FootPath: Accurate Map-based Indoor Navigation Using Smartphones

Jó Ágila Bitsch Link, Paul Smith, Nicolai Viol, and Klaus Wehrle
RWTH Aachen University/ComSys, Aachen, Germany.
Email: {jo.bitsch,paul.smith,nicolai.viol,klaus.wehrle}@rwth-aachen.de

***Abstract*—We present *FootPath*, a self-contained, map-based indoor navigation system. Using only the accelerometer and the compass readily available in modern smartphones we accurately localize a user on her route, and provide her with turn-by-turn instructions to her destination. To compensate for inaccuracies in step detection and heading estimation, we match the detected steps onto the expected route using sequence alignment algorithms from the field of bioinformatics. As our solution integrates well with OpenStreetMap, it allows painless and cost-efficient collaborative deployment, without the need for additional infrastructure.**

I. Introduction

While navigation systems for outdoor environments are readily available, navigation within buildings still poses a challenge. The main reason for this lies in the difficulty to obtain accurate position information in an easy to set-up way with minimal infrastructure and to create indoor maps.

Our approach to this problem is twofold: (1) We use simple step detection and step heading estimation. (2) We match detected steps onto the expected route from the source to the destination using sequence alignment algorithms. Instead of a more general localization problem, we solve the localization problem on a specified route. This allows us to compensate for inaccuracies and give the user accurate turn-by-turn directions.

To allow easy incremental deployment of our system, we integrate our system with OpenStreetMap [1], which already has rudimentary indoor support [2]. GPS, Pseudolites, UWB, WiFi access points, and RFID is avoided, making the system useful for protected environments like historical buildings and archaeological sites as well as hospitals, where additional RF gear might interfere with medical equipment.

A. Contributions

The main contributions of this paper are:

1) **Infrastructureless indoor navigation:** We use simple step detection and step heading detection, which we then map onto a route using sequence alignment algorithms. Additional infrastructure, like GPS, Pseudolites, UWB, WiFi access points, and RFID can be avoided.
2) **Localization on a route:** We know the route, the user intends to take. Using this knowledge, we reduce inaccuracy at corners opposed to further accumulating errors. Path matching is precise enough to allow for accurate indoor turn-by-turn directions.
3) **Easy incremental deployment:** Deploying the system for a new building simply consists of entering the floor plan into OpenStreetMap.

This research was funded in part by the DFG Cluster of Excellence on Ultra High-Speed Mob. Inf. and Comm. (UMIC), DFG grant DFG EXC 89.

978-1-4577-1804-5/11$26.00 © 2011 IEEE

II. Related Work

The localization method is the core component of an accurate navigation system. There exists a multitude of methods, but they can be categorized into the following three categories: Localization based on (1) lateration (or angulation), localization based on (2) fingerprinting, and localization based on (3) dead reckoning. However, in general, none of these methods make assumptions about the actual route of a user.

A. Lateration

In general, lateration (or angulation) techniques make use of the distance (or angle) of a user to a set of beacons, calculating the relative position with respect to those beacons. The accuracy of the distance (or angle) measurement directly influences the localization accuracy. The most prominent example used for outdoor localization is GPS. To realize a precise lateration based indoor localization, we need a dense infrastructure of beacons, called GPS pseudolites. Systems based on GPS pseudolites, e.g. [3], allow for a precision of $0.01m$ and better. However, they require very carefully placed transmitters and an exact calibration, making wide public deployment impractical and unfeasible for the time being.

B. Fingerprinting

Similarly, localization systems based on WiFi fingerprints, such as [4], [5], collect the identities and signal strengths of the WiFi access points in the vicinity at various points in the covered area. This calibration—war driving—is time consuming, and easily becomes invalidated when physical conditions change, e.g. the number of people in the vicinity or new office equipment, thereby requiring new measurements to keep the database up to date. While efforts to reduce the required fingerprint positions are promising, they still depend on exact 3D-models and are rather labor-intensive to set up. However, a systematic drawback remains: There needs to be an adequate number of access points in the vicinity. This may be problematic in protected environments like historic buildings, archaeological sites or hospitals. In comparison, our approach

