# Server based Indoor Navigation Using RSSI and Inertial Sensor Information

Manh Kha Hoang, Sarah Schmitz, Christian Drueke,
Dang Hai Tran Vu, Joerg Schmalenstroeer, Reinhold Haeb-Umbach
Department of Communications Engineering
University of Paderborn, Germany
{hoang, tran, schmalen, haeb}@nt.uni-paderborn.de

***Abstract*—In this paper we present a system for indoor navigation based on received signal strength index information of Wireless-LAN access points and relative position estimates. The relative position information is gathered from inertial smartphone sensors using a step detection and an orientation estimate. Our map data is hosted on a server employing a map renderer and a SQL database. The database includes a complete multi-level office building, within which the user can navigate. During navigation, the client retrieves the position estimate from the server, together with the corresponding map tiles to visualize the user's position on the smartphone display.**

***Index Terms*—Smartphone, indoor navigation, map tile, fingerprint**

## I. Introduction

In this paper we describe the client-server infrastructure of an indoor navigation system at the University of Paderborn. At the current project state the system is fully usable within a 4-level building and the expansion to the other buildings on the campus is scheduled.

Indoor navigation and pedestrian dead reckoning has been a research topic for several years. The former mostly employ the fingerprinting method, i.e. the measured received signal strength index (RSSI) of WLAN access points is compared with those from a training phase, where a database of RSSI values and corresponding positions has been compiled. The position whose RSSI value of the training phase most closely match the measured fingerprint is then taken as estimate of the user's location (e.g. [1], [2]). The latter is often based on inertial sensors worn or carried by a user [3]. An overview about wireless sensor network localization techniques can be found in [4].

Although the access point distribution across the university campus is rather inhomogeneous, being dense in the vicinity of lecture halls and sparse in the office areas, the overall Wireless-LAN network coverage should be sufficient for rough navigation purposes. However, in some areas additional information is required, e.g. to correctly distinguish between adjacent levels or rooms and corridors.

For these badly covered areas the step detection based upon the inertial smartphone sensors is a valuable information source. We adopted the ideas described in [5] and [6] to estimate the number of steps and the heading information (see VI for more details).

The combination of RSSI and inertial sensor information with state of the art position estimation approaches mitigates the detrimental effect of the sparse Wireless-LAN network coverage and thus enables a reliable navigation and routing functionality.

Several approaches for estimating the most likely position can be found in recent publications, e.g. support vector machines [1] or $k$-nearest neighbor ($k$-NN) [7], [8]. We used Gaussian distributions to model the signal strengths and adopted our idea described in [9] on how to exploit the information of not observing an expected access point at a given position. A detailed description about the parameter estimation of censored Gaussian distributions and the classification accounting for censored data can be found in [10].

The position information gathered by RSSI and the relative movement estimate provided by the step and heading detection can be fused by different approaches. For example a Kalman filter, a particle filter or one of its derivatives can be applied to get a combined position estimate. Alternatively the positions can be modeled as the states of a hidden Markov model (HMM), the RSSI and step detection information is then interpreted as observations of these states and a Viterbi-decoder can then be used to determine the sequence of positions as proposed in [11]. Our system uses the HMM approach with some modifications.

The paper is organized as follows: At first we give in section II a brief system overview, followed by the description of the map database in section III and the fingerprint database in section IV. Subsequently, the routing is described in section V, the step detection in section VI and the HMM in section VII. In section VIII some experimental results are discussed. Finally, we draw some conclusions and present an outlook.

## II. System Overview

In contrast to other systems, e.g. [2], we use a client-server architecture for supporting indoor positioning and navigation of the smartphone. This approach has several advantages and some disadvantages. Firstly, the map data can be easily kept up-to-date for every system user. Secondly, we can use the RSSI measurements of each user to improve our models and thus the performance for every user. Thirdly, the computationally demanding step of position estimation is handled by a server and not by the smartphone. This enables the usage

of complex algorithms for improved position estimation and filtering techniques independent of the smartphone hardware and its individual performance.

A known disadvantage of a client-server architecture is that routing and navigation requires an ongoing online connection which reduces the battery power.

