6

AN INDOOR LOCATION PROVIDER BASED ON PARTICIPATORY SENSING

This chapter describes the design of *WBroximity* as a possible positioning support component for the iVu.KOM framework, as shown in Figure 30). WBroximity is a positioning service based on WLAN- and Bluetooth- proximity sensing. Therefore, WBroximity provides a functional positioning solution in areas where GPS performance can degrade, like in indoor environments. When combined with iVu.KOM, WBroximity can provide support for indoor viewer-centric applications (e.g. augmented reality indoor navigation). The chapter explains in detail the concept of leveraging participatory sensing to collaboratively build a database of WLAN and Bluetooth fingerprints and utilize them for positioning tasks.

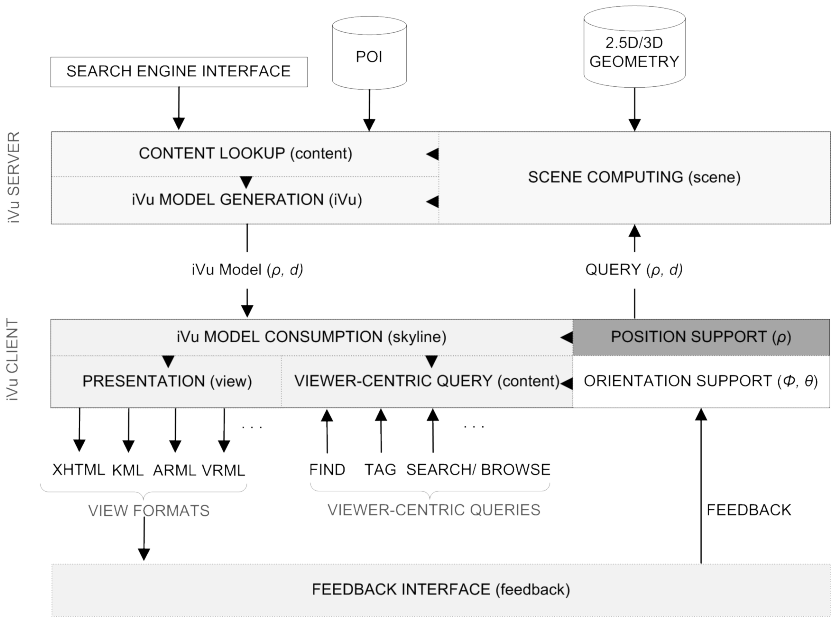


Figure 30: Components covered by this chapter

6.1 MOTIVATION

6.1.1 *Using WLAN for Positioning*

Commonly, GPS performs poorly in indoor settings due to signal blockage and multipath effects. This in turn disables GPS-equipped mobile devices from benefiting from GPS in a range of indoor location-based services (e.g., navigation and object

tracking). One promising technique for indoor positioning is to utilize the Wireless Local Area Network (WLAN) signals, because they show location-dependent characteristics. A WLAN-based technique has also economic advantages. WLAN installations are ubiquitous in indoor scenarios (office environments, as well as apartment blocks). Hence, no additional installation costs are needed. Additionally, the number of mobile devices with WLAN interface is ever increasing, allowing such devices to seamlessly benefit from a WLAN-based positioning system.

For non-LOS environments, fingerprinting proves most relevant

Performance evaluations of WLAN positioning [45] showed that a *location fingerprinting* scheme is most relevant for indoor environments because such scheme does not require a line-of-sight between the transmitter and the receiver as normally required by other schemes like Time of Arrival (TOA) [21]. Commonly, location fingerprinting models the positioning problem as a data classification problem. During the offline phase a radio map is built in the target environment by collecting the Received Signal Strength (RSS) measurements of the nearby access points at specific sampling locations. Location models are then built given the radio map using a supervised machine learning algorithm, e.g., a Decision Tree or Naïve Bayes [35]. In the online phase, RSS measurements are used to calculate the estimated location coordinates in real-time based on the models built in the offline phase.

6.1.2 *Exploiting Participatory Sensing*

Participatory sensing: users collect the fingerprints

Unsurprisingly, collecting fingerprints turns out to be the most costly phase, especially when a wide system coverage is required while maintaining acceptable levels of accuracy. For example, the commercial positioning service Skyhook [13] employs many drivers to scan WLAN hotspots by driving through streets in cities and towns. Scanning should be even done on a periodic basis to cater for the fact that the WLAN infrastructure can change over time, e.g., when some access points are shut down, relocated or replaced. Besides the incurred high costs, this approach has limitations when accuracy of indoor positioning is concerned. Firstly, as scanning is done from vehicles in the streets, real scanning of fingerprints inside buildings remains unapproachable, which in general affects the accuracy obtained indoors. Secondly, because WLAN signals are known to travel relatively long distances (up to 40m indoors [64]), precise localization requires gathering a large number of fingerprints and probably high-dimensional fingerprints (i.e. many networks per each fingerprint) which may not be affordable at all places.

6.1.3 *The Combination*

WBroximity combines WLAN, Bluetooth and participatory sensing

WBroximity (short for **W**LAN and **B**luetooth P**roximity** sensing) provides a novel solution to address the aforementioned concerns as follows:

- WBroximity combines the good of WLAN and Bluetooth technologies for location fingerprinting. Similar to WLAN, Bluetooth is already integrated in modern mobile devices, and moreover in many consumer gadgets like game consoles, printers, and speakers [14]. However, in contrast to WLAN, Bluetooth provides a short-range wireless coverage (up to 10m for Bluetooth Class 2 [14]), which makes it relevant for fingerprinting locations indoors (up to the room level).
- WBroximity uses *participatory sensing* to collect location fingerprints. In participatory sensing [20], a user of a mobile device can be viewed as a mobile sensor node that gathers data and shares it among all users. For the specific purpose of positioning, users participate with the gathered fingerprints plus user-generated *fingerprint labels*. A fingerprint label is a small piece of information indicating where the user was when a fingerprint was collected.

The benefit of using a participatory sensing paradigm here is threefold:

1. Cutting the costs. With participatory sensing it is possible to eliminate the extra costs that are otherwise needed to employ drivers to scan fingerprints. Normal users will do this job on a voluntary basis.
2. Increasing the coverage. Because of the always-with always-on nature of the mobile devices, fingerprints scanning can reach wherever the users go.
3. Increasing the accuracy. The fact that the same location can be sampled by many users independently provides useful redundancy and a way to control the quality of data gathered by each user. Moreover, with participatory sensing, it is also possible to scan fingerprints directly inside buildings, as contrary to scanning them in the street only.

6.2 SYSTEM DESCRIPTION

As depicted in Figure 31, the design of WBroximity follows a client-server architecture:

- the client runs as a background service on the user's mobile device and offers functions for scanning the WLAN and Bluetooth fingerprints, getting the fingerprints assigned with the user-provided labels and communicating the fingerprints with the server. Labels themselves are provided using a feedback frontend. If available, the GPS can be used to provide GPS data as labels (explained next in Section 6.2.1). Through a dedicated interface, LBS can access WBroximity and obtain location information. By this means, iVu.KOM can also benefit from WBroximity as a location provider.
- the server maintains a global location model, which contains all labeled fingerprints that are collected from different participants. A labeled fingerprint is used by the *model adaptation function* to re-build the model. Unlabeled fingerprints are used by the *classification function* to get a location fix.

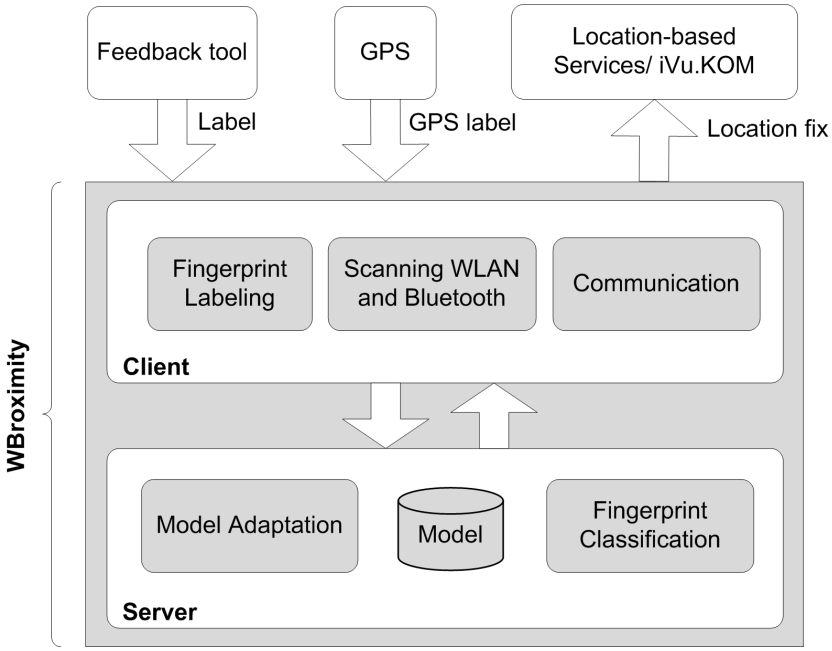


Figure 31: Client and server components of the WBroximity service

Appendix A describes the implementation of WBroximity as a location provider for the Android platform.

6.2.1 *Collecting and labeling the Fingerprints*

Collecting and labeling the fingerprints form the part of the system where users contribute to improve the overall positioning accuracy. Most WLAN and Bluetooth network cards can read at least the following information about networks in range:

- the *Basic Service Set Identifier (BSSID)*, representing the name of the WLAN/Bluetooth network.

- the *Media Access Control (MAC)* address, representing the unique physical address of the network card for a WLAN or Bluetooth device.
- the Received Signal Strength (RSS), indicating the strength of the received signal.

Therefore, a location fingerprint comprises these three parameters for all WLAN and Bluetooth networks detected at that location.

The current design of WBroximity allows fingerprints to be labeled with a symbolic location, i.e. a string representing the user's location (e.g., office, kitchen, administration building, etc.). As shown in Figure 32a, a symbolic label is created using a frontend by manually entering the name of the location the user thinks she is at. However, to display on a map a location determined by WBroximity, GPS data (longitude and latitude) are to be tagged with the symbolic label. This can be achieved by one of the following ways:

WBroximity symbolic labeling

- the symbolic label is tagged with the last position fix that is read from the on-board GPS receiver. This method assumes that the participating user is using a GPS-capable mobile device.
- for participants without a GPS-capable device (or when GPS signal is not available), a map view is offered on which the participant can manually pinpoint her location. The map view can be used outdoors as shown in Figure 32b and indoors as shown in Figure 32c. The map view (at different zoom levels) is geo-referenced, meaning that the relative coordinates of points inside the view can be easily mapped to absolute GPS coordinates.

6.2.2 *Building the Model and Classification*

On the server, labeled fingerprints are used as input to a classification algorithm to build the location model. As symbolic labels are being used here, it is needed to apply classifiers that support discrete class attributes. In the current design, Naïve Bayes [35] was chosen as a classifier as it has strong independence assumptions, which matches quite well the characteristics of the underlying fingerprints. In particular, Naïve Bayes assumes that the presence of a network is unrelated to the presence of any other network.

(a) (b)

(c) (d)

Figure 32: (a) Symbolic labeling (b) Map view for outdoor labeling (c) Map view for indoor labeling (d) Returning most probable locations

Commonly, Naïve Bayes classification works by picking the *hypothesis* (or class) that is most probable (known as the maximum a posteriori rule). The rule can be formulated as follows:

$$\text{classify}(f_1, \cdots, f_n) = \arg\max_C p(C = c) \prod_{i=1}^{n} p(F_i = f_i | C = c) \tag{6.1}$$

where $f_1, \cdots, f_n$ are the features included in the instance to be classified (i.e. the fingerprint, in our case), and C is the instance class (i.e. the location label in our case). Another advantage of Naïve Bayes is that it outputs not only the predicted class, but also a probability distribution over all classes. Using this feature, WBroximity returns not only the location label with the highest classification accuracy, but also the probability distribution of other labels (Figure 32d), which makes the users more confident and aware of the service performance.