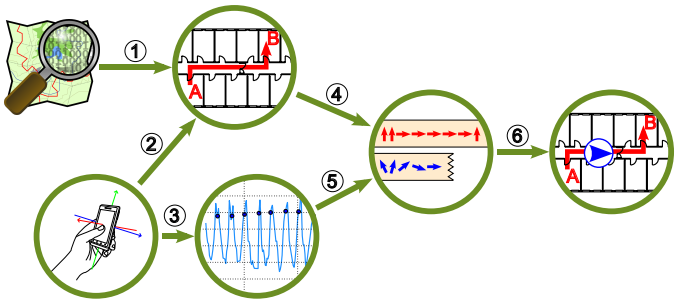

Fig. 1. Flow of information during navigation. (1) The application obtains map material from OSM. (2) The user selects her current position and her destination. The phone calculates the best route. (3) The mobile phone detects steps and directions. (4) The route is transformed into expected steps. (5) The detected steps are mapped onto the expected steps. (6) The user gets feedback about her position and her next way-points.

neither depends on war driving nor on additional RF infrastructure. Our OpenStreetMap integration makes incremental deployment possible and painless.

C. Dead Reckoning

Dead reckoning approaches, such as [6], are based on detecting steps and step headings, integrating over them to estimate the current user position. Adaptive Kalman filters and activity based map matching—e.g. resetting the user position to the nearest elevator, if elevator like patterns are detected—improve the position estimate. However, errors accumulate quickly. Our approach can reset these errors by matching the steps using sequence alignment. Thereby, it actually benefits from turns, commonly found in indoor environments.

Constandache et al. [7] estimate outdoor user location with a precision of up to $11m$ using only a compass, an accelerometer, and AGPS for the initial position. Detected steps are matched onto the currently closest path derived from Google Maps, returning the best match as the user's position. In case of mismatch, AGPS resets the position. Our approach differs in that we neither need AGPS, nor are we restricted to outdoor navigation. Also, our path matching through sequence alignment algorithms, see Section III-C2, is more robust by handling source to destination routes in their entirety, instead of per segment.

III. System Design

Figure 1 presents an overview of our system. We obtain map material from OpenStreetMap, this allows easy updating and incremental deployment on a global scale, see Section III-A. After the user selects her route, the accelerometer and compass of the user's phone are used to detect steps and step headings, see Section III-B. We then match these steps onto the map using a first fit and a sequence matching scheme, see Section III-C. Finally, we present the estimated position back to the user, together with turn-by-turn directions towards the destination, see Figure 14 for screen shots of our current prototype.

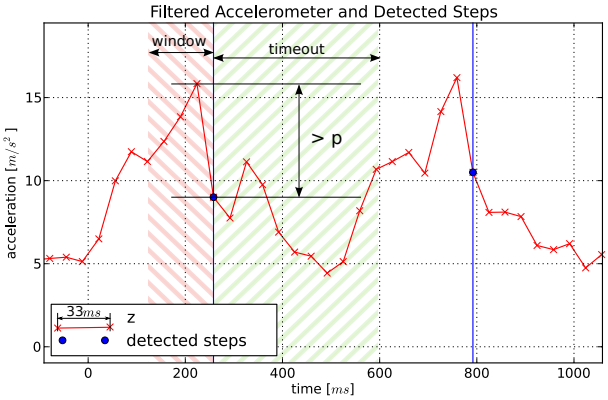

Fig. 2. Step detection with *FootPath*. A step is detected if there is a difference of at least $p = 2\frac{m}{s^2}$ on the low pass filtered z axis of the accelerometer. The difference has to occur during a window w of 5 consecutive readings, or $165ms$. After each detected step a timeout $t = 333ms$ is used to avoid false detection. The user can calibrate p and t to improve performance.

A. Generating Maps

OpenStreetMap [1] is an effort to create and distribute free geographic data, such as street maps, but also indoor maps of public buildings, albeit indoor support is still rudimentary [2]. OpenStreetMap allows wiki-style editing, thereby enabling everyone to contribute easily.

Map data from OpenStreetMap can be accessed as an XML structure consisting of nodes, ways, areas, and relations, which can be annotated with arbitrary key value pairs. Indoor nodes can be annotated using a combination of the following keywords:

- **indoor=yes** marks an object as being indoors.
- **level=*** designates the associated level or floor of an object.
- **wheelchair=yes** indicates accessibility by wheelchairs.
- **highway=steps** denotes steps with the additional keyword **stepcount=*** providing its length.
- **highway=elevator** labels elevators, connecting different floors.
- **highway=door** specifies a node to be a door. **building=entrance** as a special case denotes the entrance door to a building.
- **name=*** is used to give an object a common name.

