache der jeweiligen Plattform besitzen.

Die Entwicklung von Anwendungen für die mobilen Endgeräte kann in drei Kategorien unterteilt werden: nativ, webbasiert und hybrid. Im Folgenden werden die Ansätze vorgestellt und bewertet.

3.1 native Anwendung

Eine native Anwendung wird für eine spezielle Plattform entwickelt und ist nur auf dieser Plattform funktionsfähig. Mit einer nativen Anwendung steht dem Entwickler der volle Funktionsumfang der Plattform zur Verfügung, und der Benutzer der Anwendung hat die gewohnte Benutzeroberfläche. Native Anwendungen sind bei rechenintensiven Anwendungen performancestärker als die hybriden oder webbasierten Anwendungen, jedoch ist der Unterschied in der Performance bei Geschäftsanwendungen vernachlässigbar [CL11]. Eine native Anwendung bietet die Möglichkeit, dass sie auch ohne eine Internetverbindung genutzt werden kann. Für die Entwicklung einer nativen Anwendung für verschiedene Plattformen ist das Wissen der jeweiligen Programmiersprachen notwendig. Hinzu kommt noch das notwendige Wissen über die verschiedenen Werkzeuge und APIs. Ein gewisser Aufwand für das Deployment der Anwendung ist notwendig, da sich die Bereitstellung über einen plattformspezifischen AppStore von Plattform zu Plattform unterscheidet. Bei einigen Plattformen sind entsprechende Entwickler-Zertifikate notwendig, mit denen die Anwendung signiert werden muss, bevor sie in den AppStore geladen werden kann. Diese Zertifikate sind oftmals kostenpflichtig. Zum Beispiel kostet ein Zertifikat, um Anwendungen in den AppStore des iOS zu laden, 99$ pro Jahr.

Zur Gruppe der nativen Anwendungen können auch die Anwendungen gezählt werden, die mit Hilfe von Cross-Compilern wie MonoTouch und Mono For Android erzeugt werden. Sie bieten nahezu den gleichen Funktionsumfang wie Zugriff auf die Hardware und die Möglichkeit der Verarbeitung von Daten im Hintergrund, wie native Anwendungen und unterstützen eine größere Anzahl an Plattformen mit einer annähernd identischen Code-Basis. Anpassungen sind hier an der Oberfläche und speziellen Gerätefunktionen notwendig, da sich diese zwischen den Plattformen stark unterscheiden. Die Anwendungslogik kann dabei plattformübergreifend genutzt werden. Ein grundlegendes Verständnis der zugrundeliegenden Plattform sowie deren Programmiersprache kann bei der Entwicklung der Anwendung von Vorteil sein, da die API in C# transformiert wurden ist. Die in der Dokumentation auffindbaren Programmbeispiele können so besser in die .NET-Welt übertragen werden.

Ebenso wie die nativen Anwendungen bietet diese Gruppe auch die Möglichkeit der Nutzung ohne Internetverbindung. Lediglich für das Absenden der Daten wird eine Verbindung zum Internet benötigt, jedoch nicht für das Starten/Laden der Anwendung. Sie haben dabei auch dieselben Nachteile wie die nativen Anwendungen, wie z.B. ein unter Umständen notwendiges Entwicklerzertifikat.

3.2 Webanwendung

Als Alternative zu den plattformabhängigen Anwendungen hat sich das mobile Web erwiesen. Dabei wird die Anwendungslogik durch eine Webanwendung bereitgestellt und die Benutzeroberfläche wird mit Hilfe von Webtechnologien, wie HTML, CSS und JavaScript, gestaltet und somit plattformübergreifend definiert. Dadurch ist es möglich, eine Anwendung auf einer Vielzahl von Geräten die einen Browser besitzen zur Verfügung zu stellen, ohne die Anwendung für jedes dieser Geräte neu entwickeln zu müssen.

Eine Webanwendung kann von einer großen Anzahl von Benutzern erreicht werden, da lediglich eine Internetverbindung und keine spezielle Hardware notwendig ist. Ebenso können bei der Entwicklung einer Webanwendung die Entwicklungskosten reduziert werden, da der Entwicklungsprozess über Analyse, Design, Implementierung und Test für die Plattformen (fast) identisch ist. Da eine Webanwendung keine native Anwendung ist, ist diese auch nicht an den plattformspezifischen Verteilungskanal gebunden und kann wie jede andere Webanwendung über einen Webserver bereitgestellt werden. Dies kann als Vor- und Nachteil gesehen werden, da einerseits nicht die Einschränkungen, wie z.B. die kostenpflichtigen Entwicklerzertifikate, in Kauf genommen werden müssen aber andererseits auch das Bezahlverhalten anders ist. Bei Anwendung aus einem AppStore sind die Benutzer gewohnt, dass es auch kostenpflichtige Produkte gibt. Der mobile Webbenutzer ist es hingegen gewohnt, die Inhalte meistens kostenlos zu konsumieren [Gla12]. Ob das Geschäftsmodell mit Premium-Accounts und Werbung der klassischen Webanwendungen auf den Bereich der mobilen Webanwendungen übertragen lässt bleibt abzuwarten. Durch eine Entkopplung des AppStores hin zu der Bereitstellung auf einem Webserver können Änderungen schneller vollzogen werden als bei einer nativen Anwendung, die unter Umstände noch von dem Hersteller geprüft wird.

Da für den Aufruf einer Webanwendung eine Internetverbindung benötigt wird, gestaltet sich die Nutzung ohne

Internetverbindung schwierig. Zwar kann über ein längeres Zwischenspeichern der Anwendung in einem Cache des Browser dieser Nachteil gemindert werden, er existiert aber dennoch. Die gewohnte Möglichkeit der Benutzeroberfläche der nativen Anwendungen kann durch die Verwendung von Webtechnologien nicht vollständig geboten werden. Durch die zusätzliche Abstraktionsschicht, die der Browser darstellt, kann es bei rechenintensiven Aufgaben zu Einschränkungen in der Performance kommen. Ferner können bei Webanwendung nur solche Funktionen genutzt werden, die auch durch eine entsprechende API durch den Browser zur Verfügung gestellt werden.

3.3 hybride Anwendung

Als hybrid wird die Gattung von Anwendungen für mobile Endgeräte bezeichnet, bei der einerseits ein Großteil der Anwendung mit Webtechnologien erstellt wird und andererseits eine Anwendung erstellt wird, die Zugriff auf eine Vielzahl von Gerätefunktionen besitzt, die mit einer reinen Webanwendung nicht möglich wären. Die Anwendung wird über einen plattformspezifischen Verteilungskanal, den AppStore, bereitgestellt. Die Plattformen erlauben es, einen rahmenlosen Browser zu starten, in dem die Anwendung geladen wird.

Bei hybriden Anwendungen werden Webtechnologien wie HTML/CSS und JavaScript genutzt, um die Benutzeroberfläche zu beschreiben und die Anwendungslogik zu entwickeln. Durch diese Unabhängigkeit von einer konkreten Plattform ist es möglich, dass hybride Anwendung auch plattformübergreifend genutzt werden können. Im Fall des in diesem Artikel verwendeten Frameworks PhoneGap stehen JavaScript-APIs zur Verfügung, um spezifische Gerätefunktionen, wie u.a Kamera, Positionsbestimmung oder Adressbuch, anzusprechen. Diese APIs sind allgemein definiert, so dass sie auch plattformübergreifend genutzt werden können. Bis zu dieser Stelle benötigt der Entwickler der Anwendung normalerweise nur die Kenntnis zur Entwicklung einer Webanwendung, da alle für den Anwendungsfall benötigten Funktionen durch das Framework über eine entsprechende API angeboten werden. Soll ein Teil der Anwendung in nativer Sprache realisiert werden, besteht die Möglichkeit bei PhoneGap, dies über Plugins zu realisieren. Ein solches Vorgehen kann notwendig sein, falls eine gewünschte Funktionalität nicht über die bereitgestellten Plugins realisiert werden kann oder falls eine rechenintensive Aufgabe erledigt werden muss. Für dieses Szenario benötigt der Entwickler die Kenntnis über jede Plattform sowie deren Programmiersprache die von der Anwendung unterstützt werden soll. Denn für jede dieser Plattformen muss ein Plugin in nativer Sprache entwickelt werden.

Am Ende des Build-Prozesses einer hybriden Anwendung steht eine native Anwendung, welche über einen plattformspezifischen Verteilungskanal bereitgestellt werden kann. Somit ist eventuell auch ein kostenpflichtiges Entwicklerzertifikat für die Bereitstellung der Anwendung notwendig. Durch die zusätzliche Abstraktionsschicht kann es sein, dass die hybride Anwendung unter Umständen langsamer ist als eine native Anwendung. Da die Oberfläche der hybriden Anwendung mit Webtechnologien wie HTML und CSS beschrieben wird, können nicht dieselben Benutzeroberflächen generiert werden wie mit einer nativen Anwendung. Dies kann sich negativ auf die Benutzung durch die Anwender auswirken [OHHG12]. Eine Übersicht der Aspekte bei den verschiedenen Anwendungsvarianten findet sich in Tabelle 1.

Kriterium	**Nativ**	**Hybrid**	**Web**
Funktionsumfang	voll	abh. vom Framework	gering
Notwendige Skills	Summe über alle Plattformen	HTML/CSS/JavaScript	
Verarbeitung von Daten im Hintergrund	Ja	Nein	Nein
Benutzeroberfläche	gewohnt nativ	mit Webtechnologien (HTML/CSS)	
Performance	gut	abh. vom Anwendungsfall	

Tabelle 1: Übersicht der Aspekte der verschiedenen Anwendungsvarianten

4 Positionsbestimmung

Für ortsbasierte Dienste stellt die Bestimmung der Position eine wichtige Aufgabe dar. In diesem Kapitel werden daher die Möglichkeiten zur Positionsbestimmung der einzelnen Anwendungsvarianten vorgestellt und miteinander verglichen.

Für das betrachtete *UI-* und *Hybrid-Framework* wird die Position mit Hilfe der *Geolocation API* [W3C12] realisiert. Diese API hat zum Ziel, eine einheitliche Schnittstelle für die Ermittlung des Gerätestandortes zur Verfügung zu stellen. Dabei steht dem Browser frei, woher er diese Daten bezieht - sie könnten aus der aktuellen Funkzelle, dem eingebauten GPS-Chip oder den Standortinformationen des Betriebssystem stammen [KB11]. Folgende Optionen können durch die Anwendung eingestellt werden:

- Genauigkeit Ja/Nein
- Zeitspanne für die Verarbeitung (Timeout)
- max. Alter für zwischengespeicherte Position

Die Anwendung kann dabei den Wunsch äußern, dass sie eine hohe Genauigkeit der Position haben will. Dies ist nur möglich, falls die verwendete Plattform diese Möglichkeit zur genaueren Bestimmung der Position besitzt.

Über verschiedene Kriterien wird die Technologie für die Positionsbestimmung bei *Android* und *Mono For Android* festgelegt. Der sogenannte *Location-Provider* nutzt folgenden Optionen für eine Positionsbestimmung:

- min. Zeitabstand
- min. Entfernung

Die erste Option bestimmt den minimalen Zeitabstand in Millisekunden und die zweite Option den Abstand zwischen den Positionsbestimmungen. Die Anwendung erhält somit eine Position, sobald beide Optionen erfüllt sind.

Eine native Anwendung die für *iOS* sowie eine Anwendung die mit *MonoTouch* entwickelt wurden ist, kann über die *CoreLocation-API* die Position des mobilen Endgerätes ermitteln. Folgende Optionen beeinflussen dabei die Art der Positionsbestimmung:

- DesiredAccuracy
- DistanceFilter

Die erste Option gibt die gewünschte Genauigkeit der Position an und bestimmt somit das verwendete Verfahren zur Positionsbestimmung. Die zweite Option bestimmt den minimalen Abstand zwischen zwei Positionen. Erst wenn dieser Wert überschritten ist erhält die Anwendung eine erneute Positionsaktualisierung.

Wir haben in diesem Kapitel gesehen, dass die Art der Positionsbestimmung der einzelnen Anwendungsvarianten im Detail recht unterschiedlich ausfällt. Mit der Geolocation-API stellt der Browser als Abstraktionsschicht eine Möglichkeit zur Positionsbestimmung für das betrachtete UI- und Hybrid-Framework zur Verfügung. Die Auswahl der benötigten Genauigkeit sowie die zur Verfügung stehenden Information die aus der API gewonnen werden können sind dabei eingeschränkt. Werden zum Beispiel Informationen wie Signalstärken der WLAN-Stationen benötigt um eine Positionsbestimmung vorzunehmen, ist dies mit der Geolocation-API nicht möglich. Mit einer MonoTouch- bzw. Mono-For-Android und iOS-Anwendung können solche Informationen aus der entsprechenden API ermittelt werden.

5 Nutzung eines LBS

Dieses Kapitel stellt einen Überblick über die Möglichkeiten dar, die für die einzelnen Anwendungsvarianten zur Verfügung stehen um einen Web Service zu konsumieren. Das Hauptaugenmerk liegt dabei auf einer vereinfachten

Nutzung des jeweiligen Web Service. Dies kann erreicht werden, durch eine Generierung von Stub-Code oder der Möglichkeit eines vereinfachten Zugriffs. Die Generierung hilft Entwicklungszeit zu sparen, da ein Großteil des benötigten Quell-Codes zum Aufruf des Web Service automatisiert anhand der Dienstbeschreibung (WSDL) erstellt wird. Der Quell-Code abstrahiert von der komplexen Erstellung der Nachricht, welche an den Web Service versendet wird und bietet somit einen vereinfachten Zugriff.

5.1 native Anwendung

Von iOS gibt es wenig Unterstützung in Form von Frameworks und Bibliotheken für die Konsumierung von REST- und SOAP-basierten Web Services. Für den Aufruf eines REST-basierten Web Services mit Objective-C in iOS kann das Framework RestKit[1] genutzt werden. Es stellt eine Schnittstelle zur Verfügung, um mit dieser Art von Web Service einfach interagieren zu können. RestKit bietet Unterstützung für das von Apple bereitgestellte Objektgraph- und Persistenzframework CoreData. Somit ist es möglich das Parsen der Antwort zu automatisieren und zur Persistierung an Core Data zu überführen. Diese Parser können dank einer einheitlichen Schnittstelle bei Bedarf ausgetauscht werden, falls ein anderes Datenformat für die Übertragung gewählt wird. Mit dem *Object Mapping System* können die per REST an den Client übertragenen fachlichen Objekte in native Objective-C Objekte transformiert werden. RestKit stellt noch weitere nützliche Funktionen für die Nutzung von REST-basierten Web Service bereit, die meistens eine einfachere Nutzung der Dienste erlauben. Für weitere Informationen sei an dieser Stelle auf die entsprechende Dokumentation verwiesen.

Für die Konsumierung eines SOAP-basierten Web Service gibt es einige Werkzeuge, mit denen die clientseitigen Stub-Klassen generiert werden können. Im Folgenden werden einige davon kurz vorgestellt. Zum einen gibt es das Programm wsdl2objc[2], welches nach der Angabe der URL der WSDL die Stub-Klassen generiert. Diese Klassen können nach der Einbindung in das Projekt und nach Anpassungen an dem Projekt, welche auf der Webseite beschrieben sind, genutzt werden. Die letzte Version des unter der MIT Lizenz stehenden Programms wurde im September 2009 veröffentlicht. Den Meldung des Versionsverwaltungssystems zur Folge wird an dem Tool jedoch aktiv gearbeitet.

Das Web Service Toolkit gSOAP[3] bietet ebenfalls die Möglichkeit, Stub-Klassen anhand einer WSDL zu generieren. Diese Stub-Klassen sind entweder in der Programmiersprache C oder C++. Da bei einer nativen iPhone-Anwendung einige Programmteile in C oder C++ realisiert werden können, stellt gSOAP ebenfalls eine Möglichkeit zur Konsumierung von Web Services mit einem iPhone dar. Falls die Nutzung von Web-Service-Features wie WS-Security geplant ist, kann dies mit gSOAP realisiert werden.

Über die Webseite http://sudzc.com/ können durch Angabe der URL zu der WSDL des Web Service ebenfalls clientseitige Stub-Klassen generiert werden. Diese Webseite untersteht der Apache License 2.0 und wurde mit .NET erstellt. XSLT wird genutzt um den clientseitigen Code anhand der WSDL zu generieren. Nach der Generierung wird automatisch ein Beispielprojekt im Format von XCode und den benötigten Klassen heruntergeladen. Diese Klassen können ebenfalls in das Projekt der iPhone-Anwendung kopiert und genutzt werden. Der Fokus dieser Webseite liegt dabei auf der Generierung von Stub-Klassen für Objective-C für das iPhone, jedoch werden auch JavaScript und ActionScript in einer Alpha-Version als mögliche Zielsprache unterstützt.