6.3 WBROXIMITY LOCATION FIX: AN EXAMPLE

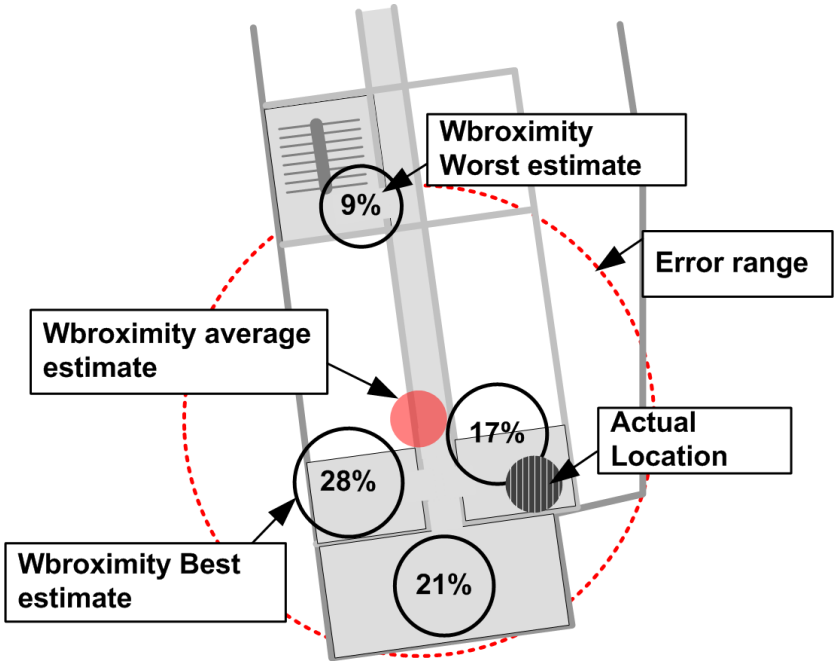
Translating a classified symbolic label into a GPS position

This example shows how WBroximity transforms symbolic labels into GPS geographic locations and how it computes the location accuracy. As mentioned in Section 6.2.1, whenever available, GPS coordinates are appended to the symbolic labels. In the test case shown in Figure 33a, the author has pinpointed his actual location (marked by the dark spot) on an accurate indoor map of the KOM lab building[1]. The map is also accurately geo-referenced so that GPS coordinates can be assigned to the locations inside the building. At this test location, 10 WLAN hotspots were detected, 3 of them are known to be operated within the test building. With 220 fingerprints used to build the location model, the 4 empty circles (with solid perimeter) in the figure represent the locations of the 4 most likely labels $\{l_1, l_2, l_3, l_4\}$ and the probability of each location as reported by the Naïve Bayes classifier[2]. Therefore, the worst location estimate has simply the GPS coordinates of the smallest empty circle (i.e., with 9%), the best location estimate has the coordinates of the largest empty circle (i.e., with 28%), and the coordinates of the average estimate (marked by the red spot) is computed according the following formula:

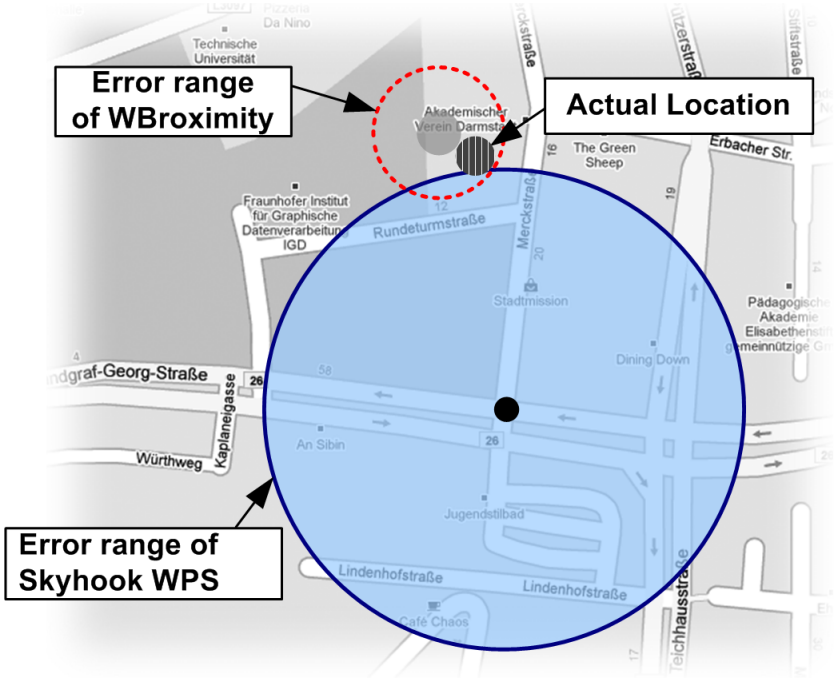
$$x_{avg} = \sum_{i=1}^{4} (p_i * x_i) \tag{6.2}$$

1 At the TU Darmstadt

2 In Figure 33a, the size of the empty circle (with solid perimeter) is proportional to the obtained probability.



(a) WBroximity worst, average and best position fixes



(b) Performance of WBroximity vs. performance of Skyhook

Figure 33: Computing WBroximity location fix from symbolic labels

where p_i is the probability for location l_i, and x_i is the coordinate (longitude or latitude) of location l_i. It is to be noticed here that the probabilities do not necessarily sum to 1.0, because only the 4 most likely labels are returned. It is assumed that labels with very low probability do not have big influence on the overall average estimate. The overall accuracy is then represented by the minimum circle enclosing the 4 most likely locations, indicated by the dotted line (with radius $\cong 17m$). This accuracy is acceptable given the relatively low number of reference fingerprints used to cover the whole area of the test location. However, even with this low number, the possibility to scan fingerprints indoors will commonly give higher accuracy compared with approaches that depend on only in-street fingerprint scanning. For example, Figure 33b depicts the accuracy achieved by Skyhook WPS [13] at the same test location. During this test, the Skyhook SDK available for Android was used on the same mobile device, this way it was assured that the same WLAN network card was used to collect the same fingerprints that were collected in the case of the WBroximity test. However, it is unclear from the SDK documentation which fingerprint attributes (i.e., RSS, Link Quality Indicator (LQI), etc.) are being included in the fingerprints. In this test, Skyhook reported a location with an accuracy of about 147m, which, although much better than the nominal $\pm 750m$ accuracy officially claimed, hardly covers the actual location. This observation suggests that it can be beneficial to use the WBroximity approach to extend the reach and accuracy of existing services that otherwise depend solely on in-street network scanning.

Chapter 8 provides an evaluation of WBroximity with focus on achieved accuracy in light of availability of WLAN and Bluetooth networks and aspects of participatory sensing like number of participating users and quality of participation.

6.4 COMPARISON TO RELATED WORK

Participatory sensing is a field that has been recently receiving serious attention. For example, NoiseTube [47] is a participative approach to measure noise pollution by turning mobile phones into noise sensors and automatically sharing the geo-localized measurements with the community. In the area of location-based services, CellSpotting [8] counts on users' participation of detected GSM and UMTS cells, their geographic locations, and nearby touristic information. A very closely related work is the Jiwire Wi-Fi Finder [15], offering a mobile application with a feature that allows users to submit newly discovered hotspots. However, in contrast to the approach presented in this thesis, the aim of [15] is WLAN-based advertising and finding free and fee-based hotspots, therefore, accurate positioning is not a goal

here. In [23], the authors even analyze the applicability of participatory sensing for mission-critical scenarios including emergency scenarios.

In general, since participatory sensing is heavily based on voluntary contribution by users, designing incentives for participation remains a hot research topic. For example, some work proposed to take advantage of gaming to obtain useful labeled images [65]. In [43], economic models are studied and an auction-based mechanism for commercial participatory sensing is proposed. A *give-and-take* scheme where users had a balance between requested and answered queries is proposed by [32], while [24] proposed an application where users can get information about cheapest prices for one product as long as they submit one themselves. In [49] the authors suggest means to motivate the users participation with information while the users are moving. The goal is to collaboratively build different generic mobility models.

In the area of localization, fingerprinting is not the only technique but there are several other works, especially on triangulation/trilateration. GPS systems cover already outdoors positioning within a few meters error rate [28], but different localization methods are needed where GPS signal is not available, for example indoors or in dense high rise urban environments (called "urban canyons"). Alternative methods, as ultra sound [52] or infrared [66], require the deployment of specific devices, while others can rely on existing infrastructures, as GSM, WLAN and Bluetooth. Assisted GPS (A-GPS) [38], being increasingly implemented in modern mobile devices, aims at shortening the time needed by a GPS receiver to lock on GPS signal. This is useful in indoor environments where the GPS signal is often disturbed by blockage and multipath effects. The drawbacks of A-GPS is that it does not work in real-time as communication with the network assistance server is needed. Besides, this communication is required every time the GPS receiver moves out of the service area. Therefore, it is only useful for locating a particular place in a small area.

In [42], GSM-based trilateration techniques alone in a city environment yielded an accuracy of $100-200\text{m}$, improved to $15-20\text{m}$ in conjunction with WLAN beacons. Such work relied on a database filled with the absolute positions of beacons provided by institutions or war-drivers (like the already mentioned Skyhook). Although this is a feasible approach, there is no guarantee or incentive to provide such data. In [41] similar results using fingerprinting were achieved without the help of WLAN beacons, but the GSM stations had to be known in advance, and their RSS and locations collected and mapped. Another fingerprinting technique used the 6-strongest GSM cells achieving 44m accuracy at best [40], while [63] reached around 5m by using the fingerprints from all the detected GSM cells in range. Unfortunately,

they achieved this result with up to 29 additional cells, hardly available everywhere, thus deteriorating such accuracy.

Bluetooth fingerprinting has been used in [37] for indoor localization, but although good room level results have been achieved, such system alone can provide location information only for a limited area and requires an extensive deployment of fixed Bluetooth devices due to their limited range. WLAN solutions are proved to reach up to meter level accuracy, especially through fingerprinting [39][44], but normally these approaches require initial (and periodical) calibration or several input information from expert users.

7 PROOF OF CONCEPT

This chapter discusses the relevant details of the prototypical implementation of iVu.KOM. In particular, the realization of the client components are described.

7.1 OVERVIEW: THE IVU.KOM CLIENT

In iVu.KOM, the client has three basic tasks to fulfill: handling the input from orientation sensor and GPS, maintaining a viewer-centric representation of the scene and communicating with the iVu.KOM server in case the model needs to be updated.

At the current stage, the modules of the iVu.KOM server were not implemented. Therefore, the client works by loading a static XML file. The difference between loading a file from the SD card of the mobile device and from the server is minimal. Whenever a file is received it has to be parsed and converted to Java Objects. The implemented parser is based on the XmlPull V1 API[1] which is part of Android. Since the Java Object structure that represents the scene was chosen to be very close to the XML syntax, the parser basically generates Java Objects from each XML node.

The handling of orientation sensor and GPS input is an important task of the client. The module that handles this task is also the place for methods that improve the signal quality. The module is connected to both sensor and GPS API of Android. For both orientation sensor and GPS locations the frequency of updates can be chosen. New sensor data is delivered every 80 millisecond while GPS data is set to be delivered once a second when a minimum change in location of more than 50 cm occurred. The orientation Sensor is split into two different parts in Android's API. A magnetometer and an an accelerometer. The raw data for both is run through a low pass filter to get rid of noise. After that the orientation is calculated from the data of both sensors. The location reported by the GPS receiver on the other hand is not altered. It is directly sent to the scene model module that updates the billboards and the skyline.

The third main part of the client is the module that handles the scene. Based on the known bounding boxes and the latest location the billboards are updated. Every billboard is first updated with current angles. After this step, the list of all billboards is sorted by distance. The last step following an update in location is the

1 see http://www.xmlpull.org/ or http://developer.android.com/reference/org/xmlpull/v1/package-summary.html

visibility calculation. Billboards that are invisible have their state changed to invisible.

Loading the XML, receiving location/orientation updates and maintaining the skyline are the basic tasks that the client performs during runtime. Since there is an application connected to the client framework there are more tasks than these. The framework can perform LoS queries and the location can be manually set by the application.

7.2 DESIGN DECISIONS

7.2.1 *Background Work Thread vs. Process*

The client is designed to run self-contained in the background. Separating the work of the iVu.KOM client from the foreground is important for Android applications because the default thread, in which every code gets executed, is the display thread. The task of this thread is to draw the GUI and react to user input and it must not be blocked nor loaded by other tasks. If Android detects the display-thread to not react in time it will present the user the well-known "Application Not Responding´´ dialog that suggests to close the activity.

Android supports several ways to implement background work. It breaks down to two basic approaches. The first and simplest way is to use a different thread. The standard Java threading mechanisms as well as some Android specific variants are supported. The other way is more specific to Android. It is possible to create a so-called Service, which is an activity without a GUI, running in it's own Linux process. A new virtual machine instance is started in the process that executes the Service. This mechanism allows system-wide services. However, since access to Objects over the boundaries of the virtual machine is impossible, one has to use the Remote Procedure Call mechanism to serialize and deserialize the objects across the boundaries

A need for the client to be system-wide and unique is not clearly given. Thus the implementation uses a background thread (inside the context of the application) rather than an own process. That way the client API has to be imported into every application that uses the framework. If multiple applications run in parallel, the iVu.KOM framework would also exist multiple times. But because of Android's behavior to pause every application that is send to the background the framework will only run actively once.

7.2.2 *Efficiency of Java: Object reuse*

The Android Developer Guide[2] states that object creation is one of the most expensive operations in Android's Java. Besides memory consumption it also causes the garbage collector to be interrupted which causes hick-ups in the display-thread.

Having this in mind, the iVu.KOM client was designed to reuse objects as much as possible. E.g. bounding boxes do not change over time, thus they form no problem here. However, billboards do change over time. However, by overwriting their values they do not need to be recreated.