The popularity of OpenStreetMap allows us to make use of a variety of tools—e.g. JOSM [8]—to create and extend maps incrementally. The OpenStreetMap community has already mapped the vicinity of our building in great detail, easing our task to integrate our indoor maps, which we derived from floor plans with outdoor footpaths and streets.

Editing paths lying on top of each other, i.e., in different floors, is still cumbersome. We alleviated this by creating one distinct map file per floor, and annotating nodes to be merged with a node in another layer with the keyword **merge_id=***. This can easily be mitigated by extending JOSM with a better indoor support plugin.

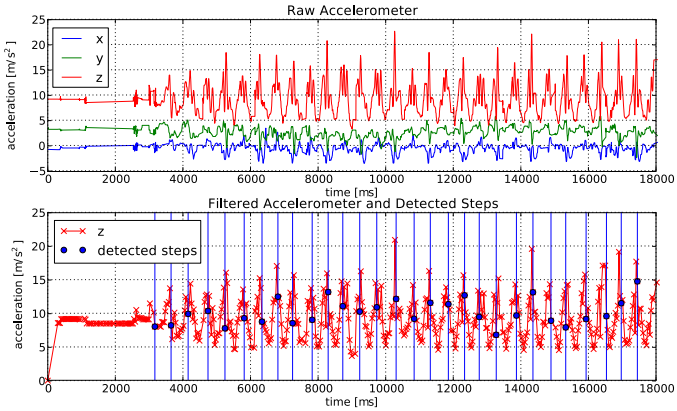

Fig. 3. Exemplary accelerometer raw data and detected steps. The upper plot depicts the raw data recorded from the phone's sensor, while a user first stands still for 2 seconds and then starts walking. The values display characteristic jiggling pattern. In the lower plot, we perform step detection on the low pass filtered z axis values.

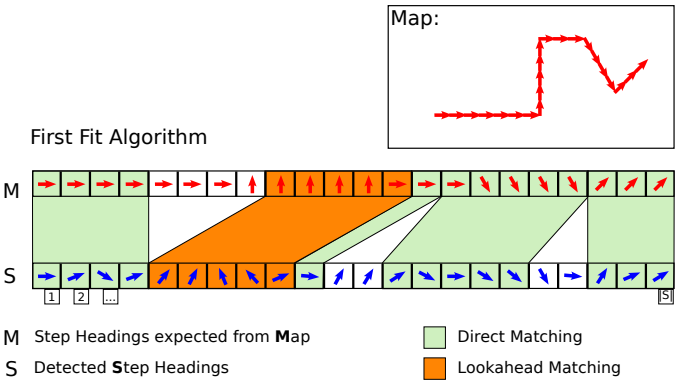

Fig. 4. Matching sequences of detected steps onto sequences of expected steps using *First Fit*. We detect direction changes and find the next possible match in a lookahead. We see that $S(5)$ does not match $M(5)$ and is thus collected for further evaluation. The same happens with the following detected step headings $S(6)$ to $S(9)$. In this example, *lookahead matching mode* is triggered after $k = 4$ consecutive unmatched steps and the new location is $M(13)$ after step $S(9)$. The matching segment from steps $S(5)$ to $S(9)$ is located from $M(9)$ to $M(13)$. Steps $S(11), S(12)$ are handled as errors and the algorithm resumes with *direct matching mode*. Steps $S(18), S(19)$ still match to the previous edge, thus *First Fit* waits for the user to resume with the next edge.

B. Step Detection

Modern smartphones are typically equipped with an accelerometer and a compass. We make use of this fact and directly use them for our step detection and step heading estimation. The accelerometer values display a characteristic regular pattern, see Figure 2. Therefore, we can detect a step, by matching the values to a sharp drop in the acceleration, attributed to the jiggling of the phone in the hand of the user while she is balancing out her steps. To further improve detection, we initially apply a low pass filter.

Formally, we detect a step whenever the acceleration value falls by at least $p = 2\frac{m}{s^2}$ within a window w of 5 consecutive samples, or $165ms$. Additionally, we define a timeout $t = 333ms$ within which no new step is detected. The parameters p, t, and the low pass filter parameter l can be calibrated to further improve step detection performance on a per user basis.

Figure 3 shows an exemplary data set where a user first stands still for $2s$ and then starts walking, while holding her phone screen facing up in front of her in her hand. We repeated this experiment with 15 users and found the parameters to be robust against body heights and walking styles.