Bei der Entwicklung der *MonoTouch*- bzw. *Mono-For-Android*-Anwendung kann über die Entwicklungsumgebung MonoDevelop der Code auf der Seite des Clients für den Aufruf eines SOAP-Web-Service generiert werden. Dafür muss ein sogenannter Webverweis dem Projekt hinzugefügt werden, welcher die URL zu der WSDL des Web-Services erwartet. Unterstützt werden von MonoDevelop die Web-Services von Windows Communication Foundation (WCF) und ASP.NET 2.0.

Die WCF ist ein Framework zum Erstellen von Services mit .NET unter dem alle bekannten Kommunikationstechnologien von Microsoft zum Erstellen von verteilten Anwendungen unter einer einheitlichen Programmierschnittstelle zur Verfügung gestellt werden [HK08]. Neben SOAP, welches das Standardnachrichtenformat bei WCF ist, können auch beliebige Formate genutzt werden. Für die Übertragung der Nachrichten kann jedes beliebige Trans-

[1] http://restkit.org/
[2] http://code.google.com/p/wsdl2objc/
[3] http://www.genivia.com/products/gsoap/

portprotokoll genutzt werden. Im Vergleich dazu wird bei ASP.NET-Web-Services als Nachrichtenformat SOAP und als Übertragungsprotokoll HTTP genutzt [Mic12].

Die Bibliothek *RestSharp*[4] bietet eine vereinfachte API für REST-Web-Services sowie die Möglichkeit der Serialisierung und Deserialisierung von Objekte in verschiedene Datenformate an. Sie wird zusätzlich zu MonoTouch für verschiedene Plattformen, wie Windows Phone Mango, Mono For Android und .NET4 angeboten.

5.2 hybride und Webanwendung

Die Anwendungslogik wird bei einer hybriden Anwendung und bei einer Webanwendung mit JavaScript beschrieben, daher stehen für beide Varianten prinzipiell die selben Möglichkeiten zur Verfügung. Durch die PhoneGap-Plugins kann zusätzlich Programmcode in nativer Sprache realisiert werden.

Aufgrund der gute Unterstützung der JavaScript-API durch die Browser ist eine Bibliothek zum Aufruf der meisten REST-Web-Services nicht zwangsläufig notwendig. Für die Konsumierung eines SOAP-Web-Service kann der *JavaScript SOAP Client*[5] oder das Programm *wsdl2js*[6] des Apache Projektes *CXF* genutzt werden. Bei der ersten Variante wird der Web Service unter Angabe der URL zu der WSDL, des Methodennamens und der Methodenparameter aufgerufen und somit kein clientseitige Code generiert. Bei der zweiten Variante wird anhand der WSDL clientseitiger Code generiert, welcher in das jeweilige Projekt eingebunden werden kann.

Durch den JavaScript SOAP Client ist keine Generierung notwendig, was weniger Code-Umfang bedeutet. Dies kann jedoch auch eine schlechtere Möglichkeit für die Fehlersuche bedeuten. Die letzte Aktualisierung ist im Jahre 2007 erfolgt, so dass nicht von einer aktiven Entwicklung ausgegangen werden kann.

Zusammenfassend lässt sich sagen, dass Web Services mit den verschiedenen Anwendungsvarianten konsumiert werden können. Aufgrund der großen Nachfragen nach diesen Diensten haben sich unterschiedliche Möglichkeiten für die Nutzung ergeben. Welche Variante dabei für die Anwendung genutzt wird, hängt von Faktoren wie die dem Generierungsprozess des Quell-Codes, der Komplexität der API sowie unterstützten Features ab.

6 Vergleich der Anwendungsvarianten

Die Anwendungsvarianten wurden anhand von verschiedenen Kriterien untersucht. Diese Kriterien waren die Performance der Anwendung, der plattformübergreifende Anteil und die Anwendungsgröße. Das Ergebnis aus diesem Vergleich wird in dem folgenden Kapitel vorgestellt.

6.1 Performance

Die Bewertung der Performance basiert auf der Messung der Verarbeitungszeit durch die Anwendung, der Übertragung der Daten bis zum Erhalten und Verarbeiten der Antwort von den Web Service durch das mobile Endgerät. Das Ergebnis dieser Messung gibt Aufschluss darüber wie lange die jeweiligen Anwendungsvarianten für die Bearbeitung derselben Aufgabe benötigen. Um eine ausreichend große Menge an Daten zur Verarbeitung und Übertragung zu erhalten, wurde die Performance-Messung mit dem Web Service der Partei LBS durchgeführt. Die Verarbeitungszeit des Web Service für die erhaltene Route, um daraufhin eine Antwort an den Client zu versenden, kann im Idealfall für alle Messungen als konstant angesehen werden. Die Betrachtung der Verarbeitungszeiten der Server-Komponenten ist notwendig, da die Messung der Verarbeitungszeit auch die Dauer für die Bearbeitung der Antwort, welche vom Server an den Client versendet wird, beinhaltet. Die empfangene Route wird durch den Web Service in einer Message Queue zwischengespeichert und daraufhin sofort mit einer Antwort quittiert. Bei der Messung war die Message Queue ausreichend gefüllt, so dass die zuständigen Message Driven Beans für die

[4] http://restsharp.org
[5] http://javascriptsoapclient.codeplex.com
[6] http://cxf.apache.org/docs/wsdl-to-javascript.html

Aggregierung der Daten nie im Leerlauf waren und somit der Web Server eine annähernd konstante Auslastung hatte.

Die Messung und der Vergleich der Verarbeitungszeit ist dabei nicht ohne weiteres möglich, da die verwendete Hardware sich unter Umständen stark unterscheiden kann. Eine ausführliche Messung mit mehreren Geräte war nicht realisierbar. Für die Messung zur Verfügung standen das iPhone 4, mit der Version 5.1.1 des iPhone OS, und das Asus Eee Pad TF101, mit der Version 4.0.3 des Android OS. In dem iPhone arbeitet als Prozessor der Apple A4 mit 1 GHz und im Eee Pad ein 1 GHz Dual-Core-Prozessor. Dem iPhone stehen dabei 512 MB und dem Eee Pad 1024 MB Arbeitsspeicher zur Verfügung.

Die Übertragung der Daten hat dabei über *Wireless LAN* (IEEE 802.11g, max. 54 Mbit/s) stattgefunden. Diese Funkverbindung kann Störungen unterliegen, die die Messergebnisse beeinflussen. Um den Einfluss der Qualität der Funkverbindung zu minimieren, wird die Messung mehrmals durchgeführt und das Ergebnis anhand des arithmetischen Mittels der Verarbeitungszeiten ermittelt. Ferner wird die Standardabweichung für die Messreihe errechnet, um zu überprüfen ob die Verarbeitungszeit annähernd konstant ist. Auf den mobilen Endgeräten steht keine kabelgebundene Übertragungsmöglichkeit zur Verfügung, so dass die Übertragung über WLAN eine akzeptable Alternative darstellt. Durch eine Messung mit dem iPhone-Simulator und dem Android-Emulator wäre eine weniger störanfällige Datenübertragung möglich gewesen, denn der Simulator bzw. der Emulator nutzt die drahtgebundene Netzwerkverbindung des Host-Systems, doch die Geschwindigkeit des Simulators/Emulators unterscheidet sich dabei einerseits untereinander und anderseits von dem entsprechenden mobilen Endgerät.

Die Abbildungen 2 und 3 stellen die Dauer für das Versenden von 6.000 Positionen an den unverschlüsselten SOAP- und REST-Web-Service dar. Die Behauptung, dass native Anwendungen immer performanter sind, hat sich für diese Messreihe nur für das Android-Gerät bestätigt. Bei dem iPhone ist die Webanwendung für den betrachteten Anwendungsfall am performantesten. Allgemein kann gesagt werden, dass native Anwendungen bei rechenintensiven Aufgaben performanter als hybride und webbasierte Anwendungen sind. Jedoch ist der Unterschied in der Performance bei Geschäftsanwendungen meistens vernachlässigbar [CL11].

Für das iPhone hat die Webanwendung sowohl bei dem SOAP- als auch bei dem REST-Web-Service die geringste Zeit in Anspruch genommen. Danach folgt die hybride Anwendung mit PhoneGap, die native Anwendung und die MonoTouch-Anwendung. Die Anwendung zeigen auf dem Android-Smartphone ein anderes Verhalten in der Verarbeitungszeit. Hier benötigt die Mono-For-Android-Anwendung die geringste Zeit, gefolgt von der hybriden Anwendung mit PhoneGap und der Webanwendung.

Die Webanwendung wird keiner weitere Transformation unterzogen wie die PhoneGap bzw. Mono-Anwendung, so dass das Endprodukt auf beiden mobilen Endgeräten identisch ist. Die Vermutung dass die Verarbeitung von JavaScript durch den Safari-Browser performanter ist als durch den Android-Browser hat sich durch verschiedene Benchmarks nicht bestätigt. Da diese Benchmarks mehr Bereiche abdecken als für die betrachte Anwendung notwendig ist, hat eine weitere Untersuchung gezeigt, dass der Android-Browser ein vielfaches der Zeit für das Laden der Daten aus der Datenbank benötigt als Safari. Dies lässt auf eine bessere Unterstützung der *Web SQL Database* durch den Safari-Browser schließen.

PhoneGap verwendet einen rahmenlosen Browser, um die Inhalte darzustellen. Für das iPhone OS wird die native Klasse UiWebView und für das Android OS die Klasse android.webkit.WebView verwendet. Die Unterschiede in den verschiedenen Laufzeiten der PhoneGap-Anwendung auf dem iPhone und dem Android-Smartphone könnten auf dasselbe Problem wie bei der Webanwendung zurückgeführt werden.

Wie später noch gezeigt wird, wird ein Großteil der Mono-Anwendung für beide Plattformen gemeinsam genutzt. Lediglich die Oberfläche sowie der Zugriff auf die Positionen wurde plattformspezfisch realisiert. Die Mono-Anwendung wird während des Build-Vorgangs in eine native Anwendung transformiert. Bei der Laufzeit der MonoTouch- bzw. Mono-For-Android-Anwendung gibt es deutliche Unterschiede. Die Anwendung auf dem Eee Pad benötigt weniger Zeit für die Verarbeitung als auf dem iPhone. Dies kann einerseits von den unterschiedlichen Ladezeiten der Positionen aus der Datenbank, von der verschiedenen Hardware der Endgeräte und/oder von dem unterschiedlichen Ergebnis des Transformationsprozesses resultieren.

Das Kapitel hat gezeigt, dass nahezu identische Anwendungen auf verschieden Plattformen unterschiedliche Laufzeiten haben und die Performance von unterschiedlichen Faktoren abhängt. Dabei wurde gezeigt, dass das Laden der Positionen aus der Web SQL Database mit dem Android-Browser um ein Vielfaches länger dauert als mit dem Safari. Anderseits ist das Ergebnis des Cross Compilers unter Android schneller als unter iOS. Insgesamt lässt sich

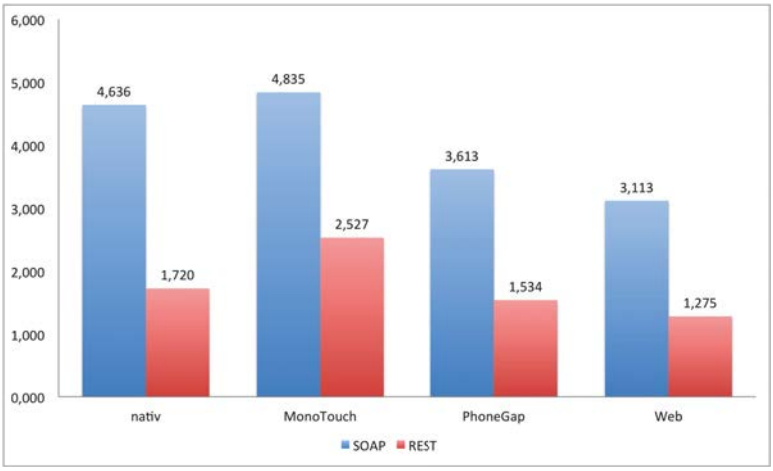


Abbildung 2: Die Abbildung zeigt die mittlere Dauer die das iPhone für das Versenden von 6.000 Positionen über den unverschlüsselten SOAP- bzw- REST-Web-Service benötigt.

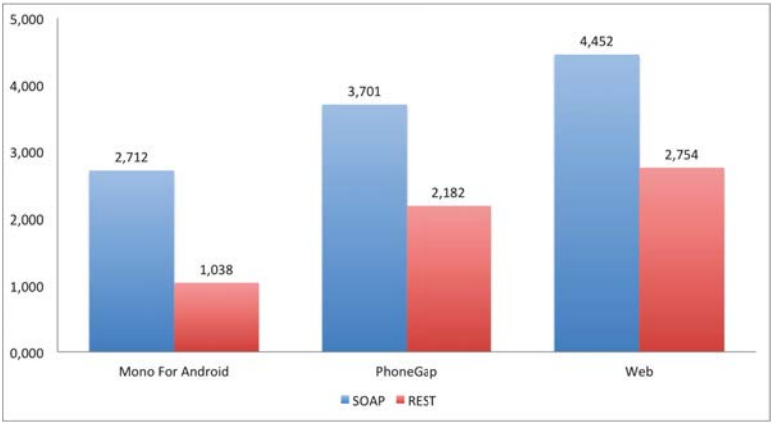


Abbildung 3: Die Abbildung zeigt die mittlere Dauer die das Android-Smartphone für das Versenden von 6.000 Positionen über den unverschlüsselten SOAP- bzw- REST-Web-Service benötigt.

also feststellen, dass für jede Software individuelle Messung erforderlich sind, um eine Aussage über die Performanz machen zu können. Eine allgemeingültige Regel, welche Variante am schnellsten läuft, kann nicht abgeleitet werden.

6.2 Plattformübergreifender Anteil

Bei der Entwicklung der verschiedenen Anwendungen wurde darauf Wert gelegt, dass ein Großteil des Quellcodes wiederverwendet werden kann. Es wurden diejenigen Lösungen gewählt, die auf beiden Plattformen zum Erfolg führen. Die Zeilen im Quellcode, die im Nachhinein als plattformspezifisch identifiziert werden konnten, wurden gezählt.

Die Abbildung 4 zeigt die jeweiligen plattformübergreifenden bzw. plattformspezifischen Anteile der übrigen Anwendungen in der Einheit Lines-of-Code. Die PhoneGap-Anwendung besitzt nur einen geringen plattformspezifischen Teil. Dieser war notwendig, da die iPhone-Anwendung mit der PhoneGap-Version 1.7.0 den Zeitstempel bei den Positionsaktualisierungen in Sekunden und nicht in Millisekunden liefert. Laut der Dokumentation der aktuellen PhoneGap-Version 2.0 gibt es diese Ausnahme nicht mehr, so dass der plattformspezfische Teil bei einer Aktualisierung der Version gegen null gehen kann. Bei den Mono-Anwendungen konnte ebenfalls ein beachtlicher Teil wiederverwendet werden. Lediglich die Benutzeroberfläche und die Positionsbestimmung müssen aufgrund der großen Unterschiede zwischen den Plattformen plattformspezifisch realisiert werden.

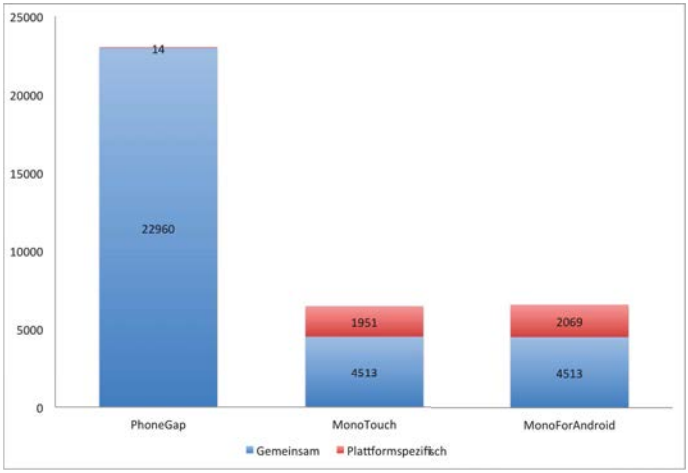


Abbildung 4: Plattformübergreifender Anteil der verschiedenen Anwendungen.

Die Abbildung 4 erweckt den Eindruck, dass die MonoTouch- bzw. Mono-For-Android-Anwendung weniger Lines-of-Code für die Erfüllung der Aufgabe benötigt als die PhoneGap-Anwendung. Dies hängt jedoch damit zusammen, dass das Erstellen der SOAP-Nachricht durch das Mono-Framework übernommen wird und ferner eine Bibliothek für den Aufruf des REST-Web-Services verwendet wird. Für die Messung der Lines-Of-Code konnten die Bibliothek und das Mono-Framework aufgrund des binären Formats nicht berücksichtigt werden. Im Gegensatz dazu steht die PhoneGap-Anwendung, bei der der Programmcode zum Aufruf des SOAP- und REST-Web-Service in reinem Quellcode vorhanden ist und somit die Zählung beeinflusst.