If an application has a reference to a billboard for example, then this reference will be valid all the time. The application does not need to get a new one and may use the data provided at any time as it will always be current data. Unfortunately, if the application and the framework run in different threads, then a concurrency problem arises: writing and reading the Object can occur at the same time. To circumvent that, the framework works with Java's `synchronized` mechanism. For the synchronization to work the application has to use synchronized access as well.

7.2.3 *Using the Framework*

For any application using the framework, a first important step is to get a reference to the running instance. Since references to Objects are not shared over Android Activity boundaries, the framework is implemented as a singleton. This allows to keep a single consistent state when multiple Activities are involved.

In a next step, the framework has to be notified if the application is in paused state or running. Since every activity gets notified via callbacks about a state change, that callback should be used to notify the framework. When the application is sent into paused state, the client disables sensor and GPS updates, which saves battery life and processing power.

After getting an instance and telling the framework that it is not paused, the application can access the required data. The data is updated during runtime by the framework whenever the location gets updated. Since Objects are reused by the framework, the application can usually keep references to these Objects but has to read the contained data again whenever it uses the data.

To support time-consuming operations of the framework, a notification mechanism was implemented. For example, calling the LoS query method does not yield an immediate result, but will send a notification to the application when the request is processed and the results can be accessed. This mechanism is based on Android's Handler API which allows to notify a handler

2 http://developer.android.com/guide/practices/design/performance.html

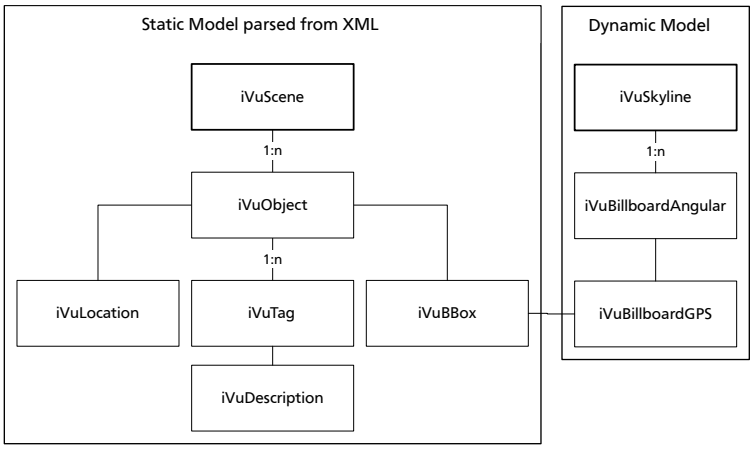


Figure 34: Sample iVu model

with a message. This mechanism is targeted to be used with different threads. The message sent from the sending thread arrives within the context of the receiving thread, thus sending and processing messages is kept in the respective thread.

7.3 INTERNAL DATA STRUCTURE

The data structure of the XML-model (see Figure 2) is directly translated into Java objects, as shown in Figure 34, where the tree structure of the XML is maintained. The root for the object-based iVu model is a single `IVuScene` object which is basically a list container. Attached to the scene are all `IVuObjects` as root for every data associated with a single object. The most important child of an `IVuObject` is the `IVuBBox` which hold the geometric information about the object (unless the object is just a POI). Other child nodes hold the location (the anchor point of the bounding box) and associated data (Tags or POIs).

Listing 2: Format of the iVu model

```
<ivu ts="2010-07-28-11-37-42" lat="49.874671" lng="8.660655"
    alt="140"/>
 <obj name="Main Building" id="obj-1">
  <obb lat="49.874681" lng="8.660666" alt="145" hdg="81.08" w
      ="59.21" h="15" d="21.7" />
        <poi id="poi-1" href="www.ivu.main-building.de" desc=
            "iVu Workshop" />
 </obj>

 <poi id="poi-2" href="www.tu-darmstadt.de" descr="TUD" lat="
    49.874562" lng="8.660633" alt="176"/>
  ...
</ivu>
```

The representation of the scene in object-space (based on coordinates and lengths) so far results from the data contained in the iVu model. Since calculation of billboards require to find the visible section, first an object is added as a child to each bounding box. These objects (`IVuBillboardGPS`) essentially hold the two or three visible points of a bounding box represented by GPS coordinates.

The user-centric representation in image-space (based on viewing angles and distances) is calculated from the scene and has it's own root: the `IVuSkyline` Object. Children of `IVuSkyline` are the `IVuBillboardAngular` objects.

7.3.1 *Data Structures for Scene and Skyline*

Several data-structures were considered for the representation of scene and skyline. Especially for data in a spatial context several specialized structures are known. The benefit of using a specialized data-structure is the possibility to access sub-regions more efficiently. The negative effect is the overhead for constructing and maintaining the structure. The scene is accessed so far only iteratively and removal does not happen since the scene model does not change after the initial loading. The skyline does not change in size for the same reasons, but it is subject to sorting. Since the scene has relatively few objects it was chosen to use a simple *ArrayList*[3] for both the scene and the skyline. Switching to a more specialized structure than a list should be considered for future implementations that have an actual server to communicate with. Especially the scene container could benefit from that since it has to be judged from location and scene model when an update from the server is required.

7.4 FROM BOUNDING BOX TO BILLBOARD

The algorithms and calculations required to maintain the skyline are discussed in this section. Initially the skyline and the scene are empty lists. When a model of the environment is received, these lists get updated with the current model. Initially, a hard-coded location is used to calculate the skyline. Whenever a new location is detected, the client starts to update the data. The update mechanism starts at the bounding boxes objects, which update their visible corner-point (similar to cross-sections) child object. These update their billboard child objects accordingly. Billboard objects are also children of the skyline object, which is not involved in the update process.

3 http://developer.android.com/reference/java/util/ArrayList.html

7.4.1 *Orientated Bounding Boxes*

Every physical object in the iVu model is associated with a bounding box. The iVu Model XML syntax defines a bounding box by width, depth and height plus the GPS location and the orientation.

As a first step, that representation is transformed to four GPS coordinates; one for every corner of the box. To achieve that, a local coordinate system is constructed that is oriented in the direction of the heading. To simplify calculations on GPS coordinates, the local coordinate system is rather based on meters. Based on the latitude value of the location, the length of one degree of latitude and longitude is calculated in meters. To transform GPS coordinates into the local coordinate system, the latitude and longitude values are multiplied by the corresponding length. The advantage of that system is that, for example, distances can be simply calculated by Pythagoras' theorem, which is not possible directly with GPS coordinates. Note that the local coordinate system is only absolutely correct at the reference latitude (in east-west direction). For distances below some hundred meters (in north-south direction), the error is only marginal.

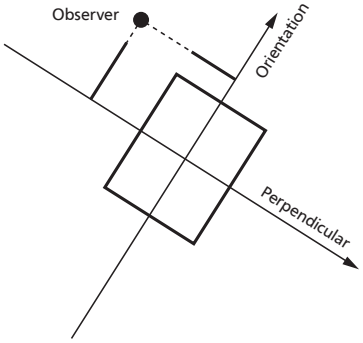
From width and depth of the bounding box, the corner-points are calculated using the local coordinate system. The four corner-points are given by $(\pm \mathit{width}/2, \pm \mathit{length}/2)$ in the local system. These points are saved in GPS coordinates as corner-points in an array.

Each of the four corner-points is assigned a number from 1 to 4, as shown in Figure 35b. To calculate a billboard, as a first step the visible corner-points of the bounding box has to be calculated. Since visibility is resolved at the end based on the billboards, these calculations are done for every billboard ignoring visibility.

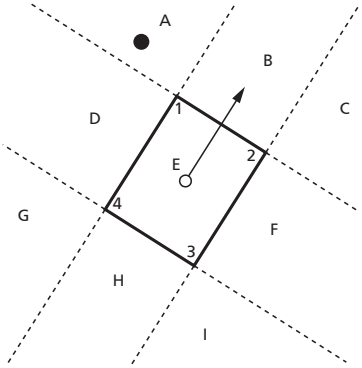
First the location of the observer is translated into the local coordinate system of the bounding box as in Figure 35a. That allows to assign the observer a sector in which she resides, as in Figure 35b. Based on that sector, the visible corner-points can be looked up in a static lookup table, as in Figures 35b and 35d. The resulting two or three GPS coordinates are stored in a child object (`IVuBillboardGps`) of the corresponding bounding box object.

7.4.2 *From Bounding Box to Billboard*

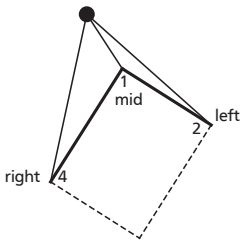
The parameters that define a billboard are two or three angles along the heading axis (i.e. horizontal direction) for each of the visible corner-points plus two angles for each point that denote the top and bottom along the tilt axis (i.e. vertical direction). That makes a total of six or nine angles. In addition to the angles, a distance needs to be calculated. Based on the object that holds



(a) Coordinates of the observer in local coordinate system



(b) A sector is given by the local coordinates



(c) Sector defines the visible corner-points

Sector	Left	Mid	Right
A	2	1	4
B	2	—	1
C	3	2	1
D	1	—	4
E	inside bounding box		
F	3	—	2
G	1	4	3
H	4	—	3
I	4	3	2

(d) Static lookup table

the visible corner-points, the angles are calculated by simple trigonometry.

For both horizontal and vertical angles, the arctangent of the direction vector has to be calculated. For the horizontal angles that would be a vector consisting of the distance in North-South (Y) direction and the distance in East-West (X) direction between the location of the observer and the current point. For the vertical angles, the viewer height, which was set to 1.7 meter, has to be considered. The bottom angle is calculated from distance to the point and the viewer height, the top angle is calculated from (building height − viewer height) and the same distance. All distances are calculated in meter.

The only thing left to calculate is the distance of the billboard. Since the distance to the two or three corner-points of the bounding box is already calculated for the angle calculations, it does not need additional calculations. The closest of these distances is set as distance of the billboard. This concludes the process required to build billboards from bounding boxes. The next step is to resolve the visibility of the billboards. The billboard objects implement the `Comparable` Java Interface (here, the distance is compared) and are stored in an `ArrayList`. That allows to call `java.util.Collections.sort(ArrayList)` which is part of standard Java.

7.4.3 *Visibility Computing Algorithm*

The visibility of billboards is solved by iterating over all billboards from closest to furthest while 'raising' the skyline. Since a building is (at least partly) visible if any part of it's top is visible, the algorithm only needs to keep track of the highest visible points. This is implemented using an array of 360 float values; one for each degree. Each value represents the maximum billboard height so far encountered in that direction. Each billboard top is tested against the height array and if it is higher than the current maximum height, then this billboard is visible and the maximum height is updated. Since the top of a billboard is only given by two or three values, the height is calculated by linear interpolation of the known values.

The implemented approach is an image accurate approach with a resolution of one degree horizontally. Note that the resolution can be easily increased by doing the same more than once per degree. Also note that this approach works with any underlying billboard type that does not include open space within.

Another value that could be of interest is the visible size or part of a billboard. This algorithm can be easily extended to calculate this information by comparing the height array before and after a billboard is processed. The algorithm scales as well linearly

with the number of billboards that are present in the scene. Each additional billboard in the scene requires as much additional tests as wide it is (in degrees). Note that billboards further away are smaller in size thus the algorithm will get faster towards the end.

7.5 SENSOR AND GPS HANDLING

One of the tasks of the client is to handle and pre-process all sensor and GPS related data. There are various ways to improve the quality of the input. Since these approaches require either hardware that is not present (e.g. a gyroscope) or are worth a thesis on their own (like visual recognition of movement or objects) the implemented pre-processing was reduced to a filter that tries to cope with the noisy orientation sensor. In addition to enhance the quality of the input, a second task is to report the quality of the data.

7.5.1 *Handling GPS Input*

Since no further enhancement of the GPS signal was implemented, all the client has to do is to report the accuracy of the location data. Android already reports the quality of every GPS-fix in terms of accuracy in meter. That value is similar to the HDOP (Horizontal Dilution of Precision) often found as output of GPS-receivers. Although it is not clear how the accuracy in meter is calculated by Android, it is probably based on the HDOP. Tests have shown that this indicator is mostly accurate. The also available *number of satellites* used in the fix showed to be a clearly inferior metric since additional satellites that are close to others have only a minor effect on the accuracy.

7.5.2 *Handling the Compass Input*

The data received from the orientation sensor is pre-processed before use. Since the raw data of the orientation sensor showed an unacceptable amount of noise in test, the signal needs to be stabilized to be usable. The noise of the sensor is unfortunately not only of high frequency. A very slow changing part[4] that can not be distinguished from movement caused by the user has to be accepted. Also prediction of future orientation is not possible, since the device does follow the arbitrary movements of a human being. Based on that, only a filter that calculates an average of past values was considered (Listing 3).