As soon as a step is detected, the phone also records the current azimuth from the compass and passes the detected step and step heading to the path matching algorithms. For a correct step heading it is importand for the user to hold the device, such that the left-right axis (x-axis) of the device is perpendicular to the walking direction. Minor deviations can be compensated by our algorithms.

C. Path Matching

Upon detection of a step, we trigger path matching. We propose two strategies for matching detected steps onto expected steps from a map: (1) *First Fit* and (2) *Best Fit*, see Figure 4 and 5. Both algorithms respond with a position on the route, which is in turn used for user feedback, see Figure 14. In the following, we describe the path matching strategies.

1) First Fit: This algorithm—similar to CompAcc [7]—makes use of the assumption that the user's detected step heading corresponds directly to the direction of the expected edge. Upon each detected step and step heading, we try to match this heading to the direction of the current edge and move along this edge. If this is the case the algorithm is working in *direct matching mode*. If the step heading and the direction of the current edge do not match for $k = 5$ consecutive detected steps, the algorithm switches to a *lookahead matching mode*, to find a position further ahead on the path.

A heading α_i and an edge direction β_j are defined to be *matching*, if the angle between them $\measuredangle(\alpha_i, \beta_j) \leq 42°$. In *direct matching mode* the position is moved along the route by increments of the initial step length estimation l, with each directly matching step.

As we progress along an edge, there are two cases:

(1) The step length l was overestimated, and the real user position is not yet at the end of this edge. This is due to the algorithm progressing faster along an edge than the user does in reality. In this case the location is corrected by the algorithm by waiting at the beginning of a new edge as long as the detected headings match the previous edge, on which the user is still located. If step headings match the new edge, we continue with *direct matching mode* along the new edge. If the step headings match neither the current edge nor the previous edge after $k = 5$ consecutive steps the algorithm switches into *lookahead matching mode*.

(2) However, if the step length l was underestimated, we obtain values not matching the current edge, because the user already walks into a different direction. This can also occur if we have consecutive headings which do not directly match the current edge, but remain in *direct matching mode* due to a directly matching step after $j \leq k$ steps. In this case, values

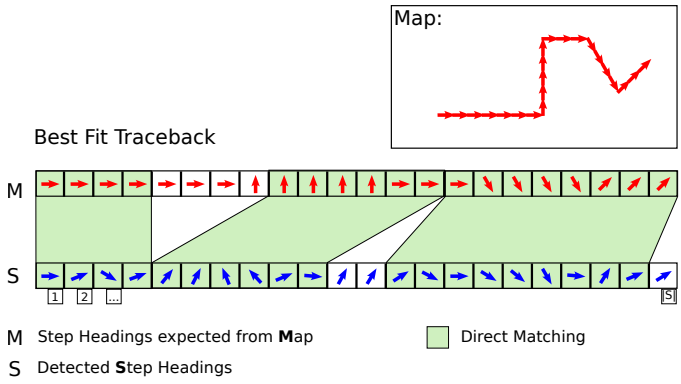

Fig. 5. Matching sequences of detected steps onto sequences of expected steps using *Best Fit*. We use sequence alignment to find the best match between the sequences. Unmatched parts correspond to overestimated and underestimated step lengths.

corresponding to the next edge are regarded as errors and the *lookahead matching mode* corrects the position to a location along the next edge.

If steps do not match expected steps they are collected for further processing in *lookahead matching mode*. If, within k steps, a matching heading α_i is detected, *direct matching* operation resumes. However, if no such step heading is detected, the algorithm will try to find a maximum amount of at least $k = 5$ steps consecutively matching to a segment on the path.

We remain in *lookahead matching mode* until we find a segment on the path which machtes to at least $k = 5$ consecutive steps of unmachted steps, going backwards from the latest unmatched step heading. If no such segment is found, more steps are collected to repeat this process.

A matching segment is found by finding the first edge along the path with direction β_j which matches the latest step heading α_i, then move the end of the segment along this edge for all previous headings $\alpha_{<i}$ which match this edge, and then establish the start of the segment by matching more previous unmatched steps backwards along the path to the edge directions $\beta_{<j}$. The location returned by this lookahead is the end of the matching segment which matches the latest unmatched steps.