Ferner wurden die Lines-of-Code der gesamten Anwendung betrachtet. Die Webanwendung (ca. 113.000) zeigte aufgrund der gewählten Architektur, durch Verwendung von iframes, und den damit verbundenen zusätzlichen Bibliotheken einen sehr großen Wert. Wenn die Webanwendung den Web Service direkt aufrufen könnte bzw. über einen Web-Service-Proxy wären die Lines-of-Code vermutlich geringer und mit den anderen Anwendungen besser vergleichbar. Die PhoneGap-Anwendung für das iPhone (ca. 38.000) und das Android-Smartphone (ca. 40.000) waren nahezu identisch. Der kleine Unterschied resultiert aus einer größeren Bibliothek von PhoneGap für

	nativ	Mono*	PhoneGap	Web
Android	-	416	1180	1884
iPhone	856	4020	2728	1884

Tabelle 2: Die Tabelle zeigt die Anwendungsgröße der Anwendung auf dem jeweiligen Endgerät.

das Android-OS im Vergleich zu iOS. Der minimale Unterschied zwischen der MonoTouch- (ca. 6.400) und der Mono-For-Android-Anwendung (ca. 6.600) resultiert aus dem plattformspezifischen Teil. Bei der iOS-Anwendung (ca. 30.700) sind die Programmteile zum Aufruf der Web Services ebenfalls im Quellcode - und nicht in einer Bibliothek oder Framework - vorhanden, wie auch bei den PhoneGap-Anwendungen. Auffällig ist hier, dass für die Erfüllung der selben Aufgabe mehr Zeilen an JavaScript- als an Objective-C-Quellcode benötigt wird.

6.3 Anwendungsgröße

Da sich eine Untersuchung der Lines-of-Code als schwierig erweisen kann, wenn Programmteile in binären Format vorliegen, wird ferner die Größe der Anwendung untersucht, die auf dem Endgerät installiert wird. Die Anwendungsgröße wird in Kilobyte gemessen und in der Tabelle 2 dargestellt.

Ein großer Unterschied ist bei den Mono-Anwendungen festzustellen. Aufgrund der Restriktionen über das Nachladen von Programmteilen zur Laufzeit, wird bei dem Erstellen der MonoTouch-Anwendung die Mono-Laufzeitumgebung statisch verlinkt, was sich negativ auf die Anwendungsgröße auswirkt. Bei Android ist die nicht nötig und die Mono-Laufzeitumgebung wird zur Laufzeit dynamisch nachgeladen, da die Restriktionen dort nicht bestehen. Der Unterschied bei den PhoneGap-Anwendung kommt zustande, weil der Erstellungsprozess der PhoneGap-Anwendung bei der iPhone-Anwendung einen Ordner mit Bilder hinzufügt, der nicht von jeder Anwendung benötigt wird. Die PhoneGap-Anwendung für Android hat eine geringere Anwendungsgröße, da dieser Ordner nicht hinzugefügt wird.

Zusammenfassend lässt sich sagen, dass die absolute Größe bei eher einfachen Anwendungen nur in geringem Maße von der eigentlichen Eigenimplementierung, sondern eher von den plattformspezifischen Gegebenheiten bei der Ausführung abhängen. Vernachlässigt man diese Aspekte lässt sich ein Trend erkennen, dass die Größe der Anwendung sinkt je näher man der Plattform kommt.

7 Fazit

Im Rahmen dieser Arbeit wurden verschiedene Ansätze für plattformübergreifende Anwendungen für mobile Endgeräte vorgestellt. Jeweils eine Anwendung aus den verschiedene Bereichen wurde entwickelt. Dies war eine Anwendung in Form einer reinen Webanwendung, eine mit dem hybrid Framework PhoneGap und eine mit dem Cross-Compiler MonoTouch bzw. Mono For Android für die Plattformen iPhone und Android. Für den Vergleich mit rein nativen Anwendungen wurde ferner eine iPhone Anwendung realisiert. Für das Android OS wurde dies bereits in [Ber11] dargestellt.

Im weiteren Verlauf der Arbeit wurde ein Vergleich anhand der Verarbeitungszeit für den Aufruf eines Web Service, der Anteile von plattformübergreifenden Code sowie der Anwendungsgröße durchgeführt. Bei der Verarbeitungszeit wurde festgestellt, dass die Emulatoren die Laufzeit des physischen Gerätes nicht immer widerspiegeln können. Ebenso ist die Laufzeit der verschiedenen Anwendungsvarianten auf den betrachteten Plattformen durchaus unterschiedlich. So ist die native Anwendung auf dem iPhone lediglich auf Platz 3 der Performance-Messung gelandet und den ersten Platz belegte dabei die Webanwendung. Dies kann auf eine performantere Verarbeitung von JavaScript durch den Safari oder auf eine ineffiziente Verarbeitung von Objective-C bzw. auf die unterschiedliche Qualität der generierten Anwendungsteile zurückgeführt werden. Bei dem Android-Smartphone zeigte die Performance-Messung, dass je näher das Framework an der Plattform ist desto performanter wird die Anwendung ausgeführt.

Bei einer näheren Betrachtung der Webanwendung wurde beobachtet, dass die Anwendung auf dem Eee Pad sowohl bei dem REST- bzw. SOAP-basierten Web Service eine erhöhte Laufzeit gegenüber der Anwendung auf dem iPhone zeigte. Dies konnte jedoch nicht auf eine langsamere Verarbeitung von JavaScript durch den Android-Browser zurückgeführt werden, sondern auf eine performantere Benutzung der Web SQL Database durch den Safari-Browser.

Als weiterer Vergleich wurde der Anteil an gemeinsamen Quellcode betrachtet. Ein hoher Anteil bedeutet, dass ein Großteil des Quellcodes für eine andere Plattform genutzt werden kann. Bei einer Webanwendung für mobile Endgeräte wird die entwickelte Software nicht gegen spezifische API's der Endgeräte entwickelt sondern gegen die Schnittstellen, die der Browser zur Verfügung stellt. Der Browser stellt demnach eine Abstraktionsschicht dar, denn die entwickelte Webanwendung ist - solange die benutzte Funktionalität auf dem Endgerät zur Verfügung steht - auch auf anderen Geräten, sogar auf Desktop-Rechnern lauffähig. So wurde die Webanwendung zu einem erheblichen Anteil auf einen Mac OS X mit Safari als Browser getestet, da dieser Browser mehr Möglichkeiten zum Debuggen bereitstellt als auf einem mobilen Endgerät. Über die Schnittstellen, die der Browser bereitstellt, ist es zur Zeit noch nicht möglich, dass eine Aufzeichnung der Positionen stattfindet, wenn sich die Anwendung in dem Fall der Browser, im Hintergrund befindet. Wenn der Browser durch das Starten einer anderen Anwendung den Fokus verliert, werden keine Aktualisierungen der Position mehr empfangen bzw. verarbeitet. Die Aufzeichnung wird erst fortgesetzt, wenn der Browser wieder im Vordergrund ist. Für eine sinnvolle Nutzung von ortsbasierten Diensten ist häufig eine lückenlose Aufzeichnung der Positionen unerlässlich. Da es für eine Webanwendung nicht möglich ist, Positionsaktualisierung zu empfangen und zu verarbeiten, wenn der Browser im Hintergrund ist, und eine andere Anwendung den Fokus hat, ist diese Form für viele, insbesondere kontinuierliche Dienste, weniger geeignet.

Diese Arbeit hat gezeigt, dass technisch alle Kombinationen möglich sind. Für welche Anwendungsvariante man sich entscheidet hängt von verschiedenen Faktoren wie Funktionsumfang, Know-How der Entwickler, der Anzahl der Plattformen, der Möglichkeit zur Bearbeitung im Hintergrundprozess sowie der Performance ab.

Literatur

[Ber11] Jens Bertram. Implementierung und Sicherheitsaspekte von Web Services für Android. 2011.

[BKZ10] Jens Bertram, Carsten Kleiner und David Zhang. Enhancing Customer Privacy for Commercial Continuous Location-based Services. In Thomas Magedanz, Ying Cai, Minglu Li, Jinchun Xia und Carlo Giannelli, Hrsg., *Mobile Wireless Middleware, Operating Systems, and Applications*, Third International Conference, Mobileware 2010, Chicago, IL, USA, June 30 - July 2, 2010. Revised Selected Papers, Spring LNCIST VOL.48: Heidelberg, S. 326-337, 2010.

[CL11] Andre Charland und Brian LeRoux. Mobile Application Development: Web vs. Native. *ACM*, 2011.

[Gla12] Kay Glahn. Aus Web wird Native. *Heise Zeitschriften Verlag*, 2012.

[HK08] Stephanie Hölzl und Jürgen Kotz. *WCF - Die Windows Communication Foundation: Verteilte Anwendungsentwicklung mit der Microsoft Kommunikationsplattform*. Addison-Wesley, München, 1. Auflage, 10 2008.

[KB11] Sandra Krüger und Helmut Balzert. *HTML5, XHMTL & CSS*. W3L, 2011.

[Mic12] Microsoft. Comparing ASP.NET Web Services to WCF Based on Purpose and Standards Used. http://msdn.microsoft.com/en-us/library/aa702755, 06 2012.

[OHHG12] Scott Olson, John Hunter, Ben Horgen und Kenny Goers. *Professional Cross-Platform Mobile Development in C#*. Wrox, 1. Auflage, 2 2012.

[OT12] J. Ohrt und V. Turau. Cross-Platform Development Tools for Smartphone Applications. *Computer*, PP(99):1, 2012.

[W3C12] W3C. Geolocation API Specifications. http://dev.w3.org/geo/api/, 04 2012.

[WIM12] Felix Willnecker, Damir Ismailović und Wolfgang Maison. Architekturen mobiler Multiplattform-Apps. *Smart Mobile Apps - Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse*, 2012.

Indoor Positioning by Fusion of IEEE 802.11 Fingerprinting and Compass Bearing

Lars Fischer, Nils Hahn

RG IT-Security Managment
Prof. Dr. Dogan Kesdogan

Andreas Hoffmann

RG Operating and Distributed Systems
Prof. Dr. Roland Wismüller

University Siegen
Hölderlinstr. 3
57076 Siegen
{fischer@wiwi.,andreas.hoffmann@}uni-siegen.de

Abstract: Location fingerprinting using Received Signal Strength (RSS) nowadays is one of the fundamental techniques for indoor localisation, albeit it proves very unreliable. It is therefore necessary to supplement RSS with additional information. Among the many sensors currently build into ubiquitous mobile devices, this work focused on magnetometers. We analysed sensitivity of 802.11 RSS fingerprints to bearing and further utilised bearing information for bayesian correction in positioning.

Positioning in the interior of buildings is still a field of opportunities for improvement. For our intended application in consumer research, high accuracy and precision better than 20 cm are required. Within a supermarket a short distance usually means the difference between one or another product in the shelf. In this paper we present work on the fusion of IEEE 802.11 Fingerprinting and compass bearing using Kalman Filters. Based on the Redpin localisation server[Bol08] we developed and analysed localisation by sensor fusion, with or without position interpolation. Results are not yet within the objective, but this work provides a base to include other additional sensors.

Localisation using IEEE 802.11 *Received Signal Strenght Indication* (RSS) fingerprints is a well known method and in spite of many optimisations not sufficiently accurate or precise [LSDR06]. Usually each distinguishable spatial location is associated with a single set of repeated RSS measurements. One improvement is to additionally distinguish sets of fingerprints at each location by orientation [KKH+06].

In this work we present a method to combine compass orientation, RSS fingerprint interpolation with Bayesian estimation to improve localisation.

1 Methodology

We use a discrete space model of disjoint cells, which is initialised by repeated RSS-measurements at known positions. A measurement at time t during a *Learning Phase*, con-

sist of a vector F_t of RSS measurements from different access points (AP) $(rssi_i, bssid_i) \in AP_t$ and orientation $theta_t$ from an internal magnetometer sensor. Each set of RSS measurements is condensed, assuming of normal distribution, to mean and standard deviation.

Basic single position localisation is undertaken by maximum probability of position x given F_t denoted $max_x P(x|F_t)$. Probability $P(x|F_t)$ is calculated as the product of conditioned probabilities of receiving a given $rssi_i$ at x.

Without interpolation a device is considered to be localised at the cell whose fingerprint distribution has the highest probability to have produced that fingerprint. We further use interpolation to increase the number of cell positions, i.e. to allow to determine positions within cells that have not been measured in the Learning Phase.

We further use interpolation during the Online Phase to increase precision of localisation. We use *weighted k-nearest-neighbours* to reduce the effects of outliers with good results.

Orientation θ is used in two ways. First to distinguish sets of fingerprints by orientation. And second for Recursive Bayesian Estimation of successive positions. We apply two models, one discrete, distinguishing four direction and one continuous to a state transition matrix $R^{n \times n}$. The transition matrix further uses an expectation of movement speed as baseline probability for transition between cells. The continuous model applies a weight a if the angle of movement between locations θ_{x_{t-1},x_t} is within an ϵ environment of the current orientation θ.

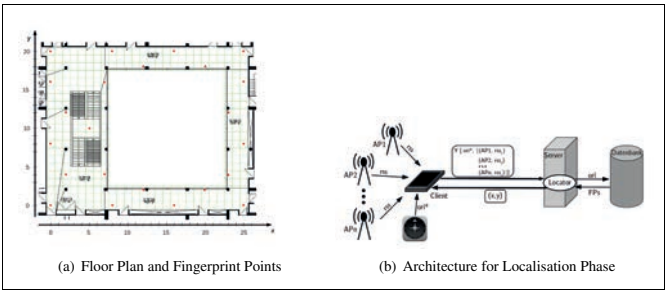
$$
\begin{aligned}
P(x_t|x_{t-1}) =& \gamma_{x_{t-1},x_t} P(x_t|F_t) \\
\gamma_{x_{t-1},x_t} =& \begin{cases} a, if & \theta - \epsilon < \theta_{x_{t-1},x_t} < \theta + \epsilon \\ 1, if & otherwise, \end{cases}
\end{aligned} \tag{1}
$$

where the parameter γ_{x_{t-1},x_t} is used as a weight to the probability of being at location x_t given measurement F_t which results in the updated probability $P(x_t|x_{t-1})$ of being at x_t, conditioned on previously observed location x_{t-1}.

2 Experiments and Results

We examined different combinations of compass bearing, fingerprint models, interpolation and weights. The testing field was a spacious floor with an area of $26 \times 21\ m$ surrounding an atrium at our campus. The floor conveniently provided square partitions of 50 cm side length that rendered a grid of 1 m cells. We installed two additional access points to produce a minimum set of available access points in the whole area. To provide comparable tests and increase efficiency we first measured at 33 locations for each four directions 20 RSS samples of all available access points (see Figure 1(a)).

To test single positioning with orientation information measurements where separated in a *learning set* and a *testing set*. The learning set represents the measurements during the Offline Phase and the testing set simulated measurements for localisation in the Online Phase. The used communication setup of APs, mobile device and localisation server is



(a) Floor Plan and Fingerprint Points (b) Architecture for Localisation Phase

Figure 1: Experimental Setup

shown in Figure 1(b).

The results of the simulation provided us with mean errors as given in Table 1. The approach to interpolate fingerprints during the Offline Phase showed little effect. We further analysed the effect of interpolation of positions using weighted k-nearest neighbours and found that the mean error approaches its minimum at $k = 5$.

Method	oriented FP?	mean error	90th percentile
Likelyhood $P(x\|F_t)$	4 ori.	$\approx 3.69\,m$	$\approx 10.31m$
Likelyhood $P(x\|F_t)$	no	$\approx 5.60\,m$	$\approx 12.47m$

Table 1: Single Position Simulation

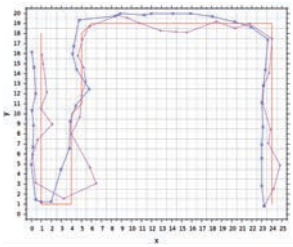
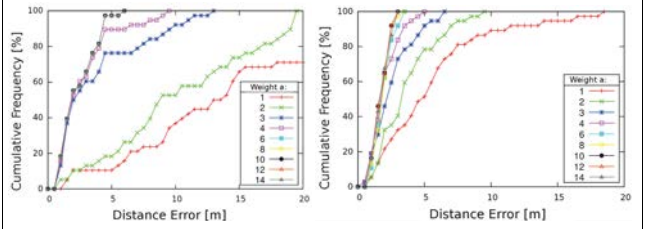


Figure 2: Tracking Approach 1 (magenta), 2 (blue)

To analyse effects of Bayesian estimation a sequence of measurements had to be produced. We undertook localisation along the path shown in Figure 2 in the test area. Distance errors where calculated for the same categories as for the simulated localisation above. We analysed two different approaches considering handling of magnetometer orientation

data; a first approach used four discrete orientations, applying weight a if the orientation lied within the discrete sector. A second approach utilised the continuous approach given by Equation (1).