4 The constantly changing results could also be caused by a slightly changing magnetic field.

Listing 3: Low-pass filter for a single value in Java

```
class LowpassFilter {
  private float output;
  private static float factor = 0.2f;

  void put(float input) {
    output = input * factor + output * (1-factor);
  }

  float get() { return output; }
}
```

Note that the amount of filtering can be adjusted by the *factor* to values between 0 and 1. The smaller the value the more the algorithm weights old values, resulting in smoother output. The higher the smoothing the more lag is introduced. Note that this low-pass filter is a Kalman filter with constant prediction (the next state will be this state) and a constant error estimation. The Kalman gain will converge to a constant value which is the smoothing factor here.

To deal with wrong orientation caused by a distorted magnetic field, iVu.KOM specifies the use of the feedback data users provide. Since without an iVu.KOM server this mechanism is incomplete, the implementation of the feedback concept was done inside the GUI application.

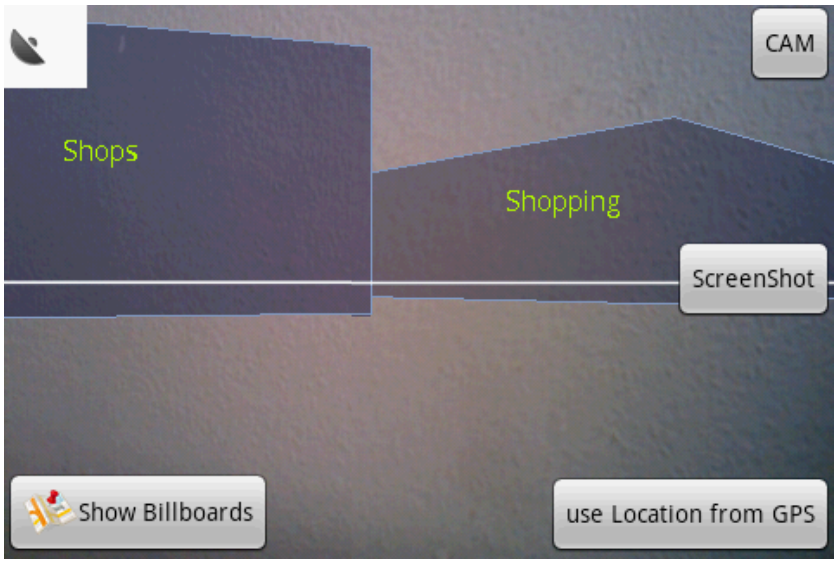
Monitoring the quality of the orientation data was implemented. Two values that have significance for the quality were identified. Firstly, the total field strength as observed by the sensor can be compared to a reference value given by a geomagnetic model. Secondly, the declination angle can be compared to the reference value of the same model. Both reference values can be computed by Android (`android.hardware.GeomagneticField`). Reference and measured values are provided to the application.

7.6 The implemented GUI application

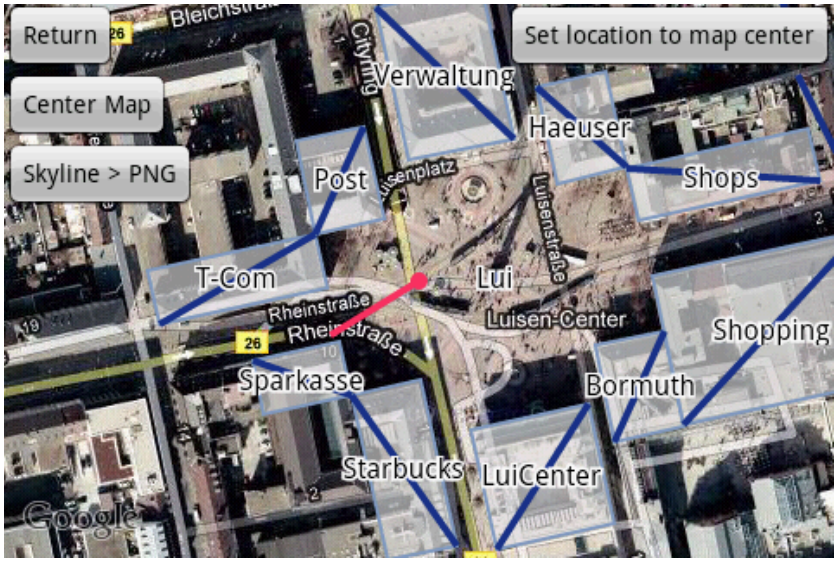
To be able to test the framework a GUI application is required. It uses the iVu.KOM client's API to visualize the results and implement the feedback mechanism described in Chapter 4.

The application consists of two Android activities. The main activity is an augmented reality viewer (Figure 35a) that uses the built-in camera as a base. The current state of the billboards is drawn on top of the live video from the camera. This allows to visualize the behavior of the framework in real-time. Delays, sensor noise and errors in calculation are all visualized.

The second activity of the application is mainly used for debugging purposes and is basically a satellite image map provided by the Google Maps API. The location and heading of the user as



(a) Augmented Reality interface



(b) Map view

Figure 35: The implemented feedback interface

well as bounding boxes and cross-sections are overlaid over the map (Figure 35b). This view makes it also possible to manually set the location, which is useful for testing purposes.

7.7 OPENSTREETMAP.ORG SUPPORT

As mentioned in Chapter 2, OpenStreetMap is used as a source for the 2.5D geometry model in this thesis. To use this source, an XSLT definition and a standalone Java tool were developed. Both allow to transform the OpenStreetMap XML format into iVu.KOM's XML format. Since buildings are represented by polygons with a variable number of points, an algorithm that creates a bounding box around these points was implemented. Also the iVu.KOM XML syntax was extended to allow a polygonal representation of buildings. Note that the XSLT approach is considered deprecated as the construction of bounding boxes is the task of the iVu.KOM server.

For bounding box creation, an optimal orientation has to be found that yields the smallest box. The implemented algorithm tests all orientations that are given by the connection of two consecutive points of the polygon. In most cases (rectangular buildings for example), the resulting bounding box will be optimal. For arbitrary buildings, the box is still good but not optimal. For testing purposes the results are sufficient.

The resulting iVu model shows the limits of the bounding box concept. Boxes are overlapping very often and since no automatic splitting of buildings into multiple boxes was implemented, the boxes are typically very large. Compared to the manually created models, the results of this approach are clearly inferior. Also the height of buildings is uniformly set to a constant value which results in a further lowered quality. The height itself was surveyed with a distance LASER meter.

8 EVALUATION

This chapter presents the validation of the work presented in this thesis. The validation covers the following:

- iVu.KOM query accuracy under different real world conditions like real GPS, compass and accelerometer data, and POI and 3D geometry distributions. The evaluation is based on the prototype described in Chapter 7. The chapter describes the methodologies and results for evaluating the query accuracy in outdoor settings. The effectiveness of using the feedback tool is also evaluated.
- WBroximity using real dataset and simulative setup, in Section 8.2

8.1 IVU.KOM QUERY ACCURACY

Query accuracy measures the correctness of the query results with respect to their spatial reference. This means that a point-to-find/tag query should identify the actual object being pointed to, and a point-to-browse/search query should return the visible POIs, i.e. those lying within the actual FoV and not being blocked by any object.

8.1.1 *Test Procedure and Test Environments*

The test procedure is based on *pointing*, i.e. to use the mobile device to point at well-defined targets. The goal of the pointing tests is to provide a qualitative evaluation of the effects of sensor uncertainty on the accuracy achieved for the viewer-centric queries. Another goal is to assess the effectiveness of the user feedback tool in increasing the query accuracy.

As the sensors readings depend dramatically on the surrounding environment, it is essential to test the system in representative environments to gain reliable insights of the system behavior under different real world conditions. These environments should represent the different kinds of locations in which a user of iVu.KOM can typically exist, mainly as a *pedestrian* user. For this purpose we differentiate between the following classes of test environments with respect to expected sensor disturbance:

8.1.1.1 *Low-disturbed Environment*

This environment is characterized by low GPS and compass disturbance. It can have medium sized buildings (2 to 3 floors) but are spaced relatively far away from the user. Therefore, buildings obstruct no or few portion of the sky and their structures have low influence on the compass. The area has also no or low electromagnetic interference. Such interference can be caused e.g. by tram power lines and normally can cause the compass heading to deviate from magnetic north. A typical example is the Herrengarten public garden in Darmstadt as shown in Figure 36.

Figure 36: An example of low-disturbed environment: side of the Herrengarten in Darmstadt

8.1.1.2 *GPS-disturbed Environment*

This terrain is characterized by some GPS disturbance and low or no compass disturbance. A typical example is a park environment with trees blocking considerable portions of the sky. It can also have few buildings which are relatively placed far away from the user as shown in Figure 37a.

8.1.1.3 *GPS- and compass-disturbed Environment*

This terrain is characterized by some GPS and compass disturbance. It has medium to large size buildings that are closely spaced from each other, narrow streets and walkways. Therefore, considerable portion of the sky is blocked and building structures are expected to affect the compass. Also it may have power lines impacting the compass further. Typical examples include down centers and streets equipped with the tram electric cables. An example area where we ran our tests is shown in Figure 37b.

It is emphasized here that there is no way to specify sharp quantitative values to describe a specific test environment. Besides, even in the same test environment more than one environment type can be encountered depending on the relative location of the user with respect to the objects. The same test environment may even exhibit different properties with respect to the expected sensor disturbance depending on the time of the year or the weather conditions. For example, trees losing their leaves in winter may

(a) (b)

Figure 37: Examples of (a) GPS-disturbed area and (b) GPS and compass-disturbed area.

become less obstructive to GPS signal, or on the contrary to that, wet trees can cause more attenuation to GPS signal.

Pointing tests were carried out in the aforementioned test environments using the feedback tool of the iVu.KOM client. The software was run on an HTC G2 mobile device which has an integrated GPS unit, a 3-axis magnetic sensor and a 3-axis accelerometer. In each test environment a number of pointing samples were collected by pointing at different target objects from different positions. Each pointing sample contains two modes: without user feedback (hence after designated as *no-feedback*) and with user feedback (hence after designated as *with-feedback*). The procedure to create a pointing sample in the two modes is as follows:

1. No-feedback mode. The mobile device is pointed towards the target object. The feedback tool will then display the target overlaid with the currently computed skyline. On pressing the camera button, a no-feedback pointing sample is stored as a KML file containing the following data:
    - The actual GPS position ρ_{act}. For this purpose, the iVu.KOM client provides a map view to pinpoint the actual position at a sufficiently large zoom level. Distinguishing marks like ground markers and building edges helped the author set the actual position on the map with very good accuracy,
    - The estimated GPS position ρ_{est} as measured by the GPS receiver,
    - The estimated GPS accuracy $\Delta_{\rho,est}$ as measured by the GPS receiver. This is reported as the Circular Error Probable (CEP) (defined as the radius of the circle within which 50% of the position fixes would fall [61]),
    - The actual GPS accuracy $\Delta_{\rho,act}$ computed as $|\rho_{act} - \rho_{est}|$,

- The estimated compass heading ϕ_{est} in degrees as measured by the compass,
- The estimated magnetic field strength ψ_{est} (in μ Tesla) as measured by the compass. This is computed as the square root of the sum of the squares of the x, y and z components of the magnetic field.
- The supposed value of the magnetic field as reported by the compass. This is essentially derived from the World Magnetic Model WMM (a model that captures the variations in the magnetic field intensity over space and time [48]). Therefore, this value is considered as a ground truth reference,
- A serialization of the estimated skyline, i.e. based on ρ_{est},
- A serialization of the actual skyline, i.e. based on ρ_{act}, and
- The current camera snapshot including the actual image and the overlaid actual skyline. Here, only part of the skyline view is displayed depending the camera view port (viewing angle) and the compass heading ϕ_{est}.

2. With-feedback mode. Here, the mobile device is pointed as in step 2 towards the object and by pressing the 'Give Feedback' button a snapshot of the target object is taken and overlaid with the current skyline. The skyline in this case is movable, i.e. it can be shifted arbitrarily to the left and the right. To give a user feedback, the skyline view is therefore shifted until it matches with the underlying image. The amount of the shift represents the compass offset Δ_ϕ which is summed to ϕ_{est} to yield the actual compass heading ϕ_{act}. After the feedback is done, the device is pointed again to the target and the KML file is created as in step 2.

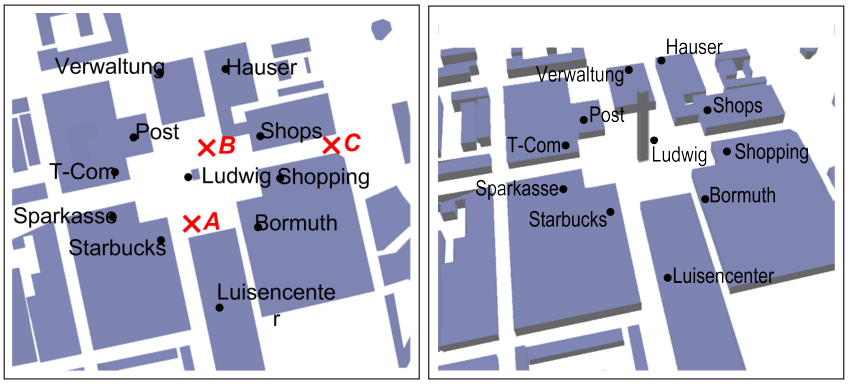
Table 4 summarizes the different parameters included in the samples. Having the samples saved as KML files allowed easier post-anaylsis by having the samples directly displayed in Google Earth and trivially associating each sample to the respective test environment. The test 3D data itself was surveyed from different sources, including the OpenStreetMap [12] which provides the 2D fingerprints of the buildings in the different test environments, and field measurements of the building heights using a long-range distance LASER meter. The amount and quality of the 3D test data collected this way is enough for the purposes of the pointing tests.