In Fig. 1 an overview about the system architecture of the indoor navigation system is depicted. The smartphone runs an application called “IndoorNavMap” and the server is subdivided in a “Map Server” and a “Localization & Routing Server”.

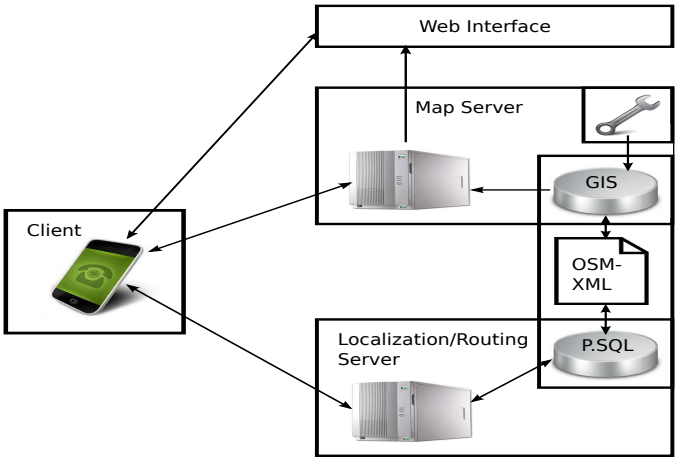


Fig. 1. Client-Server architecture for indoor navigation system

### A. Client Application

The client software “IndoorNavMap” is developed as a Java application on an Android smartphone. Its purpose is the graphically appealing presentation of the current position using the map tiles from the map server and the user route guidance if routing is activated. To this end the application periodically logs the RSSI and communicates with both servers.

In order to make the map displaying user friendly, the map is always rotated such that the user’s movement direction and the map orientation are fitted. For this, the on-board compass of the Android smartphone is utilized.

Fig. 2 shows a snapshot of the smartphone display including a navigation route (green line with arrows) which guides the user from one level downstairs to a room. On the left side on top the icons for zooming in and out are placed as well as the icon for activating the compass based map rotation. In the left corner the icons for changing the floors can be found. Thus it is possible to navigate by hand through the building plans for searching rooms or to get an overview about a proposed route. The icons for making the localization and routing request are placed in the right conner.

### B. Server-based Services

The server-based services are responsible for delivering map tiles, estimating the user position and calculating routes between two positions. The server consists of the following two main components:

Fig. 2. Android application showing map and routing information

- Map server: The map tile server utilizes an Apache webmaster [12] including the custom Apache module mod_tile [13]. It is responsible for serving map tiles, respectively it triggers the rendering of not yet rendered map tiles which are not available in the cache. Clients access this service via a web interface, which automatically handles the client map requests. In order to save computation time and to reduce the latency in the online phase the map tile data on the map server is pre-rendered using Mapnik [14] and some supported software such as Java OpenStreetMap Editor (JOSM) [15] and osm2pgsql [16].
- Localization & Routing server: This server is an own development, which works both in the offline training phase to gather fingerprints and in the online phase to handle the localization and routing requests from multiple parallel clients. A PostgreSQL database is used to store all the necessary information for localization and routing. The advantage of using a PostgreSQL database is that the server directly combines the map information with the recorded fingerprints.

## III. Map Database

As far as we know, there is neither a current map provider capable of providing floor plans of buildings for indoor navigation purposes, nor exists a solution for automatically generating floor plans from construction drawings. Consequently, all developers of indoor navigation systems have to establish their own map data bases. In the following we describe our approach of creating map data fitting into the OSM data.

### A. Map data creation and rendering

We analyzed several map editors and renderers towards their capabilities and requirements for generating map data and finally selected Mapnik and some companion software, since this software combination offered the smallest investment in time and effort. The complete map data rendering tool chain is illustrated in Fig. 3.