2) Best Fit: Inspired from sequence alignment algorithms [9], widely used in bioinformatics and related fields [10]–[12], we model the matching of detected steps onto expected steps extracted from a map as a string matching problem. The matching process is formalized as a dynamic programming problem, where we punish mismatches with a penalty. The best match is therefore the one with the smallest penalty. Underestimated and overestimated step lengths are modeled as gaps.

We define M as an array of all step headings $M(i) : 1 \leq i \leq |M|$ on the route subdivided into virtual steps according to the information in the given map. Then, we define S as the string of all detected step headings $S(j) : 1 \leq j \leq |S|$.

As discussed in Section III-B, each step is associated with an azimuth. Comparing the azimuth of a detected step with the azimuth of a point on the expected route, we define a $\texttt{score}$ depending on the angle between them. The closer the

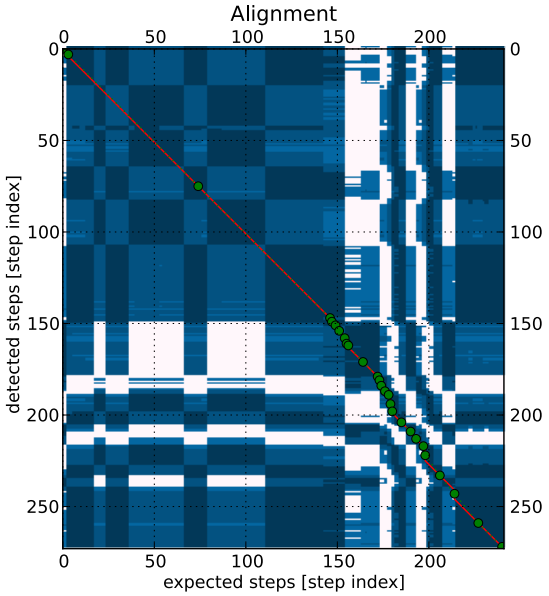

Fig. 6. Recurrence plot of detected steps and exptected steps. The darker parts signify regions where the alignment between the detected real step and the expected step from the map is very close. The lighter regions denote regions, where such an alignment is not possible. The line with the markers from the upper left corner, to the lower right corner, represents the best overall matching for this particular run.

two directions match, the smaller this value:

$$\texttt{score}(\alpha,\beta) = \begin{cases} 0.0 & \text{if } \sphericalangle(\alpha,\beta) \leq 45° \\ 1.0 & \text{if } 45° < \sphericalangle(\alpha,\beta) \leq 90° \\ 2.0 & \text{if } 90° < \sphericalangle(\alpha,\beta) \leq 120° \\ 10.0 & \text{else} \end{cases}$$

As an example, Figure 6 shows in a recurrence plot, how well detected steps line up with expected steps, using the raw data from an indoor experiments and Figure 12 depicts expected compass directions versus measured bearings on that path, see Section IV-B.

Further following the dynamic programming approach, we define a matrix D, with $D(i,j) = d_{i,j} : 1 \leq i \leq |M|, 1 \leq j \leq |S|$. We initialize this matrix with $D(0,0) = 0$, $D(i,0) = \infty : 1 \leq i \leq |M|$, and $D(0,j) = \infty : 1 \leq j \leq |S|$. The other elements are calculated using the following construction:

$$\begin{aligned} D(i,j) = \min\{&D(i-1,j-1) + \texttt{score}(M(i),S(j)), \\ &D(i-1,j) + \texttt{score}(M(i),S(j-1)) + 1.5, \\ &D(i,j-1) + \texttt{score}(M(i-1),S(j)) + 1.5\} \end{aligned}$$

Figures 7 and 11 show a visual representation of this matrix in the background. From this definition, we derive the expected position along the path—map step pos_j—after j detected steps, using this formula:

$$pos_j = \underset{i:1\leq i\leq |M|}{\operatorname{argmin}} \left(D(i,j)\right)$$

As we are only interested in the current location pos_j, the calculation of column $D(\_,j)$ only depends on the previous