Parameter	Description	Value Source
ρ_{act}	Actual position	Manually on the map
ρ_{est}	Estimated position	From GPS receiver
$\Delta_{\rho,act}$	Actual GPS accuracy	$= \lvert\rho_{act} - \rho_{est}\rvert$
$\Delta_{\rho,est}$	Estimated GPS accuracy	From GPS receiver
ψ_{est}	Estimated magnetic field strength	From compass
ψ_{act}	Actual magnetic field strength	From compass or WMM
ϕ_{est}	Estimated compass heading	From compass
Δ_{ϕ}	Offset in compass heading	Skyline shift during feedback
ϕ_{act}	Actual compass heading	$= \phi_{est} + \Delta_{\phi}$

Table 4: Parameters used in the pointing tests

8.1.2 *Tests in GPS- and compass-disturbed Environment*

These tests were run in the Luisenplatz square in the city of Darmstadt (shown in Figure 37b). The geometry in the area is shown in Figure 38, in top view (left) and 3D view (right). A total of three representative test locations were selected in this area (marked *A*, *B* and *C* in Figure 38). Location *A* represents a location where a user is nearby a building on one side, location *B* represents a location where a user is standing relatively far away from the buildings, and location *C* represents a location where a user is standing between two buildings that are spaced close from each other. For each of the three locations, 10 pointing samples (each in both no-feedback and with-feedback modes) were collected using the aforementioned procedure. Different target objects were selected during the tests, including the Ludwug status in the middle of the square, the Starbucks cafe and the Luisencenter shopping mall (all targets are labeled in Figure 38).

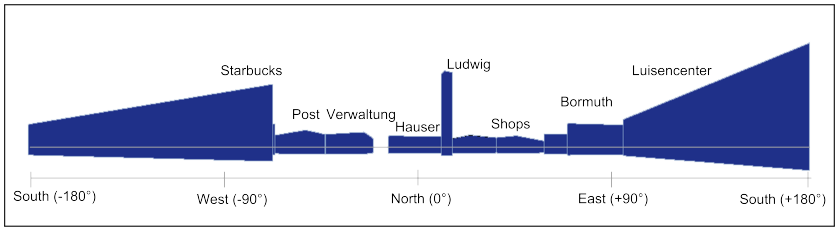


(a) (b)

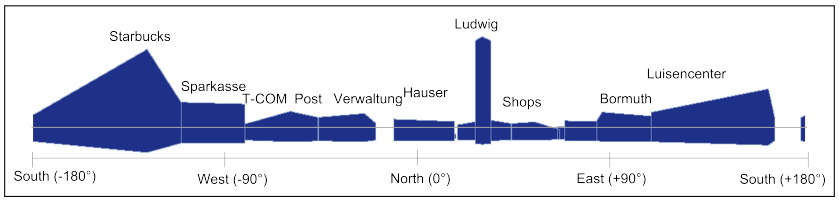
Figure 38: Geometry in GPS-compass-disturbed test area from (a) top view and (b) 3D view.

Test Location *A*: The pointing samples collected at this location yielded an average $\Delta_{\rho,est}$ of 6 meters and an average $\Delta_{\rho,act}$ of 11.8 meters. Figures 39a and 39b respectively depict the actual and estimated skylines at location *A* for one of the pointing samples. Comparing these figures, it is straightforward to inspect the variations in the sizes and visibility of the billboards depending on the current position. For example, while the the object 'T-COM' is not visible from the actual position, the estimated skyline view shows that this object will occupy considerable portion of the FoV. It is important to stress here that these figures provide qualitative estimation of the distortion of the skyline view because of the GPS disturbance. It is hard to derive a quantitative measure as the amount of distortion depends largely on the distribution of the geometry in the environment and the actual location of the user.

The effect of the distortion in the compass heading is best explained using the skyline-overlaid images. Pointing samples at location A yielded an average ψ_{est} of 63.21 μ Tesla, whereas the ψ_{act} is supposed to be about 48.50 μ Tesla according to the WMM. The increase in the magnetic field intensity can be attributed to the magnetic fields induced by the metallic structure of the nearby building (the Luisencenter in Figure 38a). As the tests showed, moving farther from the buildings, e.g. towards location B in the center, ψ_{est} exhibits a value that is closer to ψ_{act} (e.g. $\psi_{est} \approx 53.26$ at location B). Figure 39c shows the skyline when pointing at the target 'Ludwig status' in the no-feedback mode. The figure shows large discrepancy between the actual target (at $\phi_{act} \approx 14.8°$) and the overlaid skyline (at $\phi_{act} \approx 70.6°$). Repeating the test in the with-feedback mode, Figure 39d depicts very good matching between the target and the skyline after introducing a compass offset $\Delta_{\phi} \approx +56°$.



(a) Skyline based on ρ_{act}



(b) Skyline based on ρ_{est}

(c) No-feedback skyline

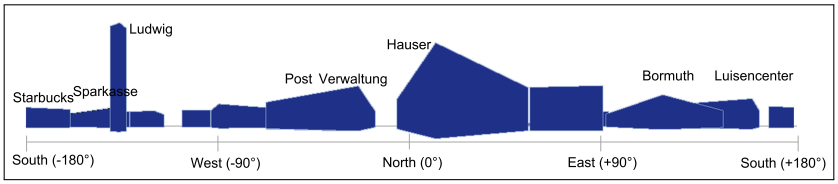
(d) With-feedback skyline

Figure 39: Pointing test at location A.

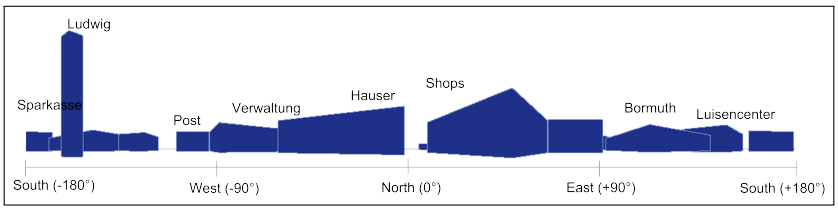
Test Location *B*: The pointing samples collected at this location yielded an average $\Delta_{\rho,est}$ of 6 meters and an average $\Delta_{\rho,act}$ of 5.8 meters. Figures 40a and 40b respectively depict the actual and estimated skylines at point B for one of the pointing samples. As shown, the discrepancy in billboards sizes due to GPS error is

less than that for the test point A. This can be attributed to the better position estimation, probably due to less GPS shadowing than the case of location A.

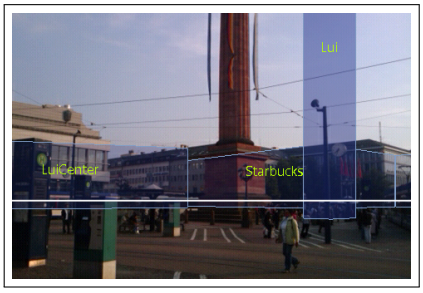
Pointing samples at point B yielded an average ψ_{est} of about 53.26µ Tesla, which is, as previously mentioned, is notably closer to the value reported by the WMM (about 48.50µ Tesla). Figure 40c shows the skyline when pointing at the same target as in point A tests (i.e. the 'Ludwig status'), in the no-feedback mode. The figure shows now less discrepancy between the actual target and the overlaid skyline (about 14° offset). Figure 40d shows the pointing result after compensating the compass offset. iVu.KOM displays in this case almost exact match between object and its billboard.



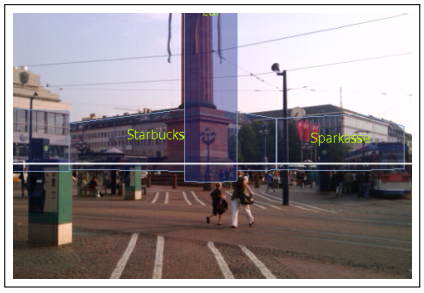
(a) Skyline based on ρ_{act}



(b) Skyline based on ρ_{est}



(c) No-feedback skyline

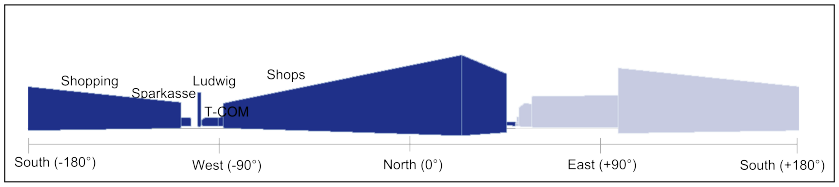


(d) With-feedback skyline

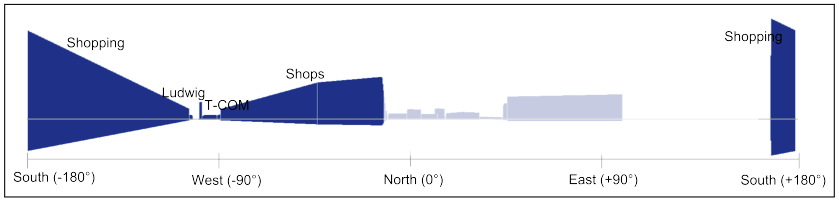
Figure 40: Pointing test at location B.

Test Location *C*: The pointing samples collected at this location yielded an average $\Delta_{\rho,est}$ of 6 meters and an average $\Delta_{\rho,act}$ of 13.6 meters. Figures 41a and 41b respectively depict the actual and estimated skylines at location C for one of the pointing samples. Interestingly, from the two figures it can be inferred that the value of the estimated position is selected such that it is farther a way from the pointing target (again, the 'Ludwig') and displaced to the left so that the building 'Shopping' looks magnified. In the two figures, the portion of the skyline which

is highlighted with a brighter shading represents objects to the east of the location *C*, i.e. they are basically outside the test area already shown in Figure 38.

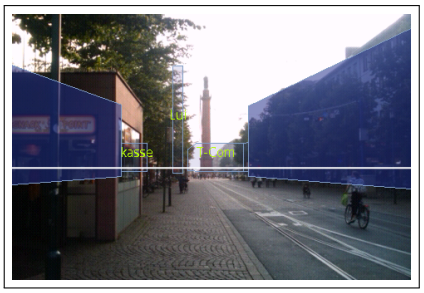


(a) Skyline based on ρ_{act}



(b) Skyline based on ρ_{est}

(c) No-feedback skyline



(d) With-feedback skyline

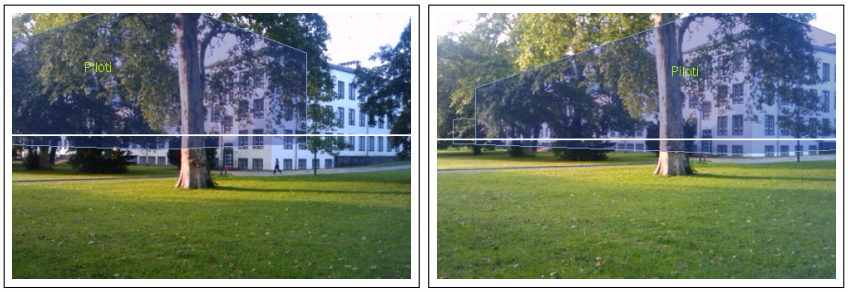
Figure 41: Pointing test at location *C*.

The magnetic field at location *C* behaves to some extent differently form that at both locations *A* and *B*. An average ψ_{est} of about 58.14µ Tesla was recorded, which lies between the values at locations *A* and *B*. However, while the estimated heading (about 18°) was far from the reality as shown in Figure 41c, the skyline was also oscillating around that heading. This can be attributed to the varying electromagnetic interference of the tram power lines, which laid directly above the test location *C*. Giving feedback was able to reduce the overall heading offset as shown in Figure 41d, however the heading continued to oscillate.

8.1.3 *Tests in GPS-disturbed Environment*

These tests were run in the Herrengarten in Darmstadt. In some parts, the Herrengarten has huge and dense trees, which is assumed to be a good representation for a GPS-disturbed environment. The results of the pointing tests here support this assumption. While on average the position was recorded with a deviation $\Delta_{\rho,\,act}$ of about 8m, the magnetic field ψ_{est} had values

that are very near to the nominal value (about 48.56μ Tesla). This indicates that this area has very low magnetic disturbance. Figure 42a shows the overlaid skyline for one of the samples based on $\Delta_{\rho,est}$. A clear offset can be observed between the object and its skyline. To verify if the source of this mismatch is the GPS or compass, the actual position $\Delta_{\rho,act}$ was pinpointed on the map and the pointing sample was repeated from the same location. The result of the second run is depicted in Figure 42b and it shows almost exact match between the object and its skyline, even without giving any feedback to adjust the compass.