The rendering procedure can be summarized as follows: First we use the JOSM tool to download the OSM map data

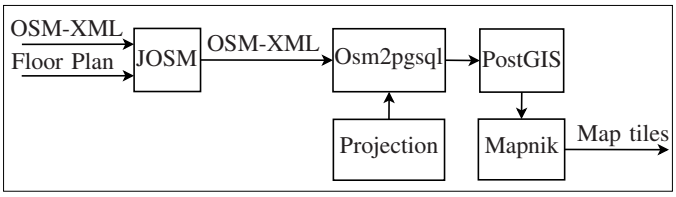


Fig. 3. Rendering tool chain used for map data preparation

of the university area from the OSM server. This gives us the map data for modeling the university campus and the position and size of the buildings in world coordinates, which becomes important if the indoor localization is extended to the campus area (outdoor navigation). Subsequently, we use the building floor plans to extend the downloaded OSM map file towards a map on room level precision. The output is again a OSM format map file, but now it contains a detailed floor plan. In the following we refer to these maps with the term "extended OSM maps". Editing maps is the most time demanding step. Afterward Osm2pgsql and a specified projection method (Spherical Mercator projection) are used to import the data from the extended OSM map file into the PostGIS database. The projection is required to minimize the impact of assuming the earth being a sphere instead of an ellipsoid. Finally, Mapnik renders the map tiles with the defined properties using the PostGIS database.

One problem we had to solve was the combination of the extended map data of different floor plans. If each floor were rendered and stored separately this would result in a database per building level. Additionally, it would increase the complexity of our routing and navigation algorithms, since routing requests might require the path search across multiple databases and the navigation itself requires the handover between floors and buildings. So we decided to create only one database for localization and routing purposes for all floors of all buildings on the campus. To this end we created only one extended OSM map file containing all floor plans and subsequently imported this data into the PostgreSQL database.

### *B. Map data distribution*

A lot of publications focus on the aspect of localization precision and disregard the task of modeling multi-level buildings. However, since our system is intended for real users, we have to handle the complex map data of multiple complete buildings and the whole campus area.

This demanded the creation of a multi-layer map distributor, since common map distribution systems are usually limited to one level (e.g. car navigation). We see here one of the main differences of our system towards other indoor navigation systems. For this purpose, we render the map tiles of each floor of each building separately and store them on the server (offline pre-rendering of map tiles). Note that the original data for the map tile rendering is the single extended OSM map described in the previous section.

In the online navigation phase, each map-request contains information about the longitude and the latitude position of the user, as well as the level within the building and the selected zoom level of the smartphone application. Subsequently, the map server delivers the corresponding map tiles with respect to the communicated information.

## IV. Fingerprint database

Indoor navigation systems usually utilize RSSI from previously recorded data, namely the fingerprints, to estimate the position of the smartphone. The main purpose of the fingerprint database is the storage of fingerprints where a location dependent database organization is obviously the best choice to ease the inference step of estimating user positions.

Therefore, we developed an application to establish tables according to the map data gathered in the map rendering step. In these tables we stored the RSSI data and the corresponding MAC addresses, which are also employed as unique identifiers of the access points. During an initial measurement campaign we filled up the tables of selected positions with at least a few measurements.

Note that during the campaign the fingerprints are already collected by our own Android smartphone application. The measurements can either be directly stored to the database by an online communication with the server or intermediately stored in an XML file on the local storage of the smartphone and afterwards imported to the database. Using the application itself for taking fingerprints reduces the risk of storing fingerprints to wrong positions, since the user has to pinpoint his position on the map during the fingerprint capture.

Collecting fingerprints via the online fingerprint collection mode offers the option to use the already stored fingerprints from the database to detect mismatches between committed RSSI data and the hypothesized position.

Please note that the density of fingerprints is considered carefully, since we are limited to the available infrastructure of our university and the overall area to be covered is quite large. The targeted average fingerprint density is defined to one fingerprint per $3 - 5$ m.

## V. Routing

Our routing system is based on the pre-defined nodes which were inserted in the database during the procedure of map editing and rendering. The routing path, if existing, is displayed within the map on the smartphone's display. It is built by the nodes which lie on the route from the starting location to the expected destination. It is possible to show a route independently of the current user position by adjusting the starting point.