(a) Distance Error Approach 1 (discrete orientation)

(b) Distance Error Approach 2 (continuous orientation)

Figure 3: Distance Error of Interpolated, Oriented Fingerprints using Bayesian Estimation.

The differences between Approach 1 and Approach 2 in Figure 3 seem obvious, alas, it stands to reason, that these values are based on single test-runs.

3 Conclusion

Albeit our methods differed only in details from known methods, our approach indicates that sensor fusion is a promising approach. Indoor localisation can not be considered solved, neither by commercial approaches nor by academic initiatives. Next steps in our work will consist in consolidating of results and extension to different sensor inputs.

References

[Bol08] Philipp Bolliger. Redpin - Adaptive, Zero-Configuration Indoor Localization through User Collaboration. In *proceedings of MELT'08*, 2008.

[KKH+06] Thomas King, Stephan Kopf, Thomas Haenselmann, Christian Lubberger, and Wolfgang Effelsberg. COMPASS: A probabilistic indoor positioning system based on 802.11 and digital compasses. In *Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization*, WiNTECH '06, pages 3440, New York, NY, USA, 2006. ACM.

[LSDR06] Binghao Li, James Salter, Andrew G. Dempster, and Chris Rizos. Indoor positioning techniques based on wireless LAN. In *LAN, FIRST IEEE INTERNATIONAL CONFERENCE ON WIRELESS BROADBAND AND ULTRA WIDEBAND COMMUNICATIONS*, pages 13–16, 2006.

Ein Taskmodell für Raum-Zeit-Scheduling

Tammo M. Stupp, Daniel Graff, Anselm Busse, Jan Richling

Fachgebiet Kommunikations- und Betriebssysteme
Technische Universität Berlin
Einsteinufer 17 / EN 6
10587 Berlin
{stupp, dgraff, abusse, richling}@cs.tu-berlin.de

Abstract: Im Bereich der Echtzeitsysteme existiert mit dem periodischen Taskmodell ein Modell, das die Lösung von Schedulingproblemen stark vereinfacht. In dieser Arbeit stellen wir uns der Herausforderung, ein ähnlich einfaches Modell für cyber-physische Systeme, bei denen neben zeitlichen Anforderungen insbesondere auch der Ort der Ausführung eine Rolle spielt, zu finden. Wir spannen dazu mit einem generischen, aber sehr komplexen Modell den Problemraum auf und diskutieren praktikable Vereinfachungen.

1 Einleitung

Das allgemeine Schedulingproblem ist NP-vollständig [Liu00], so dass zur Lösung Heuristiken oder starke Vereinfachungen zum Einsatz kommen. Ein klassisches Beispiel solcher Vereinfachungen ist das periodische Taskmodell für Echtzeitsysteme. So vereinfachte Schedulingprobleme haben höchstens einen polynomialen Aufwand und lassen sich somit zur Laufzeit lösen. Ferner werden hierdurch einfache Entscheidungen hinsichtlich der Ausführbarkeit ermöglicht. Die zur Vereinfachung nötigen Annahmen schränken die Menge der möglichen Aufgaben zwar ein, erlauben aber in der betrachteten Problemdomäne eine relevante Untermenge. Werden diese Annahmen jedoch fallen gelassen, wie es für cyber-physische Systeme erforderlich ist, die neben der Zeit auch noch weitere Bedingungen (z. B. Ortsinformationen) berücksichtigen müssen, so sind diese Vereinfachungen nicht mehr anwendbar.

Ziel dieser Arbeit ist es, in Analogie zum periodischen Taskmodell der Echtzeitwelt sinnvolle Vereinfachungen für das generische Schedulingproblem der cyber-physischen Systeme zu schaffen. Diese sollen es ermöglichen, Lösungen für relevante Anwendungen ebenfalls mit einem höchstens polynomialen Aufwand zu finden. Analog zum Echtzeitscheduling, das sich auf die Verwaltung des Prozessors als einzig relevante Ressource beschränkt, soll sich diese Arbeit auf das Scheduling in Zeit und Raum beschränken, insbesondere also alle weiteren Anforderungen ignorieren. Hierfür wird ein einfaches Taskmodell benötigt, das diese beiden Parameter berücksichtigt. Um dieses zu erlangen, entwickeln wir zunächst ein generisches Modell für das Raum-Zeit-Scheduling, das wir in einem weiteren Schritt als Ausgangspunkt für Vereinfachungen benutzen.

Diese Arbeit ist folgendermaßen gegliedert: Wir beginnen mit der detaillierten Beschreibung des komplexen Modell in Abschnitt 2, um darauf aufbauend in Abschnitt 3 Vereinfachungen zu diskutieren. Abgeschlossen wird das Papier in Abschnitt 4 mit zusammenfassenden Bemerkungen und einem Ausblick auf nachfolgende Forschungsarbeiten.

2 Modell

In unserem Modell wird eine Menge von *Aufgaben* mit Hilfe einer Menge von *Akteuren* unter Beachtung von *Raum-Zeit-Bedingungen* ausgeführt. Eine Raum-Zeit-Bedingung spezifiziert Raum-Zeit-Fenster, also eine Teilmenge der durch den Anwendungsfall aufgespannten Raum-Zeit (die Dimensionalität des Raums ist anwendungsabhängig). Raum-Zeit-Bedingungen können absolut oder relativ (z. B. zu Objekten der Umgebung) sein und bilden die Menge C. Ein Beispiel für solche Raum-Zeit-Fenster zeigt Abbildung 1, wobei der Raum im Interesse

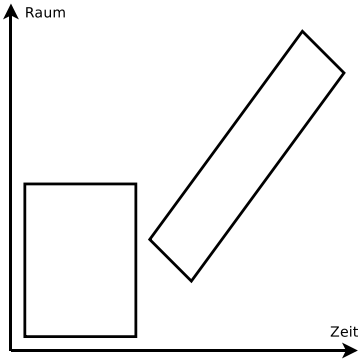


Abbildung 1: Raum-Zeit-Bedingungen mit nur einer Raumdimension.

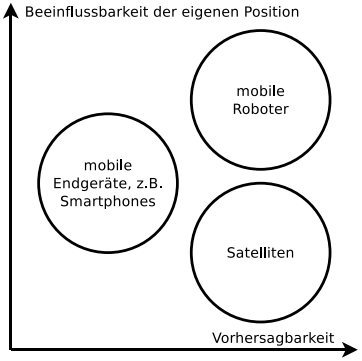


Abbildung 2: Einfache Kategorisierung von Akteuren entlang der beiden betrachteten Dimensionen.

der einfachen Darstellung auf eine Dimension beschränkt ist. Das bedeutet nun, dass sich die Akteure innerhalb der spezifizierten Zeiten nur innerhalb bestimmter Abschnitte einer Strecke befinden dürfen bzw. dass die Aufgaben innerhalb dieser Zeiten an diesen Orten bearbeitet werden müssen. Weiterhin ist es auf diese Weise möglich, ortsabhängige Deadlines zu spezifizieren, wie es das rechte Element der Abbildung zeigt.

Ein Akteur ist ein cyber-physisches System, dessen Betrachtung in unserem Modell auf die Fähigkeiten zur Ausführung von Code sowie zur Bewegung reduziert wird. Sonstige Anforderungen werden als erfüllt angenommen. Für ein gegebenes Code-Segment in Verbindung mit einem gegebenen Akteur wird die Kenntnis der *Worst Case Execution Time* angenommen. Der Code sei dabei durch jeden Akteur ausführbar. Hinsichtlich der Fähigkeit zur Bewegung betrachten wir zwei Freiheitsgrade: Die Beeinflussbarkeit der eigenen Position im Raum und die Vorhersagbarkeit künftiger Positionen im Raum.

Ein Spieler im Roboterfußball besitzt beispielsweise innerhalb des Spielfelds volle Positionierbarkeit, während ein Satellit in einer Umlaufbahn nur sehr beschränkte Möglichkeiten der Steuerung hat. Ein Navigationssystem kann die eigene Position hingegen nur über Ausgaben beeinflussen und hat damit nur einen indirekten Durchgriff. Die Vorhersagbarkeit künftiger Positionen im Raum ist bei einem Satelliten wiederum sehr einfach, während bei einem Fußballroboter bereits Unvorhersagbarkeiten in Form der Gegner oder ungenauer Aktorik einfließen. Im Gegensatz dazu kann die künftige Position eines Navigationsgeräts nur abgeschätzt werden, da es nicht vorhersehbar ist, in welcher Weise die Ausgaben umgesetzt werden. Eine einfache Kategorisierung der Akteure unter Nutzung dieser beiden Dimensionen illustriert Abbildung 2, wobei die Abgrenzungen zwischen den einzelnen Gruppen aber unscharf sind.

Eine Aufgabe besteht aus einem sequentiellen Ausführungsfaden, dessen Terminierung sichergestellt ist. Der Ausführungsfaden wird von einer Menge von Akteuren bearbeitet, wobei beliebige Unterbrechungen, Wiederaufnahmen sowie Migrationen zwischen Akteuren möglich sind. Entlang des Ausführungsfadens wird ein *Fortschritt* $p \in [0, 1]$ definiert, der beispielsweise über das Verhältnis von bereits ausgeführten Schritten (Instruktionen, logische Schritte, usw.) zur Gesamtzahl der Schritte bestimmt sein kann. Die Raum-Zeit-Bedingungen einer Aufgabe werden über eine Funktion $S : [0, 1] \mapsto C$ bestimmt, die den Fortschritt auf Raum-Zeit-Bedingungen abbildet. Dies erlaubt u. a. mit $S(0)$ und $S(1)$ Start- und Endbedingung zu spezifizieren. Ebenso können bei der Einschränkung von C auf den Zeitbereich reine Echtzeitaufgaben beschrieben werden. S ist beliebig, aber statisch. Aufgaben, deren Raum-Zeit-Bedingungen von vorangegangenen Schritten der Bearbeitung abhängen, werden in mehrere Aufgaben zerlegt. Abbildung 3 zeigt mehrere Raum-Zeit-Fenster einer Aufgabe, die mit dem Fortschreiten der Ausführung verknüpft sind. Dadurch ensteht eine Art Korridor in der Raum-Zeit, in der die Aufgabe ausgeführt werden darf.

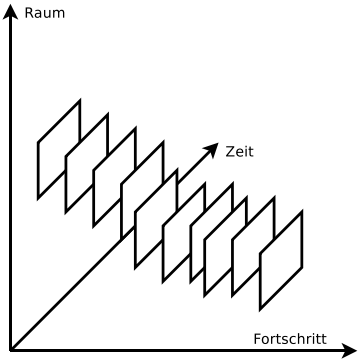


Abbildung 3: Raum-Zeit-Bedingungen einer Aufgabe entlang des Fortschritts.

Aufgaben können in beliebiger Weise voneinander abhängen, wobei wir dies in unserem Modell auf Raum-Zeit-Abhängigkeiten beschränken, die jeweils über Raum-Zeit-Bedingungen definiert sind. Eine solche Abhängigkeit $D : [0,1] \times [0,1] \mapsto C$ definiert für beide beteiligten Aufgaben relative Raum-Zeit-Bedingungen entlang der Fortschrittsverläufe beider Aufgaben, die zusätzlich zu den oben beschriebenen Raum-Zeit-Bedingungen beider Aufgaben zu erfüllen sind. Abhängigkeiten zwischen mehr als zwei Aufgaben werden durch mehrere bilaterale Raum-Zeit-Bedingungen beschrieben. Diese Abhängigkeiten erlauben es, komplexe Tätigkeiten, die insbesondere auch nebenläufige Elemente enthalten können, mit Hilfe mehrerer unserer einfachen Aufgaben, die jeweils nur aus einem sequentiellen Ausführungsfaden bestehen, zu beschreiben.

3 Vereinfachtes Modell

Das in Abschnitt 2 beschriebene generische Modell ist in der Lage, im Rahmen der beschriebenen Abstraktionen eine große Menge von Anwendungen hinsichtlich ihres Raum-Zeit-Schedulings zu beschreiben, wobei insbesondere auch Kooperation zwischen Akteuren beschreibbar ist, was in klassischen Tourenplanungsmodellen [GRW08] fehlt. Die Möglichkeit, Raum-Zeit-Bedingungen und Abhängigkeiten zwischen den Aufgaben über beliebige kontinuierliche Funktionen in mehreren Dimensionen zu beschreiben, führt jedoch zu einer sehr hohen Komplexität dieses allgemeinen Modells. Darum soll es uns als Ausgangspunkt für mögliche Vereinfachungen dienen, um somit Lösungen für bestimmte Schedulingprobleme der cyber-physischen Systeme in höchstens polynomialer Aufwand zu finden. Einen ersten Ansatz für ein solches vereinfachtes, aber immer noch relevantes Modell beschreiben wir im Folgenden.

Es wird nur eine Art von Akteuren betrachtet, die frei positionierbar und vollständig vorhersagbar sind, also innerhalb unserer Kategorisierung die größten Freiheiten besitzen. Probleme, die von stärker eingeschränkten Akteuren ausgehen, können damit unter Umständen trotzdem beschrieben werden, indem die Einschränkungen der Akteure als zusätzliche Raum-Zeit-Bedingungen für sämtliche Aufgaben beschrieben werden.

Hinsichtlich der Aufgaben liegen die Ansätze zur Vereinfachung in den Funktionen S und D. Eine einfache Version von S ist konstant, bildet also unabhängig vom Fortschritt stets auf die gleiche Raum-Zeit-Bedingung ab. Aufgaben, bei denen das nicht umsetzbar ist, können in der vereinfachten Version des Modells in eine Reihe von Einzelaufgaben zerlegt werden, die jeweils eine konstante Raum-Zeit-Bedingung erfüllen.

In der Echtzeitwelt wird in einfachen Modellen von unabhängigen Tasks ausgegangen. Im Bereich von mobilen cyber-physischen Systemen führt eine solche Einschränkung jedoch dazu, dass nahezu alle relevanten Anwendungen ausgeschlossen werden und infolge des großen Freiraums auch einfache Lösungen nicht übertragbar sind.

Somit vereinfachen wir dahingehend, dass D lediglich zu Beginn und Ende der Fortschrittsverläufe (also an den Stellen $(0,0)$, $(0,1)$, $(1,0)$ und $(1,1)$ Raum-Zeit-Bedingungen definiert. Damit wird die Komplexität stark gesenkt, ohne jedoch relevante Abhängigkeiten der Art, dass die Aufgaben zur gleichen Zeit am gleichen Ort enden müssen oder dass eine Aufgabe an dem Ort zu der Zeit beginnen muss, zu dem/der eine andere Aufgabe abgeschlossen wurde, auszuschliessen.

Diese Vereinfachungen schränken die Vorteile des generischen Modells für die ausgewählte Klasse von Akteuren nur unwesentlich ein, da Aufgaben geeignet in kleinere Teile zerlegt werden können und damit eine Diskretisierung der im vollständigen Modell kontinuierlichen Raum-Zeit-Bedingungen erreicht wird. Hinsichtlich der Akteure schließt die Einschränkung eine Reihe von Anwendungen aus, so dass Lösungen nur noch unter bestimmten Annahmen und Einschränkungen möglich sind.

4 Zusammenfassung und Ausblick

In diesem Beitrag haben wir einen Vorschlag für ein generisches Modell für Raum-Zeit-Schedulingprobleme cyberphysischer Systeme präsentiert. Ausgehend von diesem Modell haben wir eine Vereinfachung abgeleitet, die die Komplexität deutlich senkt, aber dennoch für die Beschreibung vieler relevanter Anwendungen geeignet ist.

Die Fortsetzung dieser Arbeit zielt in mehrere Richtungen: Zum einen wollen wir Anwendungen hinsichtlich des generischen Modells detailliert klassifizieren und auf dieser Basis eine verfeinerte Vereinfachung ableiten. Zum anderen wollen wir konkrete Scheduling-Algorithmen auf Basis des vereinfachten Modells entwickeln. Diese Algorithmen sollen auf Erkenntnissen sowohl aus dem Echtzeitscheduling als auch der dynamischen Tourenplanung [Auf10, GI05] basieren.

Literatur

[Auf10] Björn Aufderheide. *Dynamische Tourenplanung : Übersicht und Stand der Forschung*. Diplomica, Hamburg, 2010.

[GI05] Tore Grünert und Stefan Irnich. *Optimierung im Transport*. Shaker, Aachen, 2005.

[GRW08] Bruce L. Golden, S. Raghavan und Edward A. Wasil, Hrsg. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer Science+Business Media, New York, 2008.

[Liu00] Jane W. S. Liu. *Real-Time Systems*. Prentice Hall, Englewood Cliffs, 2000.