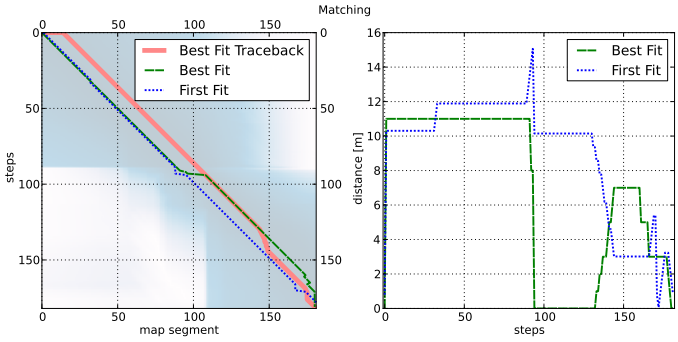

Fig. 7. Left-hand side: Displayed are the detected steps and their matched position on the path. *Best Fit* and *First Fit* are our matching algorithms. *Best Fit Traceback* is the best trace run on the matrix D, see Section III-C2, which is visualized in the background. Right-hand side: The absolute differences of our matching algorithms to the *Best Fit Traceback* match.

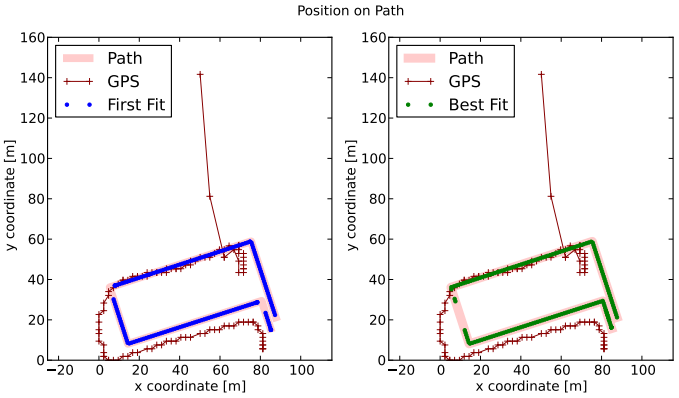

Fig. 8. During the experiment each of the 15 users walked along a predefined path. In both figures we see the detected steps and their position corresponding to the matching algorithms. GPS was tracked for comparison and is displayed in both figures.

column $D(\_, (j-1))$. This makes the implementation fast and very space efficient.

IV. Evaluation

We conducted three sets of experiments to show the feasibility of our approach: (1) An outdoor experiment on a nearby parking lot, in which we compare localization accuracy of *First Fit* and *Best Fit* to GPS, and (2) an indoor experiment, in which users use our system for navigating through our university buildings. This second experiment also includes staircases around elevators and doors, further showing the robustness of *FootPath*. (3) To showcase the ease of map creation, we also created indoor maps for a trade fair.

A. Comparison with GPS

An outdoor experiment, to be able to compare with GPS data, was conducted with 15 test users to demonstrate the functionality of *FootPath*. In the experiment, the same device was used for each run. During the run the sensor data, as well as the GPS location was traced.

An exemplary visualization can be seen in Figure 8 and Figure 9. Our results show that our estimate of the step length from the body height is not very accurate. Still, *FootPath* is able to reset the location if it finds a better match of the user's position. The average accuracy of a detected step, defined as the distance to the *Best Fit Traceback*, are $11.16m$ for *First Fit* and $8.90m$ for *Best Fit* in our 15 test runs.

Considering *First Fit*, we see that if the directions can not be matched along the path directly, due to a varying and/or incorrect initial step size, it jumps ahead after finding at least four matching steps along the path. If the step size is too large it will wait for the user to catch up and continue when the user changes the direction according to the path.

A problem is that this can lead to erroneous progress along the path if there are faulty directions read which match to the path. At this point we have to consider the trade-off to detect the correct position during the lookahead phase and the amount of steps we have to wait to obtain a new position.

In comparison, *Best Fit* matches each direction onto the path to where it received the smallest penalty. Thus, it is possible that *Best Fit* lags behind on an edge if the assumed step length is smaller than the real one. It will respond to values which correspond to the following edge with locations further along the same edge, or remain at the same position. As soon as the penalty for a location ahead on the path becomes smaller, it will jump ahead. If the directions match better to a previous position on the path it will fall back and continue from there.

B. Indoor Path

To evaluate the performance of our algorithms for indoor environments, we selected a path containing staircases, several doors and metal structures to have a distribution of error sources along the path. For example, a user will have to abandon walking directly along the path when opening doors. We chose such a route to compare the design of our matching algorithms and show that our proposition of providing an accurate indoor navigation system is justified.