(a) Skyline based on $\Delta_{\rho,est}$ (b) Skyline based on $\Delta_{\rho,act}$

Figure 42: Pointing test in a GPS-disturbed environment

8.2 WBROXIMITY EVALUATION

For this evaluation, we used the WBroximity frontend to collect real world fingerprints in locations like our laboratory and the university campus and over a period of two weeks and different times of the day. As some of the WLAN networks are built up by several access points, we considered each access point as a single hot spot and used its MAC address as a BSSID. For example, throughout our building a total of 5 WLAN networks can be detected and are formed by 26 access points. Having collected the dataset, we applied to them data analysis techniques to study the applicability of different machine learning algorithms to the positioning task. We evaluated the performance trends in light of fingerprint characteristics and the impact of user participation.

8.2.1 *Setup*

8.2.1.1 *Classifier Selection*

The initial step in our analysis was to choose a scheme to recognize fingerprint patterns, and then to use such scheme as a baseline for the rest of the evaluation. For this purpose, we selected subsets of the dataset and applied to them these classifiers: J48, NB Tree, REP Tree, Naïve Bayes, Logistic, JRip, PART and

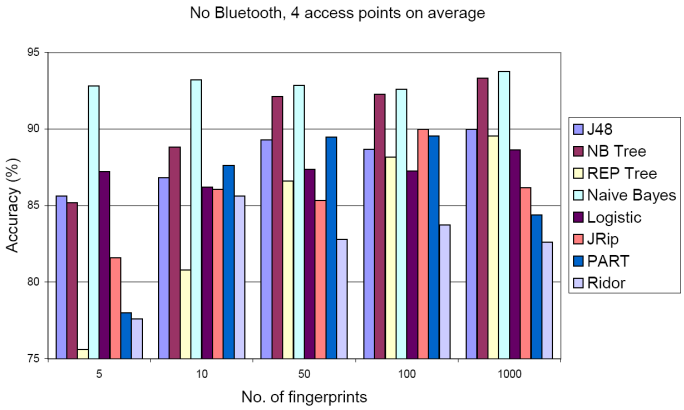


Figure 43: Performance of different classifiers

Ridor[1]. As shown in Figure 43, with an accuracy of at least 92%, Naïve Bayes (NB) performed best among the set of classifiers and for all test runs. This can be attributed to the fact that NB has strong independence assumptions, which match quite well the characteristics of the underlying fingerprints. In particular, NB assumes that the presence of a fingerprint is unrelated to the presence of any other fingerprint. This corresponds to real world situations where a hotspot suddenly disappears at a certain location because the base station is powered off or the signal becomes abruptly too weak to be detected. However, if a single network disappears, normally it does not affect the other networks. Based on this analysis, we adopted NB for the rest of our evaluation.

8.2.1.2 *Selecting Fingerprint Attributes*

The kind of attributes (BSSID, RSS, LQI) included in the collected fingerprints relates directly to the storage and processing requirements of fingerprints. Therefore, besides selecting the best classifier, it is useful to decide at an early stage which attributes are critical for the positioning task and which are unnecessary, if any. To assess this aspect, we tried omitting different attributes and attribute combinations, and measured the obtained accuracy (again for NB). Figure 44 stresses that LQI is essential for positioning. This is due to the fact that in locations which are close to each other, most often the same networks will be seen, so the BSSID alone does not provide much information. Interestingly, the BSSID can even be completely left out without affecting the accuracy, assuming that each sampled location has a characteristic LQI. However, if we consider a wide area where networks are disjoint, then the BSSID can play a role in distinguishing the fingerprints.

1 For a detailed description of the tested classifiers, the reader is referred to the documentation of the Weka tool [35].

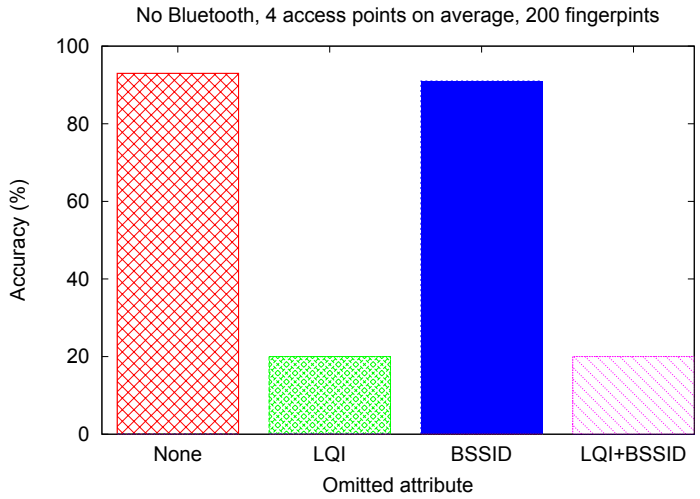


Figure 44: Effect of attribute omission/inclusion

8.2.2 *Integrating Bluetooth Information*

Bluetooth fingerprints can exhibit temporal dependency because mobile Bluetooth neighborhoods are encountered more often than stationary ones. Therefore, we examined the effect of integrating Bluetooth over time. For this purpose, we injected our real data with synthesized Bluetooth fingerprints. Figure 45 shows the effect of injecting one static Bluetooth fingerprint per user per time unit. Obviously, with Bluetooth the system classifies correctly more often. The reason is that Bluetooth networks are only visible at one location: thus, detecting one of these networks is equivalent to detecting an individual location, like a room. While we made this evaluation using static Bluetooth networks, mobile networks can be included as well in the fingerprints without causing the overall positioning accuracy to degrade, as they will not be contributing any additional information.

8.2.3 *Effects of Participatory Sensing*

WBroximity is affected by two inherent factors of participatory sensing: the number of participants and the quality of user participation.

8.2.3.1 *Effect of the Number of Participants*

We measured the accuracy over time when different number of users are participating. As shown in Figure 46, after the same amount of time units, as expected, the system depicts higher accuracy for higher number of users. However, no matter what the number of users is, the system tends to learn exponentially, and converges towards a maximum accuracy (about 90%) after

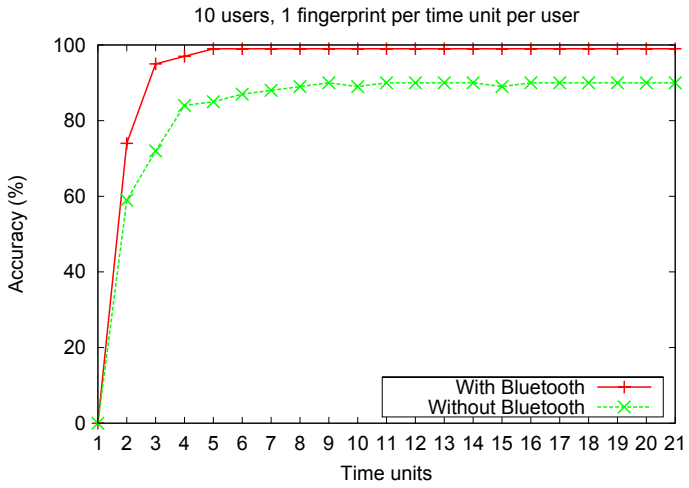


Figure 45: Effect of Bluetooth integration

a while. This is actually an interesting observation, because it suggests that for a desired level of accuracy, we need to collect a sufficient number of fingerprints. For example, to reach the 90% accuracy at a specific location, it is enough to collect about 200 fingerprints. Although it looks irrelevant at a first glance if this amount of fingerprints comes from the same user or different users, it is beneficial in a participatory sensing scenario to have these fingerprints coming from different users to increase the trustworthiness of the gathered data.

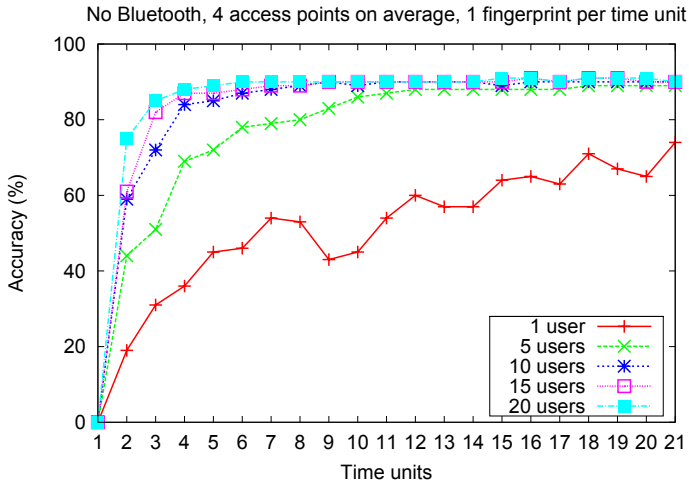


Figure 46: Effect of number of users

8.2.3.2 *Effect of the Quality of User Participation*

As users have typically no access to tamper with the collected fingerprints, the only remaining factor affecting the quality of user participation is the user-generated label. In practice, different users may provide different labels for the same location, or

even the same user may at different times provide different labels for the same location. In the worst case, malicious users may intentionally mislabel the fingerprints. To evaluate this aspect, we mimicked bogus participation by reassigning wrong labels to different number of fingerprints in our dataset. Figure 47 shows that accuracy gradually degrades as the ratio of mislabeled fingerprints increases. However, the system still depicts an accuracy over 90% after introducing 20% wrong labels.

8.2.3.3 *Mitigating the Effect of Mislabeled Fingerprints*

As a countermeasure against wrong labels, we adapted WBroximity such that it restricts the range of symbolic labels a user can assign to fingerprints. As shown in Figure 32d, WBroximity returns a list with the four most probable locations. The user can then select one of these labels only. Figure 47 shows the accuracy when the countermeasure is being applied. A 90% accuracy is still achievable even when 80% of the fingerprints are mislabelled. However, this approach has the downside that it limits the freedom of honest participants to correct the system. Therefore, we initially provided some users with administrative privileges to extend the set of labels, if required.

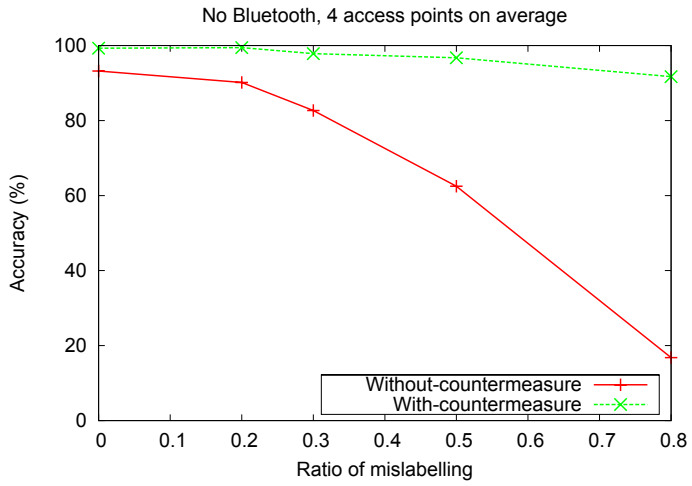


Figure 47: Effect of wrong labels and countermeasure

9 CONCLUSION AND OUTLOOK

This thesis investigated the design of a framework and a model to enable mobile location-based services which provide information to the user about objects she can really see from her current position. The thesis addressed the modeling of viewer-centric queries from the GPS and orientation sensors increasingly being embedded in emerging mobile devices. As these sensors commonly suffer from uncertainties, specially at runtime, the thesis aimed at evaluating the functionality of the proposed framework and modeling concepts under real world conditions. The evaluations were conducted using a prototypical implementation of the iVu.KOM framework on top of the Android mobile platform. For the time being, the prototype was designed and tested mainly for outdoor application. Nevertheless, the framework is equally applicable to indoor purposes, assuming the appropriate query models are adopted.

The following sections wrap up the achieved results and provide highlights of possible directions for future research.

9.1 ACHIEVED RESULTS

The results achieved in this thesis can be grouped mainly as follows:

9.1.1 *Framework and Model*

In this thesis, the iVu.KOM framework forms a key contribution where it provides the required abstraction levels in the process of creating and running viewer-centric mobile LBS:

- as described in Chapter 3, the framework is designed with an extensible and open architecture. On the client side, the framework works on the orientation and positioning information, which can be supplied by any provider. For example, the position can be provided by a self-positioning unit (e.g. GPS) or a network-based service (e.g. based on Cell ID). The framework consumes as well the iVu model and outputs it as a skyline with an application-neutral data model. By using the appropriate transformation, this data model can be translated into a presentation format that fulfills the needs of the interface implemented on the mobile device. Therefore, different kinds of applications or

interfaces can benefit from the same information without need to adapt the server response to each individual interface. On the server side, the framework interfaces with any kind of spatial database, which is only required to support simple spatial range queries. The framework accepts also points of interest from any source, e.g. as a result of a search engine query, or as a query to a dedicated database.