Location-based Access Control Providing One-time Passwords Through 2D Barcodes

Mirco Schönfeld, Martin Werner, Florian Dorfmeister

Mobile and Distributed Systems Group
Ludwig-Maximilians-University Munich
mirco.schoenfeld@ifi.lmu.de
martin.werner@ifi.lmu.de
florian.dorfmeister@ifi.lmu.de

Abstract: Location-based Access Control often relies a location proof, which ensures that a specific user is at a specific location. These locations are most often inferred from measurements. As a consequence, such systems are never trustworthy. An attacker can simply fake sensor measurements or even sensor data in absence of a trusted measurement module. Moreover, the measurement data is typically stable over time at a fixed location and can thus be replayed at later times. With this paper, we propose a system, which can provide functionality for a location proof, which does not rely on measurements and does not suffer from replay attacks. Therefore, a self-contained system is generating signed one-time passwords and communicates them via 2D barcodes for authentication of camera-enabled devices being in a specific location.

1 Introduction

An attractive and fast-growing business area in mobile context is tying certain services to specific geographic locations. Such services often provide information to which access is granted if and only if the user is physically present at a specified location. Many of these location-based services rely heavily on the correctness of location information. The verification of such information presents an active area of research, because there are certain scenarios in which tampering with sensor data to fake locations is feasible.

Moreover, for the use-case of location-based access control it is often much simpler to couple access control with interaction. Especially, if access control points are rare. A common technique to provide such functionality is given by Near-Field-Communication [ABPW07, AK06, GS08]. However, Near-Field-Communication is not implemented in recent Apple devices and hence does not allow for a "Bring-Your-Own-Device" philosophy.

If sensor data should not be trusted, infrastructure-based localization is needed, which usually introduces severe privacy risks. Therefore we can conclude, that only an architecture, which provides proximity or location information in line with the user's intention can be successful. The proposed system consists of a backend service the user wants to authenticate with and a small computing unit that is placed at a specific geographic location. To activate a computing unit against our backend service a public/private key pair is created inside this computing unit and the associated public key is registered in the backend service. After activation the unit should be physically bound to a fixed location.

For each user requiring access to the location-based information the computing unit constructs a one-time password. For this purpose it uses the cryptographic hash function SHA-1 to generate a digest from an internal counter and a timestamp. This message digest is then encrypted with the unit's private key. The resulting one-time token consists of the plain-text representation of both the counter and the timestamp together with the message digest and is transmitted to the user via a QR code. To gain access the user authenticates himself to our backend service using this one-time token. As only the backend service is able to verify the integrity of the token and as the counter effectively prohibits replay attacks, the system is secure. Moreover, the intention of the user gets expressed by scanning a visual code and hence, privacy problems occuring due to tracking of people without their attention and permission are absent.

In this paper we present a prototypical implementation of our architecture based on the open-source prototyping platform Arduino [Ard]. For transmitting the authentication token we employ QR-codes. We therefore equipped the Arduino with a LCD display, which is capable of displaying such two-dimensional barcodes. In contrast to using NFC for the token transmission merely all smartphones ship with the required hardware: a sufficient camera.

2 Implementation Details

As an appropriate basis of our prototypical implementation we employed the Arduino Mega2560. Arduino is an open-source prototyping platform that ships with a complete toolchain for implementing and running C code. The Mega2560 is run by an Atmel ATMega2560 microcontroller at 16MHz that is part of the Atmel AVR microcontroller-family. It is capable of calculating signature digests in a reasonable amount of time.

Figure 1: Arduino Prototype With LCD Display Attached

The Arduino-based controller was equipped with a SainSmart 3.2" TFT LCD Display module driven by an SSD1289 display controller, where a display interface library is readily available [UTF].

The open source software *qrencode* was modified to run on the Arduino and to display QR-codes via said library. Cryptographic routines are based on Colin Plumb's BigNum libary [BNL], which basically provides modular exponentiation of big integers to systems with CPUs supporting 32 bit integer data natively.

As for hashing, the well-known and widely used SHA-1 digest was used, though it is known to have some smaller weaknesses. In 2005, Wang et al. successfully attacked an SHA-1 digest with fewer than 2^{69} hash calculations [WYY05], which they were able to lower to 2^{63} calculations only a few months later [SHA]. Thereby, they reduced the theoretical bound of 2^{80} hash calculations of a brute-force-attack. However, we consider this acceptable for our prototypical implementation as the hashing algorithm could be easily replaced later on.

The system basically runs in a loop. However, in the initialization phase, a public/private key pair is generated and the public key is displayed using a QR-code. This QR-code can then be scanned and transmitted to a backend, where this one-time password generator is then accepted for certain service authentications. The private key is held in the device. For high-security applications, this initialization scheme should of course be implemented using a tamper-proof module or a smart card. However, a physical shielding of the device is sufficient in most situations.

Now, the system runs in a loop with configurable delay, incrementing the internal counter at each iteration, generating a signed one-time password from this counter and a timestamp, and showing this information as a QR-code readable by almost any smartphone. By transmitting this signed one-time token to a backend, the service is able to uniquely identify any known unit and trigger an appropriate response - the backend decides whether or not to grant access. A simplified scheme is given in figure 2.

In this way, location-based authentication can be triggered on a backend without an interconnected infrastructure. Neither active nor passive localization is needed, nor an Internet connection of any device except for the device, which wants to authenticate to an online service, of course.

Figure 2: Simplified Authentication Scheme

3 Use-Cases

The possible application of our architecture is diversified. It could be used to grant physical access to a restricted area since the computing unit is uniquely identifiable through its public/private key pair. Therefore, the geographic location of each unit has to be provided together with the public key in the initialization phase. Additionally, each user has to be registered with the backend through a seperate public/private key pair, too. The one-time token that has been extracted from the QR-code is signed by the user's private key before being transmitted to the backend. Thereby, the backend can both identify the user and the location that the user tries to gain access to. And thereby, this scheme can be incorporated into a fine-grained, centrally manageable room access control system.

Another use-case is mobile payment. A gas pump at a gas station for example encrypts all necessary payment details into a signed QR-code. The user decodes the QR-code and transmits the signed token to a backend. The server verifys both the gas pump and the user and triggers a payment process after successful verification.

4 Conclusion

With this paper, we presented a framework for presence-based authentication with respect to Internet services, while the authentication device does not need on any network connection.

Moreover, we developed a prototyping platform for display-based communication and gave a working proof-of-concept implementation of our approach. From a hardware perspective, the framework only needs a microcontroller, which is able to calculate hashes and perform RSA encryption and a diplay to transmit a digest via QR code. This display can easily be replaced by any near-field communication technology such as NFC or Bluetooth. However, using a display leads to authentication with a clear intention of the user. It is rather unlikely to accidentally scan a matrix code as compared to performing an NFC touch.

In future work, this will be integrated into a middleware supporting NFC, Bluetooth and other wireless personal area networks for transmission of said access tokens aiming at transparent support of interaction-based authentication for online services using smartphones, including NFC-enabled ones and rather simple ones.

References

[ABPW07] Y. Anokwa, G. Borriello, T. Pering, and R. Want. A user interaction model for NFC enabled applications. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*, pages 357–361. IEEE, 2007.

[AK06] Z. Antoniou and D.N. Kalofonos. NFC-based mobile middleware for intuitive user interaction with security in smart homes. In *Proceedings of IASTED CSN'06*, 2006.

[Ard] Arduino Open Source Electronics Prototyping Platform. Online, last accessed: 17.10.2012. `http://arduino.cc/`.

[BNL] bnlib - An SDK for Big Number Arithmetic. Online, last accessed: 17.10.2012. `http://philzimmermann.com/EN/bnlib/index.html`.

[GS08] K. Griffin and C. Stone. Near Field Communication Activation and Authorization, 2008. US Patent App. 12/241,557.

[SHA] New Cryptanalytic Results Against SHA-1. Online, last accessed: 17.10.2012. `http://www.schneier.com/blog/archives/2005/08/new_cryptanalyt.html`.

[UTF] UTFT Library. Online, last accessed: 17.10.2012. `http://henningkarlsen.com/electronics/library.php?id=52`.

[WYY05] X. Wang, Y. Yin, and H. Yu. Finding collisions in the full SHA-1. In *Advances in Cryptology–CRYPTO 2005*, pages 17–36. Springer, 2005.

Lokalisierung mobiler Roboter mittels RFID-NaviFloor®

André Heller, Christof Röhrig

Fachbereich Informatik
Fachhochschule Dortmund
Emil-Figge-Str. 42
44227 Dortmund
andre.heller@fh-dortmund.de
christof.roehrig@fh-dortmund.de

Abstract: In diesem Beitrag wird eine Möglichkeit beschrieben, wie mobile Roboter mittels RFID-Technologie lokalisiert und navigiert werden können. Für diesen Zweck wird ein RFID Boden(NaviFloor®) der Firma Future-Shape verwendet, der aus einem bestimmten Raster von RFID-Tags besteht. Der Roboter ist mit einem RFID-Reader ausgestattet und liest die entsprechenden IDs der Tags beim überqueren ein. Durch die initiale Zuteilung von ID und Position kann die absolute Pose, während der Fahrt des Roboters, bestimmt werden. Darüber hinaus werden die Informationen von weiteren Sensoren zwischen den Tags erfasst und fließen mit in die Berechnung ein. Die Zustandsschätzung erfolgt durch den Monte Carlo Partikelfilter, um die Pose zwischen den einzelnen Tags zu schätzen. Aufgrund der einmaligen Installation des NaviFloors ® benötigt der Roboter lediglich ein RFID-Lesegerät für eine globale Lokalisierung. Durch den Einsatz der RFID-Technologie kann, bei der Lokalisierung mobiler Roboter, auf teure Sensorik verzichtet werden, wodurch die Kosten des Projekts minimiert werden.

1 Einführung

Die Entwicklungen und Möglichkeiten der Lokalisierung von mobilen Robotern hat in den letzten Jahren immer weiter zugenommen. Einer der Gründe dafür ist, dass Roboter nicht mehr auf statischen Strecken fahren, sondern sich kollisionsfrei in einem Arbeitsraum bewegen sollen. Die Ausgangsfrage für eine genaue Lokalisierung lautet: "Wo bin ich?". Um diese Frage zu beantworten, werden Sensorinformationen vom Roboter und von der Umgebung benötigt. Die Informationen über die Bewegung des Roboters werden beispielsweise mittels Radencodern oder zusätzlichen externen Sensoren ermittelt. Eine Auskunft über die Umwelt des Roboters liefern beispielsweise Laserscanner, Kameras oder einlesbare Markierungen an den Wänden oder am Boden. Um die Pose des Roboters zu bestimmen können zwei unterschiedliche Lokalisierungsprinzipien angewendet werden. [1]

Bei der lokalen Lokalisierung ist die Startpose des Roboters bekannt. Mit Hilfe von kontinuierlichen Messungen der Odometrie-Sensoren, kann die Pose bei jedem Abtastschritt geschätzt werden. Bei dieser Methode summieren sich schon geringe Abweichungen der

Sensoren über die Zeit, was eine genaue Lokalisierung unbrauchbar macht. Für kleine Strecken oder für eine Lokalisierung mit geringer Genauigkeitsanforderung ist dieses Verfahren eine effektive und günstige Lösung. [1]

Bei der globalen Lokalisierung ist die Startposition des Roboters der Umwelt unbekannt. Für eine Lokalisierung des Roboters müssen erst signifikante Umgebungsmerkmale erfasst werden, die der Roboter mittels Sensoren erfasst. Durch den Einsatz eines RFID-Boden, kann die absolute Position des Roboters ermittelt werden, wenn dieser ein RFID-Tag überquert. Lediglich die Orientierung kann erst bestimmt werden, wenn der Roboter mehr als einen Tag überquert. [1] [9] [10]

Eingesetzt wird der RFID-Bodens beispielsweise schon in Krankenhäusern und Pflegeheimen, um Kosten zu senken. In diesen Einsatzgebieten werden häufig Transportdienste verwendet. Beispiele für Transportdienste sind z.B. das Ausgeben von Essen, die Verteilung von Medikamenten oder das Waschen von Wäsche. Transportroboter können sich über den NaviFloor Lokalisieren und Informationen vom Patienten weiterleiten. Zusätzlich können weitere Gegenstände wie Betten oder Rollstühle mit RFID-Lesegeräten ausgestattet werden, damit deren Position im Gebäude bekannt ist.[4] [2]

Darüber hinaus können sich mehrere Roboter über den NaviFloor bewegen und die eingelesenen IDs an einem zentralen Server weiterleiten, der die Roboter navigieren kann, um Kollisionen zu vermeiden. [10]

1.1 RFID-Technik

Mittels RFID-Technologie können Objekte und Personen über eine eindeutige Identifikationsnummer eingeordnet werden. Laut VDI-Richtlinie 4416 besteht ein vollständiges RFID-System aus folgenden Komponenten:

- Transponder (Tag), Datenträger
  Dieser wird an dem zu identifizierenden Objektes oder an einer bestimmten Position im Raum angebracht. In diesem Projekt sind die Tags in einem Raster von 50x50 cm auf dem NaviFloor aufgebracht.
- Empfangsgerät, RFID-Lesegerät (als Lese- und Schreibgerät erhältlich)
  Die Lesegeräte versorgen die entsprechenden RFID-Tags mit Strom, wodurch die Tags ihre Identifikationsnummer senden. Die gesendete ID wird vom Lesegerät empfangen und kann dann weitergeleitet werden.
- Informationssystem
  Das Informationssystem verarbeitet die eingelesenen IDs und hat Informationen der jeweiligen Identifikationsnummer gespeichert, z.B. die absolute Position des Tags.

[2]

Energieversorgung

Die Energieversorgung der RFID-Tags unterteilt sich in der aktiven und passiven Versorgung. Die passiven Transponder verfügen selbst über keine eigene Stromversorgung und müssen sich deshalb die Energie aus den Funkwellen des Lesegerätes beziehen. Die aus dem Lesegerät empfangene Energie wird dazu benötigt, damit sich die Tags aktivieren und die hinterlegten Informationen senden.

Passive Transponder besitzen eine Antenne, die als Spule fungiert und Kondensatoren auflädt, um die Stromversorgung sicher zu stellen. Dabei ist zu beachten, dass die Versorgung des Lesegeräts kontinuierlich übertragen werden muss, damit die Übertragung fehlerfrei funktioniert. Passive Tags sind kleiner und können kostengünstiger hergestellt werden, weshalb dieser Versorgungstyp im NaviFloor verwendet wird.

Aktive RFID-Transponder werden durch eine Batterie betrieben und haben eine größere Reichweite ihre Daten zu senden. Dadurch vergrößert sich auch die Bauform und die Kosten des Tags. Da im NaviFloor große Mengen an RFID-Tags gebraucht wird und eine kleine Bauform besitzen muss, ist dieser Tag-Typ der Energieversorgung nicht optimal. [2] [4]

Datenmenge

Die Datenmenge eines RFID-Tags definiert die zurückgegebenen Informationen, nachdem die Stromversorgung durch ein Lesegerät gewährleistet wird. Die eingebetteten RFID-Tags liefern eine entsprechende Antwort, die aus der *MessageLength*, der *DataLength* und der einmaligen *ID* des Tags besteht. Eine Reihe von RFID-Tags liefern nur eine Ein-Bit-Nachricht zurück, um die Existenz in einem bestimmten Bereich festzustellen. Diese Variation wird beispielsweise in Einkaufsläden benutzt, um festzustellen ob Ware am Ausgang bezahlt wurde oder nicht. [2]

Frequenzbereiche & Reichweite

Eines der wichtigsten Unterscheidungsmerkmale ist der Frequenzbereich der RFID-Systemen. Anhand der Wahl der Frequenz kann die Reichweite und Leistungsfähigkeit festgelegt werden, da die Differenz in den unterschiedlichen Eingenschaften der elektromagnetischen Wellen liegt.

Es gibt drei Bereiche, in den die Frequenzen für ein RFID-System eingeteilt werden können.

- LF-Bereich (low frequency): 30kHz bis 300kHz
- HF-Bereich (high frequency): 3MHz bis 30MHz
- UHF-Bereich (ultra high frequency): 300MHz bis 3GHz

Die verbauten RFID-Tags im NaviFloor senden im HF-Bereich mit 13,56 MHz und besitzen eine Reichweite von bis zu 10 cm zwischen Tag und Lesegerät. Die Reichweite variiert je nach Bodenbelag, durch die die Tags ihre Informationen zurücksenden müssen. Das Übertragungsverfahren nennt sich induktive Kopplung, bei der der Transponder die Energie durch Induktion bezieht. Außerdem wird ein Signal erzeugt, um die Daten vom Transponder zum Lesegerät übertragen zu können.