## VI. Inertial Navigation using a Step Detection

The step detection system as depicted in Fig. 4 works as follows. The accelerometer delivers the 3-dimensional acceleration vector $\boldsymbol{a}$. We calculate the absolute acceleration value $\|\boldsymbol{a}\|$ and subtract the gravity constant $g$. To reduce the influence of sensor errors we apply a 5-Hz lowpass filter as proposed in [6]. Subsequently, we estimate the amount of steps

by a new decision rule which we found after examining some experimental trajectories.

In the literature we found two different approaches for detecting steps using accelerometer data. Firstly, in [6] a peak detection with a minimum threshold is used to count the steps. This has the disadvantage that sometimes the maximum is split in 2 local maxima and thus an additional step is detected. The second method we found calculates the zero crossing rate of the accelerometer data (e.g. [5]), which in case of noisy data may also result in additionally detected steps. For our new decision rule we combined the strengths of both approaches by counting the crossing rate of an threshold. Hence, a split maximum is only counted once and since noisy data do not exceed the threshold no additional steps are counted.

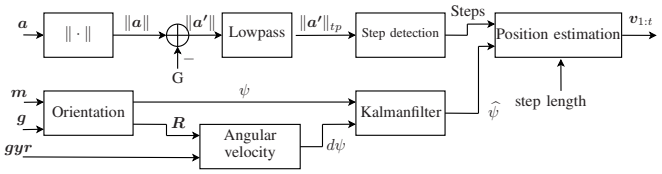


Fig. 4. Step detection and position estimation system overview

The heading estimation uses the magnetometer data $\boldsymbol{m}$ and the gravity information $\boldsymbol{g}$ from the Android operating system to calculate the rotation matrix $\boldsymbol{R}$ and the magnetometer yaw angle $\psi$. The matrix $\boldsymbol{R}$ is required for projecting the gyroscope data $\boldsymbol{gyr}$ into the world coordinate system and subsequently estimating the yaw angle velocity $d\psi$. We tried the approach from [5] for fusing the yaw angle information of the magnetometer with the gyroscope data, however, the amount of heuristic parameters made it difficult to find an optimal, data independent parameter set. Thus we decided to use a Kalman filter for fusing the two information sources towards a common yaw angle estimate $\widehat{\psi}$.

In the position estimate block we combine the information about detected steps with the yaw angle estimates at the corresponding time instance to predict the position of the user. For each smartphone user an average step length has to be estimated in advance.

## VII. Hidden Markov Model And Viterbi Decoder

We combine the step detection and the RSSI information by a HMM, i.e. the hidden states equal the positions were the reference fingerprints where recorded and the step detection and RSSI information are handled as observations emitted by the hidden states. Further, we assume the initial probabilities of being in one position is equal for all states. The position itself is estimated every 1.5 seconds which is the update rate of the smartphone WiFi scans.

Let the hidden state random variable at time instance $t$ be given by $s_t$ and the sequence of RSSIs up to time $t$ be given by $\boldsymbol{o}_{1:t} = [\boldsymbol{o}_1, \dots \boldsymbol{o}_t]$ and the step detection information be given by $\boldsymbol{v}_{1:t} = [\boldsymbol{v}_1, \dots \boldsymbol{v}_t]$. In order to obtain a combined step detection information of the last time period (1.5 s) the steps are accumulated with respect to the corresponding direction information to a 2-dimensional movement vector.

The position estimate can now be performed by finding the most likely state given the sequence of observations. The probability of being in the $j$-th state is given by

$$p(s_t{=}j|\boldsymbol{v}_{1:t}, \boldsymbol{o}_{1:t}) = \frac{p(s_t{=}j, \boldsymbol{v}_{1:t}, \boldsymbol{o}_{1:t})}{p(\boldsymbol{v}_{1:t}, \boldsymbol{o}_{1:t})} \sim p(s_t{=}j, \boldsymbol{v}_{1:t}, \boldsymbol{o}_{1:t}) := \alpha_t(j), \quad (1)$$

where we neglected the denominator term which has no effect on the maximum search. Using Bayes rule it follows