- the framework is offered as modules with appropriate Application Programming Interface (API). Therefore, the internals of running the queries and producing the skyline are kept transparent to the application developer, as described in Chapter 4.

- the framework supports both retrieving and generating content. This is made available by supporting different kinds of queries like the *point-to-find* and *point-to-tag*.

While the above results concern mainly the functional requirements outlined in Chapter 1, the framework has equal focus on the non-functional requirements:

- the framework offers means to display the overall accuracy of the executed query. Displaying this information besides the information in the query result makes the user aware of the quality of the information and raises the usability of the service. Nevertheless, the accuracy should be presented in a way appropriate to the interface type implemented by the application. Chapter 5 describes the design of an interface for augmented reality applications. Moreover, the design uses the skyline concept as a design resource, which adds more value to the skyline.

- besides presenting information about the accuracy of the query, the framework offers means to allow the user to provide feedback to this accuracy. The interface design in Chapter 5 also offers this possibility by allowing the user to modify the orientation of the skyline superimposed on the camera image. The framework is able to consider the user's feedback through the query process.

The client components of the framework as well as the augmented reality interface were implemented as a prototype for the Android platform. This implementation was sufficient for the purposes of the evaluation done in this thesis. The server modules where not implemented but were substituted with geometry models for the test areas in the city of Darmstadt. The models were loaded directly on the client.

Through intensive pointing tests done carried out in representative environments, the following can be drawn:

- in sufficiently open areas, with low density of obstacles and magnetic interference, the framework delivers a level of accuracy, which is adequate for a viewer-centric interaction. This holds even when using low-cost GPS and orientation sensors that are found in the mobile devices on the market.
- in urban areas, which normally have dense buildings and magnetic interference (e.g. due to tram's overhead power lines), the delivered accuracy can degrade, which affects the overall user experience.
- the skyline-based feedback interface proved very useful and direct means to recognize query accuracy. Using the interface to give feedback manually proved also useful to improve the query accuracy. However, it was reported during the tests that it will be appreciable in a mobile setting if the effort to give the manual feedback can be reduced. This point is discussed further as a future work, in Section 9.2.

9.1.2 *Applying Participatory Sensing for Positioning*

WBroximity, as explained in Chapter 6, provides a positioning solution that uses both WLAN and Bluetooth fingerprints. Furthermore, such hybrid fingerprints are collected by using the paradigm of participatory sensing, thus cutting the extra costs needed to employ special personnel for this task, and allowing the system coverage to expand to wherever participants reach.

A prototype for WBroximity was realized as a client for Android platform. The server modules were implemented on top of the ContextFramework.KOM [56]. The system was evaluated both in simulation and using the prototype. Evaluation results show that:

- needed fingerprint attributes: the Link Quality Indicator (LQI) is found essential for positioning. This is due to the fact that in locations which are close to each other, most often the same networks will be seen, so the network ID (BSSID) alone does not provide much information. Interestingly, the BSSID can even be completely left out without affecting the accuracy, assuming that each sampled location has a characteristic LQI.
- role of Bluetooth: with stationary Bluetooth networks, the system is found to provide better positioning accuracy. The reason is that Bluetooth networks are mostly only visible at one location. Mobile Bluetooth networks generally do not lead to better accuracy, as they do not convey location-specific fingerprints.

- role of participatory sensing: as the number of participants increases, the accuracy increases accordingly. In fact, the system is found to learn exponentially as the number of provided fingerprints for a certain location increases, regardless if these fingerprints are provided by one or more users. However, in practice it is favorable if the fingerprints are coming from different users to count for malicious users.

WBroximity can be used seamlessly as a location provider, similar to GPS and network-based positioning. iVu.KOM is one example case which can benefit from this, specially in indoor settings.

9.2 FUTURE WORK

As next steps to the work conducted in this thesis, the following research directions are identified:

- Towards enabling viewer-centric mobile location-based services, the issue of accuracy can still receive further research. In particular, one viable research direction will be the design and application of algorithms to recognize sensor errors and to automatically react to such errors. This is essential and can be combined with the technological advancements which promise more accurate positioning (like Differential GPS) and more accurate heading (like gyroscope-compensated compasses). For usability reasons, it is also interesting to design means to present and visualize the uncertainty in the retrieved information. Such means should be intuitive to the mobile user. Further research should also investigate the role of location- and time-based caching when combined with the iVu model. Factors like the velocity of the user, the density of buildings and points of interest, and the caching capabilities of the mobile device, are all interesting when it comes to caching.

- To further understand the potential of using participatory sensing for purposes of positioning, it is essential to research the issues related to quality of user participation and the design of incentives for participation. Trends in the system performance can be in practice investigated if the system is deployed to a broad audience base. Another open issue with WBroximity is the usage of GPS labels instead of the currently used symbolic labels. This has the advantage that same place is labeled with same coordinates. Nevertheless, as GPS coordinates are provided as triples (longitude, latitude and altitude), this makes the classification a bit

more complex. Research should investigate machine learning approaches which can handle this multi-dimensional classification problem efficiently.

APPENDIX

A

WBROXIMITY IMPLEMENTATION FOR ANDROID PLATFORM

WBroximity designed to harmonize with standard location providers

One design goal for WBroximity was to use it as a location provider in the same manner other location providers are used. The Android development environment offers two standard location providers:

- `GPS_PROVIDER`: this determines the position using the GPS satellites, and
- `NETWORK_PROVIDER`: this determines the position based on the availability of network (e.g. GSM) cell towers.

Custom location providers can be seamlessly plugged in the application stack by extending the `LocationProviderImpl` class. As shown in Figure 48, analogous to the location providers `GPS_PROVIDER` and `NETWORK_PROVIDER`, the `WBROXIMITY_PROVIDER` offers the methods needed to enable, disable and query the status of the location provider. The specific functionality of the WBroximity provider is offered through two methods:

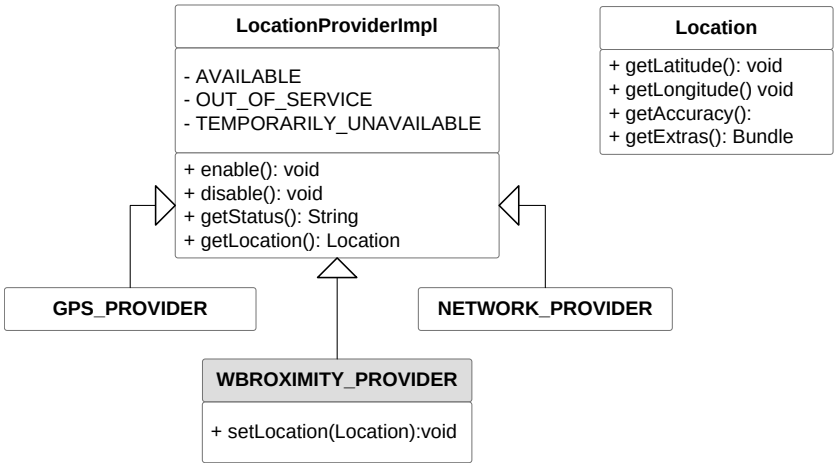
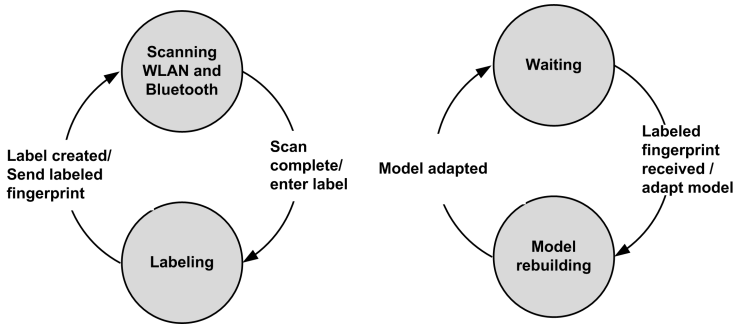


Figure 48: WBroximity as a location provider for Android

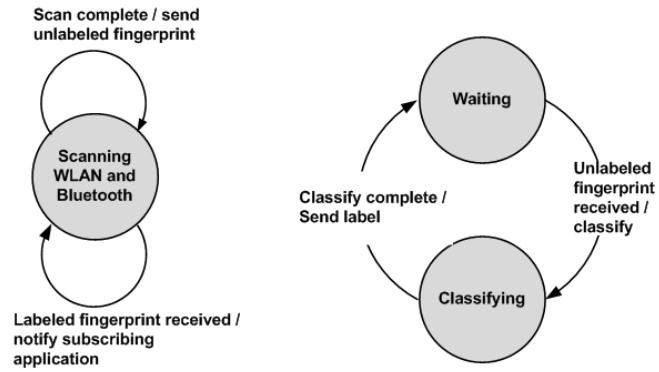
- The `getLocation()` method: this method provides the latest location fix by scanning the WLAN and Bluetooth fingerpints and sending them for evaluation by the WBroximity server. The return value of the method is a `Location` object as shown in Figure 48. This object holds the GPS longitude and latitude coordinates of the location fix, location accuracy, as well as a key-value map (accessed through the

`getExtras()` method), which stores additional information specific to the location provider. Therefore, this map is a good place to store the fingerprint label.

- The `setLocation()` method: this method is specific to the `WBROXIMITY_PROVIDER` (not existing in the base location provider or other providers). This method stresses the fact that WBroximity is based on participatory sensing, where the users can provide location information too. The `Location` object passed to this method will typically contain the user-provided fingerprint label plus a GPS location that is either automatically or manually determined, as explained next.



(a) State transition in training phase: client (left) and server (right)



(b) State transition in classification phase: client (left) and server (right)

The server modules of WBroximity are realized as a module in the ContextFramework.KOM [56], which provides, among others, an evaluation service that seamlessly interfaces to the different machine learning algorithms provided by the Weka tool [35].

A position fix can be delayed due to required scan period

Figures 49b and 49a summarize the state transitions for the WBroximity client and server during the training and classification phases, respectively. Due to the nature of standard WLAN and Bluetooth network interfaces, applications should wait until the next scan period to finish before a new position fix can be generated. Therefore, applications should generally get the locations in a publish-subscribe manner. Applications may alternatively use the position fix produced by latest scan period. However,

the freshness of the position fix should be checked against the requirements of the application[1].

1 An online demo of WBroximity can be found here http://www.kom.tu-darmstadt.de/~fzaid/wbroximity.html

B

XML FORMAT OF OPENSTREETMAP

Listing 4 shows the XML description for the fingerprint "Darmstadtium´´ in Figure 8a. The rest of the fingerprints and some redundant information were omitted due to the verbosity of the syntax.