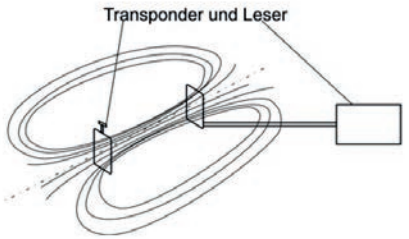


Abbildung 1: Induktive Kopplung zwischen zwei Antennen, [2]

[2]

2 Aufbau & Umgebung

iRobot Create®

Zur Realisierung des Projekts wird ein mobiler Roboter der Firma iRobot eingesetzt. Diese spezielle Version des Roboters wird für die Entwicklung und Forschung eingesetzt und verfügt, im Gegensatz zu den anderen Modellen, über keine Saugeinheit. Der Roboter besitzt auf der Oberseite einen *Cargo Bay Connector*, um die Daten der Sensoren auszulesen und Steuerbefehle zum Roboter zu senden. [5]

Für die Lokalisierung ist es notwendig mehrere Sensoren des iRobot auszulesen, um Kollisionen zu vermeiden und die Position zu bestimmen. Zur Kollisionserkennung besitzt der Roboter an der Vorderseite einen Bumper, mit dem Kollisionen festgestellt werden können, wenn der Roboter gegen Hindernisse fährt. Über die Radencoder werden anhand der Radumdrehungen die zurückgelegten Distanzen der einzelnen Räder erfasst, um die Pose des Roboters berechnen zu können. [5] [6]

Arduino-Board

Für die Erfassung und Verarbeitung von Sensordaten wird eine Arduino-Plattform verwendet, die mehrere digitale und analoge Ein- und Ausgänge besitzt und über den Mikrocontroller angesteuert wird. Über die Entwicklungsumgebung wird der ATmega328 Mikrocontroller so programmiert, dass das Board die Informationen von den angeschlossenen Komponenten erfasst und verarbeitet. Über mehrere Software- UARTs werden die Komponenten angesprochen und die Informationen verarbeitet, um diese dann an einen PC zu senden. [7]

Bluetooth-Modul

Um die verarbeiteten Daten von dem Arduino-Board an den PC zu senden, wird ein Bluetooth-Modul von der Firma Stollmann verwendet. Das ausgewählte Modul, BlueMod+B20, besitzt eine integrierte UART-Schnittstelle, mit der die Daten einfach über eine Bluetoothverbindung zum PC gesendet werden können.[8]

NaviFloor® & RFID-Reader

Der NaviFloor® der Firma Future-Shape ist ein RFID-Underlay, der für die Navigation von Roboter entwickelt wurde. Der NaviFloor® ist ein Underlay und kann daher unter Laminat, Teppich oder Kunstharzböden verlegt werden und bietet damit ein breites Spektrum an Einsatzgebiete. Die RFID-Tags werden in einem Raster von 50x50 cm integriert, sodass eine realtiv häufige Überquerung des Roboters eine gute Positionsermittlung ergibt. [4]

Wenn die passiven Tags über ein elektromagnetisches Feld ausgelesen werden, liefern diese eine eindeutige Identifikationsnummer zurück, welche eine Position repräsentiert. Im Vorfeld muss jeder Tag per Hand ausgelesen und deren Position in einer Datenbank abgelegt werden. Somit kann bei jeder Überschreitung eines Tags die Position des Roboters bestimmt werden. [3] [2]

Als RFID-Lesegerät wird das SkyeModule M1 von der Firma *skyetek* verwendet. Dieses ist für 13,56 MHz Tags entwickelt worden und unterstützt mehrere RFID-Protokolle. Die geringen Maße von 38mm x 40mm x 4mm erlauben es, das Gerät an der Unterseite des Roboters anzubringen und über die integrierte UART-Schnittstelle mit dem Arduino-Board kommunizieren zu lassen. [3]

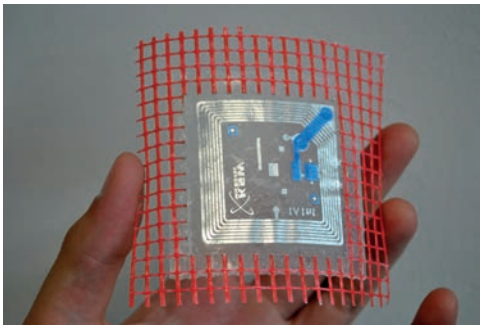


Abbildung 2: RFID-Tag aus dem NaviFloor®

Verarbeitung und Darstellung

Der komplette Aufbau des Projekts besteht aus der Kombination der einzelnen Komponenten. Das Bluetooth-Modul und das RFID-Lesegerät wird an das Arduino-Board angeschlossen und mit dem iRobot Create® verbunden. Das Arduino-Board wird im Vorfeld so programmiert, dass es die Daten des Roboters und des RFID-Readers in regelmäßigen Abständen abfragt und per Bluetooth an dem PC sendet.

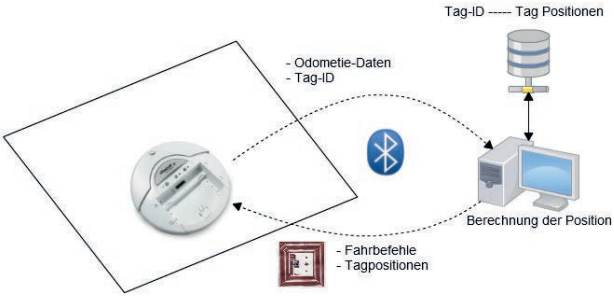


Abbildung 3: Übersicht des Aufbaus

Die Verarbeitung und Darstellung der Informationen wird über das Programm *MATLAB* realisiert. Vorab wurden die einzelnen RFID-Tags per Hand ausgelesen und die Identifika-

tionsnummer einer Position zugeordnet und in einer Tabelle gespeichert. Ein MATLAB-Skript empfängt die übertragenden Bluetooth-Daten vom Arduino-Board und berechnet anhand der Odometrie-Daten und der Tag-IDs die geschätzte Position.[1] [3]

Zur Veranschaulichung stellt die entwickelte GUI den Raum dar, in dem der Teppich verlegt wurde und die unterschiedlichen Wege des Roboters aus den realen und den geschätzten Daten. Die GUI kann auch zur Steuerung des Roboters genutzt werden, indem die spezifischen Fahrbefehle des iRobot Create® an das Arduino-Board übertragen werden und dieses die Befehle an den Roboter weiterleitet. [8] [5]

3 Entwicklung

Lokalisierung

Das Ziel des Projekts ist es anhand der RFID-Tags eine globale Lokalisierung des Roboters sicherzustellen. Für eine globale Lokalisierung werden Informationen der Umwelt benötigt, welches die Zuordnung von Position und Identifikationnummer der RFID-Tags liefert. Bei jeder Überfahrt eines Tags kann eine globale Position bestimmt werden. Dazwischen jedoch werden die Odometriedaten des Roboters genutzt, um eine Positionsschätzung durchzuführen[1]. Die Odometriedaten der Radencoder sind aber fehlerbehaftet, sodass keine genaue Lokalisierung möglich ist. Durch unterschiedliche Untergründe oder einem unterschiedlichen Reifendurchmesser weichen die zurückgelieferten Werte von den realen stark ab. [6]

Um diesen Fehler zwischen den RFID-Tags zu minimieren wird eine Zustandsschätzung durchgeführt, die anhand der Sensorwerte, den Abweichungen und den Stellgrößen berechnet wird. [1] [9] [10] [11]

Arduino-Board Aufgaben

Das Arduino-Board ist die Schnittstelle zwischen dem iRobot Create® und der Steuerung und Verarbeitung des PCs. Beim Start werden die initialen Einstellungen für die Sensoren-Schnittstellen und des mobilen Roboters vorgenommen. Eine der Aufgaben ist die Kollisionsverarbeitung, wenn der Roboter gegen ein Gegenstand fährt. Wenn eine Kollision über den Bumper entsteht, fährt der Roboter eine definierte Distanz rückwärts, dreht sich auf der Stelle und nimmt dann wieder eine Geradeausfahrt auf. Für diesen Ablauf benötigt der iRobot Create® bestimmte Steuerbefehle, die das Arduino-Board über die UART-Schnittstelle an den Roboter überträgt. [6] [7]

Desweiteren werden in regelmäßigen Abständen von ca. 130 ms alle Sensoren nacheinander abgefragt. Dazu gehören:

- Steuerbefehle vom PC (Bluetooth-Modul)
- Odometriedaten vom Roboter
- Bumper
- RFID-Reader

Diese werden dann in über ein Protokoll per Bluetooth weitergeleitet und über die Matlab GUI dargestellt.

Lokalisierungsverfahren

Zur Zustandsschätzung wird der Monte Carlo Partikelfilter verwendet, der es ermöglicht mit Hilfe von Partikeln eine Approximation des Zustand und dessen Wahrscheinlichkeit in dem Zustandsraum zu bestimmen. Mit der zuvor erstellten nichtlinearen Systemfunktion und einer Menge von endlichen Partikeln, können diese als mögliche Position des Roboters angenommen werden. [1]

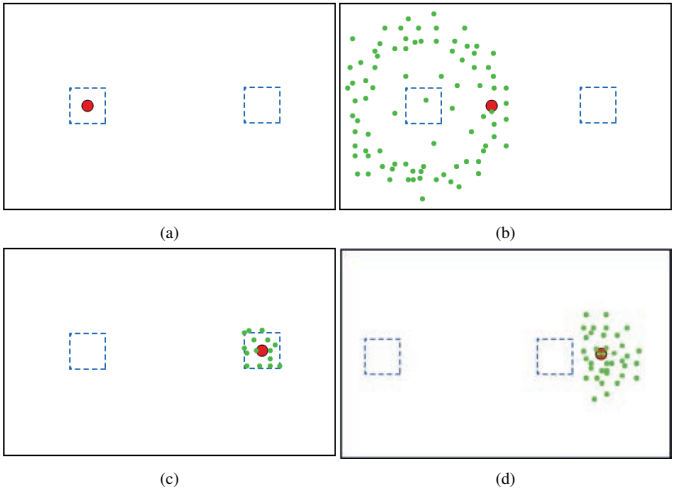


Abbildung 4: Darstellung der Partikel bei Überquerung der Tags

Am Anfang ist die Verteilung der Partikelmenge so verteilt, dass jeder Partikel eine Position darstellen kann und dieselbe Wahrscheinlichkeit besitzen. Beim Start kann der Roboter jede Position im Raum besitzen, bis der Roboter ein RFID-Tag überquert, wodurch die

zugeordnete globale Position bestimmt wird (siehe Abb. 4a). Dabei verdichten sich die Partikel um den überfahrenden RFID-Tag.

Im nächsten Schritt wird die Prädiktion und ein Aktualisierungsschritt durchgeführt, bei der für jedes Partikel die Zustandsübergangsfunktion mit Rauschen angewandt wird. Das Rauschen ist abhängig von der jeweiligen Varianz, die bei den Zuständen und dem Messvorgang auftreten können.

In Abbildung 4b konnte die globale Position bestimmt werden, jedoch nicht die Orientierung. Diese kann erst berechnet werden, wenn der Roboter mindestens zwei RFID-Tags überquert hat. Deswegen verteilen sich die Partikel im Aktualisierungsschritt in alle Richtungen, bis in Abbildung 4c durch das Einlesen eines weiteren Tags die Orientierung berechnet werden kann. Die neue Position wird mit jedem Partikel verglichen und ‚je nachdem wie groß die Differenz ist, bekommen die Partikel eine gute bzw. eine schlechte Gewichtung. Die Partikel mit einer geringen Differenz, zwischen Erwartungswert und der Messung, bleiben für die weitere Berechnung erhalten. Die schlechten jedoch müssen durch gute Partikel ersetzt werden, sodass sich durch einen gewichteten Mittelwert die neue geschätzte Position anhand der Partikeldichte berechnet werden kann. Durch die Bestimmung der Orientierung verteilen sich die Partikel im nächsten Aktualisierungsschritt in Richtung der tatsächlichen Fahrtrichtung (siehe Abb. 4d). Allerdings verteilen sich die Partikel aufgrund der großen Varianzen der Odometriedaten immer weiter von der tatsächlichen Position des Roboters, welche sich erst wieder verdichten, wenn ein weiterer RFID-Tag eingelesen wird. [1]

4 Auswertung

In Abbildung 5 ist die Matlab-GUI dargestellt, die eine Testfahrt des Roboters zeigt. Es werden verschiedene Trajektorien auf der Karte visualisiert, die aus verschiedenen Positionsdaten basieren. Die gelben Marken stehen für die Positionberechnung unter Verwendung der Stellgrößen, die an den Roboter gesendet werden. Durch Unebenheiten, Untergrundbeschaffungen und der Radencoderunsicherheit bewegt sich der Roboter nicht identisch anhand der Steuerbefehle.

Die Positionen aus den verrauschten Steuerbefehlen werden in grün dargestellt. Diese werden abhängig von den gegebenen Varianzen verrauscht und für den Zustandsschätzer benötigt, um ein Vergleich der Messwerte vorzunehmen. Durch die wiederholende Abfrage und Verarbeitung der Odometriedaten werden die berechneten Positionen rot dargestellt. Zu sehen ist, dass sich die Daten der verschiedenen Quellen unterscheiden und der Zustandsschätzer anhand dieser Daten und den dazugehörigen Varianzen eine Schätzposition errechnet.

Die Partikel des Monte Carlo Partikelfilters werden durch blaue Punkte repräsentiert, die die geschätzte Pose des Roboters durch die Partikeldichte bestimmt. Die geschätzte Pose wird als schwarzes Kreuz dargestellt und berechnet sich aus einem gewichteten Mittelwert.

Die Abbildung 5 stellt eine Roboterfahrt dar, die in (0,0) startet und in (-500,-1000) endet, wobei die Maßeinheiten Millimeter sind. Während der Fahrt überquert der Roboter insge-

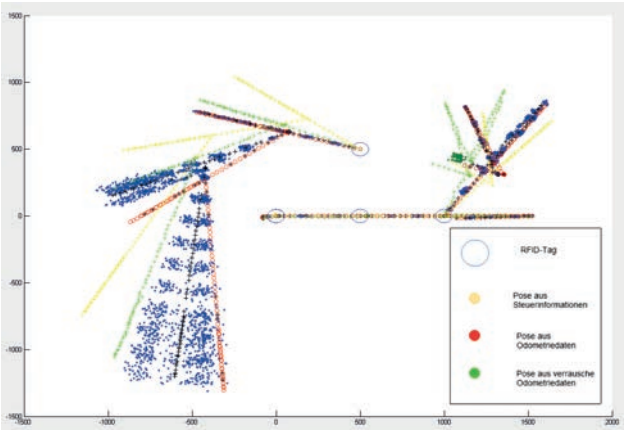


Abbildung 5: Matlab-GUI: Darstellung der Roboterbewegungen

samt vier RFID-Tags,die als große blaue Kreise abgebildet werden, die im Raster 500mm x 500mm in den Underlay eingearbeitet wurden. Der Roboter bewegt sich in diesem Beispiel solange Geradeaus, bis er auf ein Hindernis stößt, fährt eine kleine Distanz zurück und dreht sich auf der Stelle. Danach nimmt der Roboter wieder die Geradeausfahrt auf.

Vom Start bis zum ersten Hindernis überquert der Roboter insgesamt drei RFID-Tags und versucht durch die Drehung das Hindernis auszuweichen. Nach der zweiten Kollision mit dem Hindernis sind die summierten Fehler durch die Odometrie so groß, dass die reale Position nicht mehr mit der geschätzten übereinstimmt. Der Roboter überquert in (500,500) ein RFID-Tag, somit ist die reale Position bekannt. Die Schätzung der Position lag jedoch bei ca (1000,500). Für den weiteren Verlauf wird die neue Position des RFID-Tags in die Berechnung aufgenommen und die Partikel werden neu um den Tag verteilt. Wenn längere Zeit kein Tag eingelesen wurde, verteilen sich die Partikel immer weiter im Raum, was eine genaue Lokalisierung erschwert. Zu beobachten ist, dass bei der Drehung des Roboters die größten Unsicherheiten entstehen, weil die Ansteuerung der Räder nicht genau genug ist. Die Unsicherheit kompensiert sich erst wieder, bei der erneuten Überquerung eines RFID-Tags. [1]

5 Ausblick

Das Projekt bietet viel Potential für Verbesserungen und Weiterentwicklungen. Ein Vorschlag wäre die automatische Kartenerstellung des Raums. Durch die fest definierte Rastergröße des NaviFloors® kann der mobile Roboter eine bestimmte Fahrstrategie abfahren und über das RFID-Lesegerät die Identifikationsnummern der Tags Positionen zuordnen. Voraussetzung dafür ist, dass die initiale Ausrichtung des Roboters bekannt ist.

Zur Zeit werden die Sensordaten an den PC gesendet, der den Monte Carlo Partikelfilter anwendet und dem Roboter neue Steuerbefehle zurücksendet. Von Vorteil wäre es, wenn die ganze Lokalisierungsberechnung auf den Roboter selbst von statten geht, um den Roboter autonom fahren zu lassen. Da der Partikelfilter viel Rechenleistung benötigt, wäre eine Alternative andere probabilistische Verfahren zu benutzen, wie z.B. den Erweiterten Kalmanfilter. Mit Hilfe dieser Verfahren können die Daten direkt auf dem Mikrocontroller verarbeitet werden und müssen nicht mehr über eine kabellose Verbindung auf einem PC übertragen werden. Auf diese Weise bewegt sich der Roboter autonom durch den Raum, nachdem er eine Zielkoordinate oder eine Streckenfahrt übergeben bekommt.