$$\begin{aligned}\alpha_t(j) &= \sum_i p(s_t{=}j, s_{t-1}{=}i, \boldsymbol{v}_{1:t}, \boldsymbol{o}_{1:t}) \\ &= \sum_i p(\boldsymbol{v}_t|s_t{=}j, s_{t-1}{=}i, \boldsymbol{v}_{1:t-1}, \boldsymbol{o}_{1:t-1}, \boldsymbol{o}_t) \\ &\quad \cdot p(\boldsymbol{o}_t|s_t{=}j, s_{t-1}{=}i, \boldsymbol{v}_{1:t-1}, \boldsymbol{o}_{1:t-1}) \\ &\quad \cdot p(s_t{=}j|s_{t-1}{=}i, \boldsymbol{v}_{1:t-1}, \boldsymbol{o}_{1:t-1}) \\ &\quad \cdot p(s_{t-1}{=}i, \boldsymbol{v}_{1:t-1}, \boldsymbol{o}_{1:t-1}) \quad (2)\end{aligned}$$

Applying the properties of the HMM and assuming the step detection and RSSI information of time instance $t$ to be statistically independent from each other and independent from the information of other time instance enables the simplification of eq. (2) to

$$\begin{aligned}\alpha_t(j) &= \sum_i p(\boldsymbol{v}_t|s_t{=}j, s_{t-1}{=}i) \cdot p(\boldsymbol{o}_t|s_t{=}j) \\ &\quad \cdot p(s_t{=}j|s_{t-1}{=}i) \cdot \underbrace{p(s_{t-1}{=}i, \boldsymbol{v}_{1:t-1}, \boldsymbol{o}_{1:t-1})}_{=\alpha_{t-1}(i)} \quad (3)\end{aligned}$$

The terms in eq. (3) can be interpreted as follows. The transition probabilities $p(s_t{=}j|s_{t-1}{=}i)$ are given by the possible paths between two positions, i.e. the transition matrix at entry $(i, j)$ has only a value larger than zero if a path between the positions represented by the $i$-th and $j$-th state exist. The RSSI information is modeled by the term $p(\boldsymbol{o}_t|s_t{=}j)$, which is considered as the likelihood of the RSSI measurement at time instance $t$ for a given position represented by state $s_t$. The distribution $p(\boldsymbol{o}_t|s_t{=}j)$ is modeled as a Gaussian with $p(\boldsymbol{o}_t|s_t{=}j) = \mathcal{N}(\boldsymbol{o}_t; \mu_{j,k}, \sigma^2_{j,k})$

In order to calculate the RSSI likelihood, we need the mean $\mu_{j,k}$ and variance $\sigma^2_{j,k}$ of the RSSI distribution of the $k$-th access point at location $s_t = j$. Note, that in real scenarios for some locations and at some time instances the RSSIs of some APs may fall bellow a minimum threshold of the sensitivity of the WiFi sensor. This results in censored data which has to be accounted for in both parameters of RSSI distribution estimation and online classification (see [10] for details). Accounting for censored data increases the performance of the algorithm significantly.

The movement information gathered from the step detection is regarded with the term $p(\boldsymbol{v}_t|s_t{=}j, s_{t-1}{=}i)$. This probability density function is assumed to follow a Gaussian distribution with a mean vector $\boldsymbol{\mu}_{i,j} = \boldsymbol{l}_j - \boldsymbol{l}_i$ and a diagonal covariance matrix $\Sigma_v$, where $\boldsymbol{l}_j$ is the 2-dimensional location vector

belonging to the $j$-th state.

$$p(\boldsymbol{v}_t|s_t{=}j, s_{t-1}{=}i) = \frac{e^{\left(-\frac{1}{2}(\boldsymbol{v}_t-\boldsymbol{\mu}_{i,j})^T\Sigma_v^{-1}(\boldsymbol{v}_t-\boldsymbol{\mu}_{i,j})\right)}}{\sqrt{(2\pi)^2|\Sigma_v|}} \quad (4)$$

The precision of the approach can be further enhanced by fusing the set of the most likely positions $\boldsymbol{P}$ instead of using only the most probable one. The final location estimate $\hat{\boldsymbol{l}}$ is then obtained by the weighted average:

$$\hat{\boldsymbol{l}} = \frac{1}{\sum\limits_{k\in P} p(s_t = k|\boldsymbol{v}_{1:t}, \boldsymbol{o}_{1:t})} \sum_{k\in P} p(s_t = k|\boldsymbol{v}_{1:t}, \boldsymbol{o}_{1:t})\boldsymbol{l}_k. \quad (5)$$