Looking at the distribution of the estimated locations for *Best Fit* and *First Fit* in Figure 10, we see that *Best Fit* deals better with inaccuaracies compared to *First Fit*. The latter algorithm looses accuarcy especially around the elevator, as the metal structures found in this staircase disturb the compass. Here, the detected compass values are considered as noise as the compass values change rapidly within few steps, whilst walking around the elevator in the center of the staircase.

In Figure 10 we can see how *First Fit* uses the *lookahead matching mode* to reestablish a location around corners of the path. This works as intended, as corners are anchor points which produce distinguishable compass headings which are detectable by both algorithms. With more corners, i.e. with a more complex route through an environment, our algorithms have a higher accuracy, as there are more points to reset location errors. Still, narrow staircases around metal structures, pose a problem to *First Fit*, as the changes in the compass directions happen too fast and are considered noise. This leads to fewer established locations during navigation.

In this experiment, we have an average accuary of $1.6m$ for *Best Fit* and $5.6m$ for *First Fit*. Due to the design of *Best Fit*, which consists of constantly updating the penalty

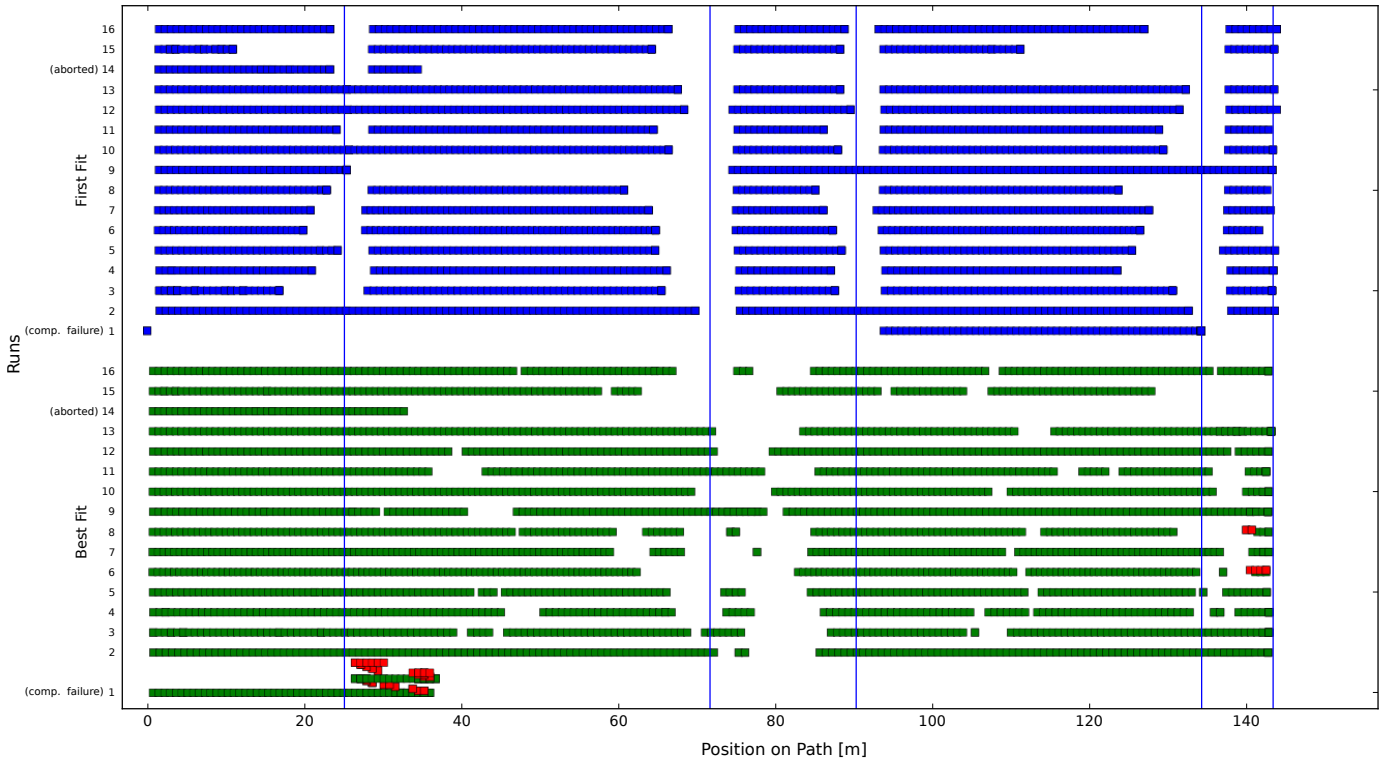

Fig. 9. The distribution of detected lcoations along the path for each run. We see steps matched consecutively as segments forming a horizontal connected line along the graph. Each square is a step. Red slightly elevated squares depict segments starting with a backwards jump using *Best Fit*. We see each algorithm jumping forwards along the path from the gaps between two segments. *First Fit* does this with the use of a lookahead, and *Best Fit* by choosing the position with the smallest penalty. The compass failed during run 1, and during run 14 the application was accidentally closed.