Eine der großen Fehlerquellen, während der Lokalisierung, ist die Orientierungsänderung bei einer Drehung des Roboters. Schon geringe Messfehler der Radencoder bewirken eine größere Abweichung der Orientierung. Um diese Abweichung zu kompensieren, können weitere Sensoren genutzt werden, die die Orientierungsänderung messen. Solche Gyroskope messen die Drehgeschwindigkeit und können zur Errechnung der Orientierung mit in die Berechnung hinzugenommen werden.

6 Zusammenfassung

Zusammenfassend kann gesagt werden, dass diese Methode der Lokalisierung eine kostengünstige Lösung ist, im Vergleich zu teuren Sensoren wie ein Laserscanner. Wenn angenommen wird, dass sich mehrere mobile Roboter auf dem ausgelegten NaviFloor® bewegen, benötigt jeder Roboter lediglich einen RFID-Reader, um die Positionen zu bestimmen. Zudem ist die benötigte Rechenleistung für die Aufnahme der Tag-Informationen sehr gering, im Gegensatz zu Laserscannern oder einer kamerabasierte Lokalisation. Als Ergänzung zu den Radencodern, sollte ein Gyroskop-Sensor für die genaue Messung der Orientierungsänderung eingesetzt werden, damit die Ungenauigkeiten bei einer Drehung kompensiert werden.

Abbildungsverzeichnis

Literatur

[1] Probabilistic Robotics (Intelligent Robotics and Autonomous Agents),
Sebastian Thrun, Wolfram Burgard, Dieter Fox, The Mit Press, 2005

[2] RFID-Handbuch: Grundlagen und praktische Anwendungen von Transpondern, kontaktlosen Chipkarten und NFC,
Klaus Finkenzeller, Hanser Verlag, 2008

[3] RFID-Reader SkyeModule M1,
http://www.skyetek.com/ProductsServices/Modules/SkyeModuleM1/tabid/82/Default.aspx, 15.10.2012

[4] FutureShape NaviFloor®,
http://www.future-shape.de/de/technologies, 15.10.2012

[5] iRobot Create® Manual, *iRobot Create® Owner's Guide*

[6] Hacking Roomba,
Tod E. Kurt, ExtremeTech, 2006

[7] Arduino Uno Dokumentation,
http://www.arduino.cc/en/Main/arduinoBoardUno, 15.10.2012

[8] Stollmann B20-Bluetooth-Modul,
http://www.stollmann.de/de/module/bluetooth-produkte/bluemod-b20bt21.html, 15.10.2012

[9] "Reader antennas' configuration effects for two wheeled robots on floor-installed RFID infrastructure - analysis of forward-backward configuration effect -,"*Kodaka, K.; Sugano, S.*,Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on , vol., no., pp.5718-5724, 18-22 Oct. 2010

[10] „Autonomous Mobile Robot Navigation Using Passive RFID in Indoor Environment," *Sunhong Park; Hashimoto, S.*, Industrial Electronics, IEEE Transactions on , vol.56, no.7, pp.2366-2373, July 2009

[11] "Mathematical formulation of RFID tag floor based localization and performance analysis for tag placement," *Youngsu Park; Je Won Lee; Daehyun Kim; Jae Jin Jeong; Sang-woo Kim*, Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on , vol., no., pp.1547-1552, 7-10 Dec. 2010

Lessons Learned: Mobile und Selbstorganisierende Kommunikations- und Datenplattform für Einsatzkräfte im Projekt SpeedUp

Volkmar Schau[1], Christian Erfurth[2], Karsten Krohn[3], Steffen Späthe[1], Wilhelm Rossak[1]

[1]Friedrich-Schiller-Universität Jena
Fakultät für Mathematik und Informatik
Ernst-Abbe-Platz 2
07743 Jena
{volkmar.schau, steffen.späthe, wilhelm.rossak}@uni-jena.de

[2]Ernst-Abbe-Fachhochschule Jena
Carl-Zeiss-Promenade 2
07745 Jena
christian.erfurth@fh-jena.de

[3]Universitätsklinikum Jena
Katastrophenschutzbeauftragter des Universitätsklinikums Jena
Bachstrasse 18
07743 Jena
karsten.krohn@med.uni-jena.de

Abstract: Das Projekt SpeedUp[1] hatte die Zielstellung, Einsatzkräfte in Großlagen durch ein mobiles und weitgehend selbstorganisierendes technisches System zu unterstützen. Durch einen interdisziplinären Ansatz konnten in die Konzeption organisatorische Aspekte der beteiligten Behörden und Organisationen einfließen und aus dem Szenario Großlage ergaben sich Rahmenbedingungen und Einschränkungen für die verwendete Technik sowie Anforderungen an die Infrastruktur. Aus den damit verbundenen Herausforderungen lassen sich verschiedene Aspekte ableiten, die für ein unterstützendes IT-System als ortsbasierte Anwendung eine hohe Relevanz haben.

1 Motivation

Unachtsamkeit, technisches Versagen oder eine Naturkatastrophe - innerhalb weniger Minuten verwandelt sich eine Umgebung in ein Katastrophengebiet. Bilder zeigen verheerende Verkehrsunfälle und Brände in unsicher gebauten Tunneln, Entgleisungen von Personenzügen und Terroranschläge. Es sind unerwartete Ereignisse, die Einsatzkräfte vor extreme Herausforderungen stellt. In kürzester Zeit müssen Einsatzteams aus unterschiedlichen Behörden die Situation erfassen, Menschen retten, verletzte Personen identifizieren und für Sicherheit und Ordnung sorgen. Dabei stellt das nur einen Teil ihrer gesamten Aufgaben dar und unterscheidet sich in der Bewältigung stark nach der Organisation- und Arbeitsstruktur der einzelnen Kräfte.

Polizei, Feuerwehr und Rettungsdienst sind verschieden aufgebaut und agieren mit unterschiedlichen Zielen. Dabei stellen gerade der Austausch und die Kooperation eine schnelle Hilfe vor Ort sicher. Eine übergreifende Zusammenarbeit mit einem gemeinsamen Wissensaustausch ist bis heute vollkommen offen. Für die bei einem Großeinsatz beteiligten Einsatzkräfte existierte bisher kein integrative Informations- und Kommunikationssystem unmittelbar am Schadensort. Derzeit erfolgt die Koordination durch persönliche Absprache vor Ort, über Behördenfunk oder auf Grundlage papierbasierter Dokumentationen. Bei großräumigen Szenarien wie Massenunfällen sind persönliche Absprachen oder die Beschaffung und Weitergabe von

[1] SpeedUp wurde im Rahmen des Programms "Forschung für die zivile Sicherheit" der Bundesregierung (Bekanntmachung "Schutz und Rettung von Menschen") durch das Bundesministerium für Bildung und Forschung (BMBF) gefördert (Projektlaufzeit 01.05.2009 - 31.07.2012).

Dokumenten jedoch kaum zu bewältigen. Die Folgen sind Zeitverlust und im schlimmsten Fall eine steigende Zahl von Opfern.

SpeedUp unterstützt die Einsatzkräfte in Großlagen durch ein mobiles und weitgehend selbstorganisierendes technisches System. Hierzu werden Kommunikations- und Datenplattformen zur Koordination und Vernetzung aller Einsatzkräfte vereint und die Lösung so ausgelegt, dass sie mobile Plattformen und lokale Ortungssensoren standardmäßig integriert. Das System muss dabei unterschiedliche Kommunikations- und Organisationsstrukturen - auf Befehl - weitgehend selbstorganisiert errichten und stabil betreiben. SpeedUp ist ein interdisziplinäres Vorhaben, das Forschungsergebnisse, Technologien und Partner aus unterschiedlichen Richtungen verknüpft. Dabei ist der Aufbau eines „shared mental models", die Sicherstellung einer „situational awareness" und der Umgang mit „Kulturunterschieden" entscheidend [KG+10]. Sie dienen der Erarbeitung und Optimierung adäquater Interaktionsmodelle mit dem Ziel einen umfassenden Informationsfluss sowohl intra- als auch interstrukturell zu erreichen. Mit anderen Worten erfolgt eine Unterstützung der Kräfte im Einsatz nur, wenn das System ständig verfügbar ist, einfach zu bedienen und die notwendigen Informationen orts- und rollenbezogen zusammenträgt [WY+11].

2 Lessons Learned – Ortsbasierte Anwendung für Einsatzkräfte

Die Arbeit im Projekt hat schnell gezeigt, dass eine interdisziplinäre Herangehensweise für eine erfolgreiche Technologieeinführung Voraussetzung ist. Antworten auf die Frage, welche Technologie geeignet ist und was die notwendigen Funktionalitäten sind, können erst beantwortet werden, wenn der Einsatzbereich und der organisatorische Kontext geklärt sind. Das gewählte Anwendungsszenario einer Großlage wie der Massenanfall Verletzter (MANV) bedeutet eine zusätzliche Komplexität durch die Kooperation verschiedener Organisationen mit unterschiedlichen Aufgaben an einem mehr oder weniger zufälligen Ort.

2.1 Eine Zusammenstellung der Herausforderungen

Ein technisches System zur Unterstützung in Großlagen muss auf fachlicher Ebene eine geeignete Unterstützung für die verschiedenen Einsatzkräfte ermöglichen sowie auf technischer Ebene einen zuverlässigen Betrieb gewährleisten. In Konsequenz fließen in die Konzeption organisatorische Aspekte der beteiligten Behörden und Organisationen ein und aus dem Szenario Großlage, ggf. auch Einsätze in Gebäuden oder unterirdischen Einrichtungen wie Tunnel, ergeben sich Rahmenbedingungen und Einschränkungen für die verwendete Technik und Anforderungen an die Infrastruktur (siehe Abb.1).

Prozesse, Organisationen, Informationen: Ohne klare und gelebte Prozesse zur Bereithaltung von unterstützenden IKT-Systemen, sind diese Systeme nicht sinnvoll einsetzbar. Diese Organisation kostet Kraft und Zeit, ist aber zwingende Voraussetzung für die Einführung von unterstützenden IT-Systemen. Bisherige Prozesse im Einsatz insbesondere bei Großlagen basieren auf „Papier & Bleistift" – das Wissen zum korrekten Umgang mit existierenden Formularen wird durch kontinuierliche Schulungen und Einsatz von „Gesundem Menschenverstand" realisiert. Die Abbildung des „gesunden Menschenverstands" in Software ist jedoch seit langem Gegenstand der Forschung und noch immer weit weg von praktischer Einsatzbarkeit [KG+10, KW+12]. Bisherige Prozesse werden sich wandeln müssen, neue Prozesse entstehen, alte entfallen. Beispielsweise gab es in Leitstellen der Polizei oder des Rettungsdienstes solche Veränderungen von Prozessen durch den Einzug von IT. Gesammelte Informationen in papierbasierten Verzeichnissen (Krankenhauskapazitäten, Rufnummern, Dienstleister etc.) sind elektronisch und in besserer Qualität verfügbar. Aber die alten Prozesse bleiben im Bestand und werden beübt, um bei einem Ausfall der Technik weiterhin handlungsfähig zu sein. Damit ist eine gewisse Abwärtskompatibilität für einen neuen Prozess wie auch für ein neues System von Bedeutung.

Zu den wichtigsten Informationen am Einsatzort gehören neben den räumlichen Bedingungen, Informationen über vorhandene Einsatzkräfte sowie über technische Möglichkeiten. Prozesse zur (zentralen) Stammdatenerfassung und Verteilung auf mobile Geräte sind daher entscheidend. Der Informationsfluss richtet sich dabei nach den Organisationsformen und Aufgaben (Rollen) der Einsatzkräfte. Vor Ort soll der Kommunikationsfluss unter Zuhilfenahme verfügbarer mobiler Geräte möglichst robust und selbstorganisierend etabliert werden. Je nach den örtlichen Gegebenheiten (verfügbare Kommunikationsinfrastruktur, Weitläufigkeit,

Hindernisse in Kommunikationswegen, etc.) ist das Bestreben, die mobilen Geräte der sich ggf. bewegenden Einsatzkräfte unabhängig von der Organisation durch eine Vernetzung als Hub oder Transportmittel für Informationen zu nutzen [SH+12]. Schwierigkeiten ergeben sich aus den Organisationsunterschieden (unterschiedliche Kommunikationswege) und rechtlichen Fragestellungen zum Datenaustausch zwischen Organisationen (logischer Datenschutz vs. technische Datenverteilung).

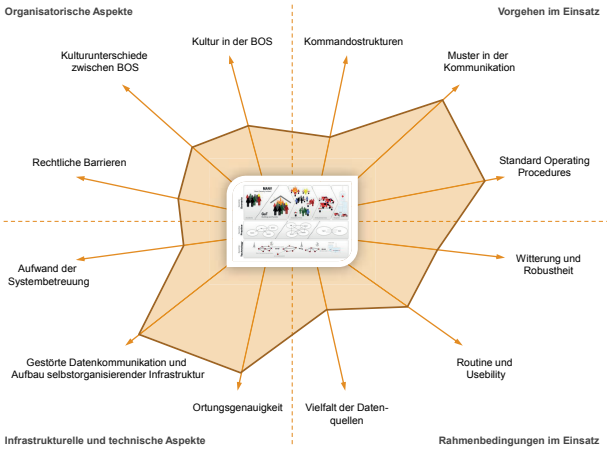


Abbildung 1: Bewertung verschiedener Aspekte zum Projektstart (Bedeutung steigt in Pfeilrichtung)

Die Kulturunterschiede der Organisationen und deren Aufgabenbereiche beeinflussen stark die Funktionalität des IT-Systems und die erforderlichen Daten [WY+11]. Während beim Rettungsdienst die Information über den Status von Verletzten wesentlich ist und diese in einer Einsatzleitung vor Ort zusammenfließen, sind personenbezogene Daten und Informationen über das Geschehen am Einsatzort für eine Polizei, die ggf. einen Führungsstab im Hintergrund hat, von Relevanz. Der Informationsbedarf einer Einsatzkraft ist zusätzlich abhängig von der Rolle und den damit verbundenen Aufgaben. Informationen müssen unter Umständen zusammengefasst werden und unter Hinzunahme externer Informationsquellen schnell zu Entscheidungen (ggf. unterstützt durch automatisiert ermittelte Entscheidungsvorschläge) führen können, z.B. Sichtung und Abtransport von Verletzen.

Für Informationen, ohne explizites Quellsystem, muss ein adäquater Ersatz und damit eine zentrale Informationsquelle geschaffen werden. Neben dem System selbst, müssen auch hier explizite Zuständigkeiten festgelegt werden. Noch schlechter als keine Daten sind falsche Daten. Zur Sicherstellung korrekter und aktueller Daten ist die Anbindung an existierende, externe Systeme soweit wie möglich voranzutreiben – nur dann kann ein unterstützendes IT-System sinnvoll und übergreifend genutzt werden. (z.B. freie Betten in Krankenhäuser). Notwendige Stammdaten (Fahrzeuge, Personen, Einrichtungen etc.) müssen soweit wie möglich automatisiert erfassbar sein und aus führenden Systemen über standardisierte Schnittstellen integrierbar sein. Problem auch hier: Organisation (rechtlich und technisch) der Schnittstellen sowie Sicherstellung des Betriebs.

Modularität und Funktionalität: Die Anpassbarkeit der bereitgestellten Funktionsauswahl für das SpeedUp-System hat sich als extrem Wichtig gezeigt, da trotz gleichen Aufgaben und Zielstellung, jede Organisationseinheit andere gelebte Praxis hat. In der Architektur und dem Design der Softwarelösung wurde durch modulare Bausteine diese Anforderung umgesetzt (siehe Abb. 2). Der gewählte modulare Aufbau ist geeignet, um verschiedene Organisationen sowie unterschiedliche Rollen innerhalb der gleichen Behörde oder Organisation im Einsatz abbilden zu können. Wobei zu beachten ist, dass die Rolle einer einzelnen Einsatzkraft sich dynamisch ändern kann und zeitweise eine Einsatzkraft auch zwei Rollen „teilweise“ innehaben kann.

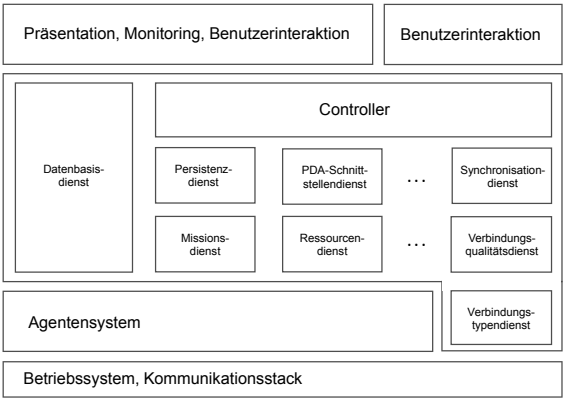


Abbildung 2: Modularer Aufbau des SpeedUp-Demonstrators mit organisationsspezifischen Modulen

IT im Rettungseinsatz: Die Akzeptanz eines IT-Systems muss sich durch eine gute Praxistauglichkeit bewähren, d.h. auf die Aufgaben und Abläufe passende Funktionalität zur Verfügung stellen, einfache und intuitive Bedienung ermöglichen und damit den „Rettungsbetrieb“ nicht stören. Die Akzeptanz hängt dabei nicht nur von der Softwarelösung ab sondern auch entscheidend von der Hardware: Gewicht und Robustheit der Geräte, Betriebsdauer im Batteriebetrieb, Aufwand zur Erhaltung der Betriebsbereitschaft und zum Aktivieren der Geräte sowie ggf. Aufbauaufwand einer notwendigen Infrastruktur. Die Geräte dürfen im Einsatz nicht als störend empfunden werden [HC+12].