## VIII. Experimental results

In Figure 5 an example trajectory from a staircase of one of the university buildings is depicted. In dark green the rectangular path of the smartphone user is shown. The red circles indicate the detected steps and the blue line shows the estimated path of the user. Two different errors are visible in the trajectory. Firstly, additional steps are detected after the first turn which causes an elongated path. Secondly, the electrical machines of the three elevators (right side of the figure) obviously affects the magnetic field sensors of the smartphone which results in an orientation error. However, the deviation of the localization remains in a tolerable scope. Furthermore, a map matching approach using information about walls, doors and possible paths would enable a further error reduction.

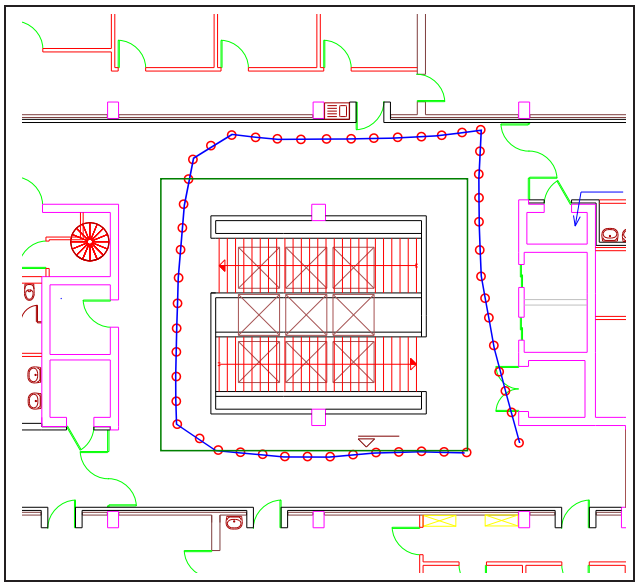

Fig. 5. Trajectory example of the step detection system projected into the map data of construction plans

We have tested our new parametric approach in one floor of one building in the University of Paderborn where our department is located. The results are shown in Fig. 6 in terms of the cumulative distribution function (CDF) of the position error. The CDF is defined as the probability that the positioning error $\epsilon$ is lower than a certain distance $d$:

$$\mathtt{CDF}_\epsilon(d) = P(\epsilon \leq d) \quad d \geq 0. \quad (6)$$

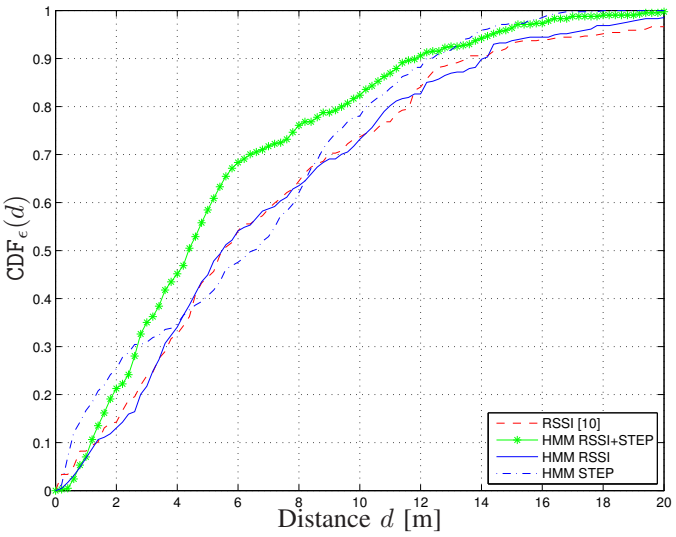


Fig. 6. CDF of the positioning error for different systems.

In our experiments the step detection alone ("HMM Step") performed better than the RSSI information processed by the HMM ("HMM RSSI"). However, if only the step detection information is used additional information about the starting position is required. The combination of both resulted in the green line ("HMM RSSI+Step"). This approach mostly outperformed our proposal from [10] ("RSSI [10]").