Listing 4: Fingerprint XML description in OpenStreetMap

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
 <bounds minlat="49.8732200" minlon="8.6571870" maxlat="
     49.8750740" maxlon="8.6611350"/>
 <node id="277280358" lat="49.8745188" lon="8.6588539" />
 <node id="295349803" lat="49.8744529" lon="8.6586876" />
 <node id="295155196" lat="49.8743839" lon="8.6588223" />
 <node id="295349804" lat="49.8742582" lon="8.6582974" />
 <node id="277280359" lat="49.8742716" lon="8.6576947" />
 <node id="277280360" lat="49.8742124" lon="8.6577022" />
 <node id="277280361" lat="49.8739797" lon="8.6568383" />
 <node id="277280362" lat="49.8744913" lon="8.6566755" />
 <node id="295155200" lat="49.8746499" lon="8.6573102" />
 <node id="295155203" lat="49.8746533" lon="8.6576076" />
 <node id="277280363" lat="49.8748488" lon="8.6582718" />
 <way id="25442519" user="pabrabbin" timestamp="2010-06-26
     T07:50:03Z">
        <nd ref="277280358"/>
        <nd ref="295349803"/>
        <nd ref="295155196"/>
        <nd ref="295349804"/>
        <nd ref="277280359"/>
        <nd ref="277280360"/>
        <nd ref="277280361"/>
        <nd ref="277280362"/>
        <nd ref="295155200"/>
        <nd ref="295155203"/>
        <nd ref="277280363"/>
        <nd ref="277280358"/>
        <tag k="name" v="darmstadtium (Kongresszentrum)"/>
        <tag k="website" v="http://www.darmstadtium.de"/>
 </way>
</osm>
```

BIBLIOGRAPHY

[1] Android. Last accessed: April, 2010.

[2] Google googgles for android.

[3] Layar. http://layar.com/.

[4] Mysql 5.0 reference manual- spatial extensions.

[5] Wikitude. http://www.wikitude.org/.

[6] Ogc kml, April 2008.

[7] Mobile local search report, February 2009.

[8] CellSpotting, 2010.

[9] Flickr: Explore everyone's photos on a map, April 2010.

[10] Google maps, April 2010.

[11] Google mobile, April 2010.

[12] Openstreetmap: The free wiki- world map, June 2010.

[13] SKYHOOK Wireless, 2010.

[14] The Official Bluetooth Technology Info Site, 2010.

[15] Wi-Fi Finder App for Android, 2010.

[16] Wikitude drive - ar navigation system, April 2010.

[17] DNA India, ANI. Microscopic gyroscopes could make mobile phones personal navigation tools. http://www.dnaindia.com/scitech/report_microscopic-gyroscopes-could-make-mobile-phones-personal-navigation-tools_1328417, December 28, 2009.

[18] J. Bittner and P. Wonka. Visibility in Computer Graphics. *Journal of Environmental Planning*, 30:729–756, 2003.

[19] A. J. Brimicombe. GIS - Where are the frontiers now? In *Proceedings GIS 2002, Bahrain*, pages 33–45, 2002.

[20] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory Sensing. In *Proceedings of the Workshop on World-Sensor-Web: Mobile Device Centric Sensor Networks and Applications*, pages 117–134, 2006.

[21] M. Ciurana, F. Barcelo-Arroyo, and S. Cugno. Tracking Mobile Targets indoors using WLAN and Time of Arrival. *Comput. Commun.*, 32(13-14):1552–1558, 2009.

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[23] D. Costantini, A. Reinhardt, P. S. Mogre, and R. Steinmetz. Exploring the Applicability of Participatory Sensing in Emergency Scenarios. In *9th GI/ITG KuVS Fachgespraech, Drahtlose Sensornetze*, 2010.

[24] Linda Deng and Landon P. Cox. LiveCompare: Grocery Bargain Hunting through Participatory Sensing. In *Proceedings of the 10th Workshop on Mobile Computing Systems and Applications*, pages 1–6, 2009.

[25] M.-A. Dru and S. Saada. Location-based mobile services: the essentials. *Alcatel Telecommunications Review*, 2001.

[26] M. J. Egenhofer. Spatial information appliances: A next generation of geographic information systems. *GEOINFO 1999: 1st Brazilian Workshop on GeoInformatics*.

[27] Christer Ericson. *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology)*. Morgan Kaufmann, January 2005.

[28] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pages 29–39, 2008.

[29] J. Fagerberg and A. Malm. Strategic Analysis of the European Mobile LBS Market. Technical report, Berg Insight, 2010.

[30] F. Faisal. Query-by-Pointing: Algorithms and Pointing Error Compensation. Master's thesis, University of Maine, 2003.

[31] N. Freed and N. Borenstein. Multipurpose internet mail extensions (mime) part one: Format of internet message bodies, 1996.

[32] Shravan Gaonkar, Jack Li, Romit Roy Choudhury, Landon Cox, and Al Schmidt. Micro-Blog: Sharing and Querying Content Through Mobile Phones and Social Participation. In *Proceedings of the ACM 6th International Conference on Mobile Systems, Applications, and Services*, 2008.

[33] K. Gardiner and J. D. Carswell. Viewer-Based Directional Querying for Mobile Applications. *Web Information Systems Engineering Workshops, International Conference on*, 0:83–91, 2003.

[34] K. Gardiner, J. Yin, and J. Carswell. EgoViz - A Mobile Based Spatial Interaction System. In *Web and Wireless Geographical Information Systems (W2GIS)*, pages 135–152, 2009.

[35] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[36] Open Geospatial Consortium Inc. Opengis geography markup language (gml) encoding standard. OpenGIS Standard, 8 2007.

[37] Markus Jevring, Robert de Groote, and Cristian Hesselman. Dynamic Optimization of Bluetooth Networks for Indoor Localization. In *Proceedings of the 5th International Conference on Soft Computing as Transdisciplinary Science and Technology*, pages 663–668, 2008.

[38] M. D. Karunanayake, M. E. Cannon, and G. Lachapelle. Analysis of Assistance Data on AGPS Performance. *Measurement Science and Technology*, 18(7):1908, 2007.

[39] J. Kwon, B. Dundar, and P. Varaiya. Hybrid Algorithm for Indoor Positioning using Wireless LAN. In *Proceedings of the 60th IEEE Vehicular Technology Conference*, volume 7, pages 4625–4629, 2004.

[40] H. Laitinen, J. Lahteenmaki, and T. Nordstrom. Database Correlation Method for GSM Location. In *Proceedings of the 53rd IEEE Vehicular Technology Conference*, volume 4, pages 2504–2508, 2001.

[41] B.D.S. Lakmali and D. Dias. Database Correlation for GSM Location in Outdoor & Indoor Environments. In *Proceedings of the 4th International Conference on Information and Automation for Sustainability*, pages 42–47, 2008.

[42] Anthony Lamarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, Jason Tabert, Pauline Powledge, Gaetano Borriello, and Bill Schilit. Place Lab: Device Positioning using Radio Beacons in the Wild. In *Proceedings of the Third International Conference on Pervasive Computing*, pages 116–133, 2005.

[43] Juong-Sik Lee and Baik Hoh. Sell your Experiences: A Market Mechanism based Incentive for Participatory Sensing. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications*, pages 60–68, 2010.

[44] Binghao Li, Andrew Dempster, Chris Rizos, and Joel Barnes. Hybrid Method for Localization Using WLAN. In *Proceedings of SSC 2005 Spatial Intelligence, Innovation and Praxis*, 2005.

[45] Tsung-Nan Lin and Po-Chiang Lin. Performance Comparison of Indoor Positioning Techniques based on Location Fingerprinting in Wireless Networks. In *Proceedings of the International Conference on Wireless Networks, Communications and Mobile Computing*, volume 2, pages 1569–1574, 2005.

[46] David Luebke, Benjamin Watson, Jonathan D. Cohen, Martin Reddy, and Amitabh Varshney. *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002.

[47] N. Maisonneuve, M. Stevens, M. E. Niessen, and L. Steels. NoiseTube: Measuring and mapping noise pollution with mobile phones. In *ITEE*, pages 215–228. Springer, 2009.

[48] S. McLean, S. Macmillan, S. Maus, V. Lesur, A. Thomson, and D. Dater. The us/uk world magnetic model for 2005-2010. Technical report, NOAA National Geophysical Data Center and British Geological Survey Geomagnetism Group, December 2004.

[49] P. S. Mogre, M. Hollick, N. d'Heureuse, H. W. Heckel, T. Krop, and R. Steinmetz. A Graph-based Simple Mobility Model. In *4th Workshop zu Mobilen Ad Hoc Netzen, KiVS 2007*, 2007.

[50] R. Nevatia and K. E. Price. Automatic and Interactive Modeling of Buildings in Urban Environments from Aerial Images. pages III: 525–528, 2002.

[51] B. Preissl, H. Bouwman, and C. W. Steinfield. *E-Life after the Dot Com Bust*, chapter The development of location based services in mobile commerce. Physica Verlag, Springer, 2004.

[52] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The Cricket Location-Support System. In *Proceedings of the 6th Annual International Conference in Mobile Computing and Networking*, pages 32–43, 2000.

[53] Carlo Ratti, Riccardo Maria Pulselli, Sarah Williams, and Dennis Frenchman. Mobile landscapes: using location data from cell phones for urban analysis. *Environment and Planning B: Planning and Design*, 33(5):727–748, 2006.

[54] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The context toolkit: aiding the development of context-enabled applications. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 434–441, New York, NY, USA, 1999. ACM.

[55] A. Schmidt, M. Beigl, and H. W. Gellersen. There is more to context than location. *Computers and Graphics*, 23, 1998.

[56] Johannes Schmitt. *Anpassungsfähige Kontextbestimmung zur Unterstützung von Kommunikationsdiensten*. PhD thesis, Technische Universität Darmstadt, Fachbereich Elektrotechnik und Informationstechnik, Multimedia Kommunikation, Dec 2009.

[57] R. Simon. *Mobilizing the Geospatial Web - A Framework and Conceptual Model for Spatially-aware Mobile Web Applications*. PhD thesis, TU Wien, 2008.

[58] R. Simon, H. Kunczier, and H. Anegg. Towards orientation-aware location based mobile services. *3rd Symposium on LBS and TeleCartography*, 2005.

[59] I. Singh, B. Stearns, and M. Johnson. *Designing enterprise applications with the J2EE platform*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[60] S. Strachan, J. Williamson, and R. Murray-Smith. Show me the way to Monte Carlo: density-based trajectory navigation. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1245–1248, New York, NY, USA, 2007. ACM.

[61] NovAtel: GPS Navogation Systems. Gps position accuracy measures. December 2003.

[62] M. Uhlirz. A market and user view on lbs. In G. Gartner, W. Cartwright, and M. P. Peterson, editors, *Location Based Services and TeleCartography*, Lecture Notes in Geoinformation and Cartography, pages 47–58. Springer Berlin Heidelberg, 2007.

[63] Alex Varshavsky, Eyal de Lara, Jeffrey Hightower, Anthony LaMarca, and Veljo Otsason. GSM Indoor Localization. *Pervasive and Mobile Computing*, 3(6):698–720, 2007.

[64] H. Velayos and G. Karlsson. Limitations of range estimation in wireless LAN. In *Proceedings 1st Workshop on Positioning, Navigation and Communication*, 2004.

[65] Luis von Ahn and Laura Dabbish. Labeling Images with a Computer Game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 319–326, 2004.

[66] Andy Ward, Alan Jones, and Andy Hopper. A New Location Technique for the Active Office. *IEEE Personal Communications*, 4:42–47, 1997.

[67] V. Zeimpekis, G. M. Giaglis, and G. Lekakos. A taxonomy of indoor and outdoor positioning techniques for mobile location services. *ACM SIGecom Exchanges*, 3, 2002.

[68] Jun Zhang, Manli Zhu, Dimitris Papadias, Yufei Tao, and Dik Lun Lee. Location-based spatial queries. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 443–454, New York, NY, USA, 2003. ACM.

ACRONYMS

LoS	Line-of-Sight
FoV	Field-of-View
LBS	Location-based Services
LoD	Level of Detail
OBB	Oriented Bounding Box
POI	Point of Interest
GPS	Global Positioning System
WLAN	Wireless Local Area Network
RSS	Received Signal Strength
LQI	Link Quality Indicator
BSSID	Basic Service Set Identifier
MAC	Media Access Control

PUBLICATIONS

JOURNAL ARTICLES

1. F. Zaid, P. S. Mogre, A. Reinhardt, D. Costantini and Ralf Steinmetz. iVu.KOM: A Framework for Viewer-centric Mobile Location-based Services. Accepted to the *Praxis der Informationsverarbeitung und Kommunikation (PIK), Themenheft Georeferenzierte Daten*, November 2010.

CONFERENCE AND WORKSHOP CONTRIBUTIONS

1. F. Zaid, D. Costantini, P. S. Mogre, A. Reinhardt, J. Schmitt and Ralf Steinmetz. WBroximity: Mobile Participatory Sensing for WLAN- and Bluetooth-based Positioning. In *Proceedings of the 5th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2010)*, October 2010.

2. F. Zaid. Towards Robust and Efficient Visibility-aware Sensor-based Mobile Spatial Interaction. In *IEEE PERCOM 2010 PhD Forum*, March 2010.

3. F. Zaid, J. Schmitt, P. S. Mogre, A. Reinhardt, M. Kropff and R. Steinmetz. Sorting the Wheat from the Chaff: Adaptive Sensor Selection for Context-aware Applications. In *Proceedings of The Second International Workshop on Information Quality and Quality of Service for Pervasive Computing*, March 2010.

4. A. Reinhardt, J. Schmitt, F. Zaid, P. S. Mogre, M. Kropff and R. Steinmetz. Towards Seamless Binding of Context-aware Services to Ubiquitous Information Sources. In *Proceedings of the 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2010)*, February 2010.

5. F. Zaid, R. Berbner and R. Steinmetz. Leveraging the BPEL Event Model to Support QoS-aware Process Execution. In *Kommunikation in Verteilten Systemen 2009 - KiVS 2009: Informatik aktuell*, no. 1431-472X, Springer, January 2009.

CURRICULUM VITÆ

personal details

Name: Farid Zaid

Date and Place of Birth: 2nd June 1977 in Palestine

education

03.2007 - now	Doctoral candidate at the Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt. Topic: *Viewer-centric Mobile Services: A Framework and a Query Model*
2003 - 2006	M.Sc. of Information and Communications Engineering, Technische Universität Darmstadt. Grade: 1,4/1.0 (1.0 being maximum)
1995 - 2001	B.Sc. of Electrical Engineering, The University of Jordan, Faculty of Information Engineering, Jordan. Grade: 3.21/4.0 (4.0 being maximum)
1993 - 1995	General high school education, Palestine. Grade: 98,5 % (100% being maximum). Ranked first country-wide.

work experience

07.2010 - now	Software Engineer, Deutsche Telekom (Products & Innovations), Darmstadt Software Engineer, motionet AG, Düsseldorf
01.2005 - 12. 2005	Working student, Siemens AG, Information and Communications, Munich
06.2002 - 07.2003	GSM Radio Database Engineer, Jordan Mobile Telephone Services, Amman, Jordan

January 7, 2011

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe. Die Arbeit hat bisher noch nicht zur Prüfungszwecken gedient.

Darmstadt, 2010