Für den zweckmäßigen Gebrauch im Einsatz ist eine Routine im Umgang mit dem System entscheidend. Um dies zu erreichen muss das System im Idealfall als Standardwerkzeug im Regelbetrieb und nicht nur in eher seltenen Großlagen genutzt werden. Unterstützend helfen Simulationen, Schulungen und Übungseinsätze um die Angst an einem neuen technischen System zu nehmen. Die Passgenauigkeit und Tauglichkeit der Softwarelösung, die Usability, spielt eine entscheidende Rolle dabei. Das äußert sich u.a. in einer intuitiven Bedienung, die zugeschnitten auf typische Einsatzszenarien und die verwendete Hardware ist [HC+12].

Technische Herausforderung: Im Verlauf der technologischen Betrachtung und der Implementierungen hat sich bei den Kommunikationsprotokollen, den Routing- und Ortungstechnologien deutlich gezeigt, dass die sinnvoll nutzbaren Ergebnisse unbefriedigend sind. Unter Kommunikationsaspekten ergibt sich eine Notwendigkeit nach einer austragbaren und leicht transportablen Kopplungslösung [SE+11]. Die am Markt verfügbaren Systeme weisen einen integralen Fahrzeugansatz auf. Dessen Anwendbarkeit beschränkt sich auf einen befahrbaren Zugang. Eine tragfähige Kleinlösung in Form eines Kleinkoffers hat sich als sinnvoll herausgestellt. Dabei ist die Sicherstellung der Konnektivität mit marktverfügbaren Systemen extrem problematisch, da kein Betriebssystem für einen Ad-hoc Betrieb ausgelegt ist. Bezüglich der Ortungstechniken haben sich die Anforderungen auf der einen Seite in Form der Genauigkeit entspannt und auf der anderen Seite der Wunsch nach einer flächenübergreifenden Lösung (Nutzbarkeit in Gebäuden und Feld) verstärkt. Im Ergebnis haben sich die positiven Aspekte einer Ortung in der Fläche sowohl in der Führung als auch in der Visualisierung von Einheiten deutlich gezeigt. Bei allen technischen Lösungen steht die einfache Anwendbarkeit im Vordergrund und ist eine Praxistauglichkeit notwendig. Aktuell ist die Erreichbarkeit dieser Güteklasse im Realbetrieb nur aufwändig realisierbar.

2.2 Aspekte für die Konzeption ortsbasierter Anwendungen für Einsatzkräfte

Aus den Herausforderungen lassen sich einige Aspekte (siehe Abb.3) ableiten, die für ein unterstützendes IT-System eine hohe Relevanz haben. Dabei spielt die Spezifika des Einsatzgebietes sicher eine bedeutende Rolle. Dennoch lassen sich die Aussagen auch verallgemeinern und sind übertragbar auf ortsbasierte Anwendungen.

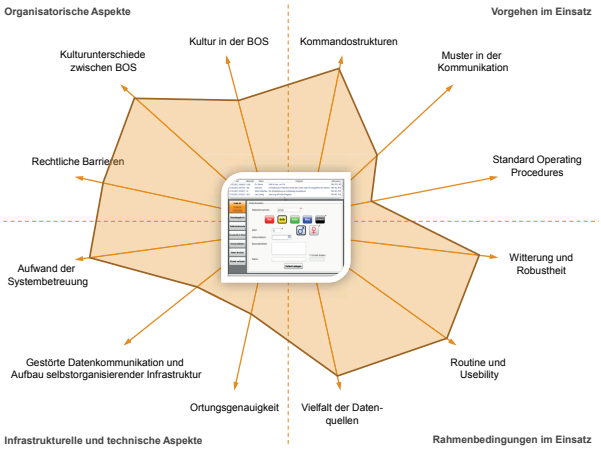


Abbildung 3: Bewertung verschiedener Aspekte zum Projektstart (Bedeutung steigt in Pfeilrichtung)

Bei der Bewertung dieser Aspekte kommt man zu unterschiedlichen qualitativen Aussagen. Insbesondere verändert sich die Sichtweise mit zunehmendem Kenntnisstand über die organisatorischen Aspekte, den fachlichen Anforderungen sowie den eher technischen Aspekten.

3 Zusammenfassung

Im Laufe des Projektes SpeedUp hat sich klar die Sichtweise von einer auf die Funktionalität und Technik einer ortsbasierten Anwendung zu einer Sichtweise entwickelt, die den organisatorischen Kontext und die angestrebte Akzeptanz stärker einbezieht (vergl. Abb.1 & 3). Diese Veränderung ist schwer messbar, dennoch an den Herausforderungen und den „Lessons Learned“ erkennbar. Sie ist ebenso wichtig, um die Erfolgschancen eines IT-Systems und damit eines IT-Projektes zu erhöhen.

Literaturverzeichnis

[HC+12] Hal, G., Coskun, T., Artinger, E., Benzina, A., Klinker, G.: User-centered comparison between classical and edge interaction on a heavy rugged tablet PC used in MCIs. Workshop zur IT-Unterstützung im Emergency Management & Response. Im Rahmen der Informatik 2012, Braunschweig (Germany), 2012.

[KG+10] Krüger, U., Gabdulkhakova, A., Schau, V., Beckstein, C.: Information and Management Support for Mass Casualty Incident Scenarios. FSU-Jena, Technical Report Nr. Math/Inf/04/10, 2. Workshop IT-Unterstützung für Rettungskräfte. Im Rahmen der Informatik 2010, Leipzig (Germany), 2010.

[KW+12] Krüger, U., Wucholt, F. and Beckstein, C.: Electronic Checklist Support for Disaster Response. 9th ISCRAM, Human Experiences in the Design of Crisis Response and Management Services and Systems, Vancouver (Canada), 2012.

[SE+11] Schau, V., Erfurth, C., Eichler, G., Späthe, S., Rossak, W.: Geolocated Communication Support in Rescue Management. Proceedings 8th International Conference on Information Systems for Crisis Response and Management (ISCRAM), Lissabon (Portugal), 2011.

[SH+12] Schau, V., Hellfritzsch, S., Scharf, S., Eichler, G., Erfurth, C., Rossak, W.: Simulation of wireless, self-organizing and agent-based dynamic communication scenarios. Proceedings 9th International Conference on Information Systems for Crisis Response and Management (ISCRAM), Vancouver (Canada), 2012.

[WY+11] Wucholt, F., Yildirim-Krannig, Y., Mähler, M., Krüger, U., Beckstein, C.: Cultural Analysis and Formal Standardised Language - a Mass Casualty Incident Perspectiv. 8th ISCRAM, User Centred Design Process for Emergency Management Information Systems, Lisabon (Portugal), 2011.

Towards Location-based Services in a Cloud Computing Ecosystem

Sebastian Zickau, Axel Küpper

Service-centric Networking
Technische Universität Berlin
TEL19 - Ernst-Reuter-Platz 7
10587 Berlin
{sebastian.zickau, axel.kuepper}@tu-berlin.de

Abstract: With the introduction of inexpensive mobile devices, which are capable of using location-based services (LBS), the popularity of such applications has grown rapidly. A lot of these applications are available to the public and used by them on a daily basis. In addition to this trend cloud computing was also rapidly adopted by professionals and private consumers alike. The project TRESOR[1] evaluates and develops cloud computing solutions for the health sector domain. In this context location-based services are being researched. The paper presents different scenarios, which were all deducted from the current project work. The different areas such as location-based access control (LBAC), geofencing, mobile devices, geotagging, etc. are in the scope of the work in progress.

1 Introduction

Two of the fastest growing information technologies are location-based services and cloud computing. These areas are also very well known to people who do not have a broad understanding of communication technologies and protocols. One reason for this progression is the worldwide success of smartphones in recent years. People deal with cloud services and location applications on a regular basis. The goal of the TRESOR project is to investigate how cloud computing, with all its benefits and challenges, can be introduced to the health sector domain. A cloud ecosystem will be developed to address specific issues that arise from this data sensitive domain.

New technologies bare new security issues especially in network dependent areas particularly if combined with new paradigms such as cloud computing. Access to services in the cloud mostly rely on username/password authentication. Stronger authentication mechanisms are applied but mainly focus on similar principles like a known or owned token. As security is often characterized as a process, the awareness to include different technologies during the authentication and authorization process has been increased. Often the access to resources is limited to a specific user group or to an environment, which can be demarcated by borders, both virtual and non-virtual.

Apart from being a security feature the location services of a cloud ecosystem can also be used by services, which can compute and react to positioning information. Providing an employee with needed information who is at a customer's site is an example for such a location-aware cloud service. The service could limit the access to secure data by considering location information as well as time constraints.

Another cloud field where location information will become relevant is the compliance of federal laws within the context of international cloud providers. This paper discusses and describes the fields of application of location services within a cloud ecosystem and shows the added values. The next chapter gives a short overview of the different TRESOR components. In Chapter 3 the ideas around location-based services in a cloud ecosystem are evolved. The paper concludes with a brief summary and an outlook of future work.

[1] **TR**usted **E**cosystem for **S**tandardized and **O**pen cloud-based **R**esources, http://www.cloud-tresor.com/

2 The TRESOR Ecosystem

In the project TRESOR an ecosystem of different cloud components is being developed [SZT+12]. The cloud ecosystem consists of the following parts:

- A Platform as a Service (PaaS) component, which is based on the standardized Open Services Gateway initiative (OSGi)-framework[2] to overcome lock-in effects through a non-proprietary architecture with common APIs.
- A generic marketplace that is focused on data sensitive cloud applications.
- A generic broker, which is used to negotiate cloud service conditions between consumers and providers. It is also used at runtime holding two repositories that store information about services and consumer constraints respectively.
- A generic distributed proxy, which is designed to be run by a trusted third party provider and logs and monitors the service consumption.

The proxy component in conjunction with the broker repositories are determined to be the active components that encapsulate the supplementary location-based services because it will also be the place to enforce such compliance.

The project will evaluate its ecosystem in two scenarios. The first scenario deals with continuous medical data records without any media format changes between different health institutions. The second scenario will provide a cloud service, which checks a patient's different medication together with any preconditions resulting in possible medication interaction warnings that are given to the doctors. An overview of the TRESOR components is given in Figure 1. On the left hand side the TRESOR PaaS-platform is outlined, the broker and proxy are shown in the middle and the right hand side drafts the different consumer roles.

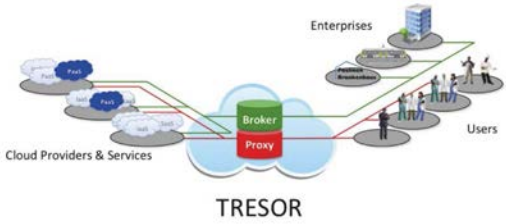


Figure 1: The TRESOR Cloud Ecosystem

3 Location-based Services in a Cloud Computing Environment

A solution needs to be developed, which addresses the different requirements of the given location-based examples. A component needs to be integrated into the cloud proxy functionality of the proposed cloud ecosystem. The specific cloud-enabled location management component is then used to gather and exchange the location information with related services and applications. The common technologies used to determine the location are: GPS, Cell-ID, WLAN and IP-based positioning [AB09]. Additional to that combinations of existing technologies are under investigation, for example using mobile device capabilities to enable location-based applications on stationary devices, see Subsection 3.3.

[2] http://www.osgi.org/Main/HomePage

The following subsections deal with different aspects of location-based services in the presented cloud ecosystem, such as geotagging, geofencing, and the usage of mobile device capabilities.

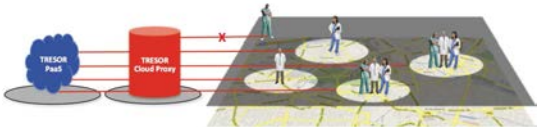


Figure 2: Location-based Services in TRESOR

3.1 Geotagging

In recent years the equipment used by medical staff is commonly producing digital data of a patient. Usually patients are in contact with a variety of different medical institutions and resulting form that also in contact with a lot of medical staff, such as doctors or nurses. The trend to put medical data into the cloud is based on current developments already noticeable. In recent years the data, which was collected in a hospital was archived within its boundaries. With the removal of these preconditions it is a logical step to (geo)tag the medical data of a patient, additional to ID information, such as name, address, date of birth, etc. This information will also support the hospitals' business processes that are related to accounting and billing of the provided treatment. It is important for hospitals to differentiate which medical services were offered to which patient in order to compile a correct report which is then evaluated by the paying health insurance companies.

3.2 Geofencing and SLAs

Due to the nature of cloud services, they can be consumed from various locations of the world. In a scenario where different price models are applied to services depending on the country in which they are consumed these borders need to be represented inside a location service. The information needs to be monitored and evaluated in the component which controls the access to a service.

In the process of accessing cloud services that are tied to a service level agreement, the company needs to make sure that the access to these services is limited to people working on the company's site. Multi-layered security approaches feature different technologies. One can assume that the company's premises and access to their computers is limited. An additional security feature is the information of the user's location. This is being researched in the field of location-based access control [ACdVS09]. Information about the location is also evaluated by the service along with other authentication and authorization credentials, exemplified in Figure 2. It shows that different located users have individual accesses to services. Security features along with other centralized services will be provided by the cloud proxy. In the context of health applications there are legal constraints that certain online services, with additional human interaction, need to be consumed within a maximum range of the hospital premises. The reasons for these regulations lie in corporative interests and are due to pricing considerations. A service for the evaluation of X-ray scans by (tele)-radiologists is one example. These services need to be within a certain radius, e.g. 100 km, of the medical institution, represented in Figure 3. The hospital is only allowed to use the health service in its vicinity [Bun11]. In the future cloud computing is used to provide additional services. Evaluating these positioning constraints can be monitored and restricted through the cloud ecosystem.

Figure 3: Location Constraints while consuming Cloud Services

3.3 Mobile Device Assisted Positioning

To enable location-based services on stationary desktop computers the usage of mobile device capabilities can be used. Mobile devices, e.g. smartphones, have a wide range of positioning capabilities, such as GPS, Cell-ID and WLAN positioning. To enable services, such as location-based access control, these capabilities can be coupled to the desktop computer or other stationary computational devices. For coupling these devices technologies, such as Bluetooth or NFC are under investigation. This can be useful in scenarios where static medical computers lack positioning capabilities and the shifting medical staff on a hospital premises are identified and located with their mobile devices. These devices can connect to stationary workstations in a background process in order to provide location information to applications.

4 Conclusion and Outlook

The work and ideas presented here are the result of investigating how to integrate location-based services in a cloud computing ecosystem to enrich it with additional (security) features. In our increasingly mobile and location independent working environments new concepts and paradigms demand envisioning new solutions for current unconventional situations and scenarios. The location-based functionality is part of the TRESOR infrastructure and integrated into it, i.e. the cloud proxy, the cloud services and the client devices. The external input, such as medical regulations, which are collected as part of the ongoing project work, already show us new areas where location information is needed in future technologies and domains. Location-based services will be evaluated and developed in the two health care scenarios mentioned above. Further location applications, related to cloud computing paradigms, will be researched in the ongoing work.

Acknowledgement. The work presented in this paper was performed in the context of the TRESOR project. TRESOR is funded by the German Federal Ministry of Economics and Technology (BMWi).

References

[AB09] Chao Li Allan Brimicombe. *Location-Based Services and Geo-Information Engineering*. Wiley-Blackwell, 2009.

[ACdVS09] Claudio A. Ardagna, Marco Cremonini, Sabrina De Capitani di Vimercati, and Pierangela Samarati. Access Control in Location-Based Services. *Privacy in Location-Based Applications*, Privacy in Location-Based Applications, LNCS 5599:106–126, 2009.

[Bun11] Bundesgesetzblatt. Verordnung über den Schutz vor Schäden durch Röntgenstrahlen (Röntgenverordnung - RöV). http://www.gesetze-im-internet.de/r_v_1987/index.html, 2011.

[SZT+12] M. Slawik, S. Zickau, D. Thatmann, J. Repschläger, T. Ermakova, A. Küpper, and R. Zarnekow. Innovative Architektur für sicheres Cloud Computing: Beispiel eines Cloud-Ecosystems im Gesundheitswesen. *Proceedings of the 42th Annual Conference of the Gesellschaft fr Informatik e.V. (INFORMATIK 2012)*, 2012.