## IX. Conclusions

In this paper we have presented an indoor navigation system for the University of Paderborn as developed by our department. It is a flexible client-server architecture which offers navigation services even to low-cost smartphones. We explained how the database generation, using the OpenStreetMap data as a starting point, can be conducted.

Additionally our step detection system for dead reckoning was briefly described. It enables the smartphone application to continue the route guidance even if no network access is available.

We are currently extending our system towards online learning capabilities by utilizing the results of the step detection system and the moving direction estimation in combination with the RSSI information. It will enable our system to fill up the fingerprint database itself from user committed RSSI positioning requests, and consequently, improve the positioning accuracy.

## Acknowledgment

We wish to thank Sebastian Kowelek for providing ideas and software code to the indoor navigation project.

## References

[1] Carlos Figuera, José Luis Rojo-Álvarez, Mark Wilby, Inmaculada Mora-Jiménez, and Antonio J. Caamaño, "Advanced support vector machines for 802.11 indoor location," *Signal Processing*, vol. 92, no. 9, pp. 2126–2136, 2012.

[2] Eladio Martin, Oriol Vinyals, Gerald Friedland, and Ruzena Bajcsy, "Precise indoor localization using smart phones," in *Proceedings of the international conference on Multimedia*, New York, NY, USA, 2010, MM '10, pp. 787–790, ACM.

[3] Kerem Altun and Billur Barshan, "Pedestrian dead reckoning employing simultaneous activity recognition cues," *Measurement Science and Technology*, vol. 23, no. 2, pp. 025103, 2012.

[4] Guoqiang Mao, Barış Fidan, and Brian D. O. Anderson, "Wireless sensor network localization techniques," *Comput. Netw.*, vol. 51, no. 10, pp. 2529–2553, July 2007.

[5] Wonho Kang, Seongho Nam, and Youngnam Han, "Improved Heading Estimation for Smartphone-based Indoor Positioning Systems," in *Proceedings IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2012, PIMRC '12.

[6] Martin Mladenov and Michael Mock, "A step counter service for Java-enabled devices using a built-in accelerometer," in *Proceedings of the 1st International Workshop on Context-Aware Middleware and Services: affiliated with the 4th International Conference on Communication System Software and Middleware (COMSWARE 2009)*, New York, NY, USA, 2009, CAMS '09, pp. 1–5, ACM.

[7] Halabi Hasbullah Azat Rozyyev and Fazli Subhan, "Combined K-Nearest Neighbors and Fuzzy Logic Indoor Localization Technique for Wireless Sensor Network," *Research Journal of Information Technology*, vol. 4, no. 4, pp. 155–165, 2012.

[8] Paramvir Bahl, Venkata N. Padmanabhan, and Anand Balachandran, "Enhancements to the RADAR User Location and Tracking System," Tech. Rep., Microsoft Research, 2000.

[9] R. Haeb-Umbach and S. Peschke, "A Novel Similarity Measure for Positioning Cellular Phones by a Comparison With a Database of Signal Power Levels," *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 1, pp. 368–372, jan. 2007.

[10] K. Hoang and R. Haeb-Umbach, "Parameter Estimation and Classification of Censored Gaussian Data with Application to WIFI Indoor Positioning," *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, May 2013.

[11] Jochen Seitz, Thorsten Vaupel, Steffen Meyer, Javier Guttierrez Boronat, and Jörn Thielecke, "A Hidden Markov Model for pedestrian navigation.," *Proc. Workshop on Positioning, Navigation and Communication (WPNC'10)*, pp. 120–127, 2010.

[12] "Apache web-master server," http://httpd.apache.org/.

[13] "Custom Apache module Mod_tile for serving the map tiles," http://wiki.openstreetmap.org/wiki/Mod_tile.

[14] "Mapnik: map tile renderer," http://mapnik.org/.

[15] "JOSM: Java OpenStreetMap Editor," http://josm.openstreetmap.de/.

[16] "Osm2pgsql: tool for converting OSM data to a PostgreSQL database," https://github.com/openstreetmap/osm2pgsql.