for each location along the path, this algorithm performs better and more stable than *First Fit*. The impact of noisy compass values is tolerated and considered against all positions along the whole route. Thus, this algorithm can recover a location estimate after noisy compass headings, where *First Fit* would regard these noisy compass values as errors, and does not use these later along the path. Walking around the elevator, produces more distinctive penalties, and thus a higher accuracy, see Figure 6 and 11.

C. Map Creation for a Trade Fair

To support our claim of easy incremental deployability of *FootPath*, we looked into possible use cases. While public buildings, such as universities, are obvious use cases for indoor navigation, we also consider trade fairs and conference venues. Visitors attend these venues typically without extensive pre-knowledge about the locality in cities they are unfamiliar with. Also, the exact setup between specific trade shows changes, even when they take place at the same exhibition center, making easy and fast deployability all the more important.

As a proof of concept, we created a map for a trade fair in a nearby city, see Figure 13. The trade fair organizer provided reference floor plans covering over $20\,000 m^2$ on his website. With those plans available, it took a single person less then two hours to enter the plan into OpenStreetMap, including the necessary labels by which users can look for an exhibitor. User experience was good, but we were not able to run repeated

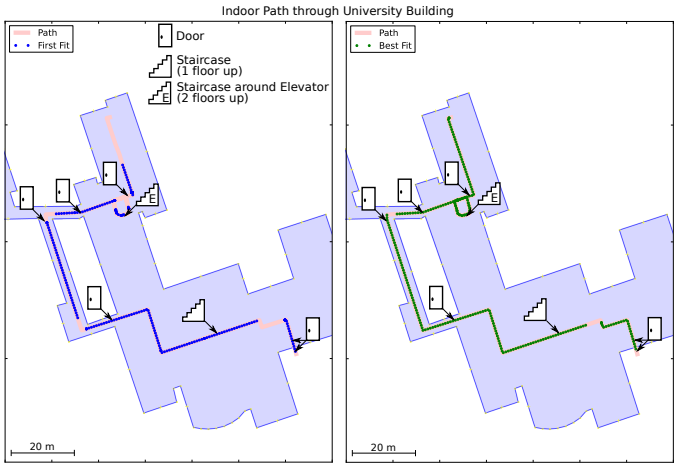

Fig. 10. In this experiment, the user had to navigate our university building using *FootPath*. We chose the path to pass by two staircases and several doors to show the robustness of our approach. While both *First Fit* and *Best Fit* work reasonably well, *Best Fit* deals with inaccuracies inherently arising in human mobility more robustly. For both algorithms the calculated positions are plotted along the path.

experiments for logistical reasons. Therefore, we have only isolated measurements (not shown).

Thus we provide a fast deployable system which is ready to be deployed depending only on the given map. Collaboratively working on map creation can further scale this process such that we obtain a fast deployment for even larger or complex environments.

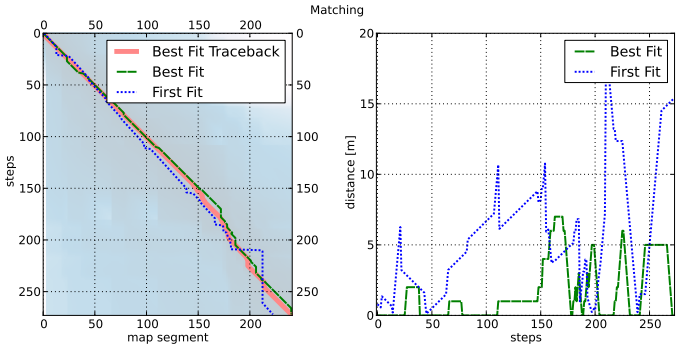

Fig. 11. Left-hand side: Displayed are the detected steps and their matched position on the path. Right-hand side: The absolute differences of our matching algorithms to the *Best Fit Traceback* match. The average location error for *Best Fit* is $1.6m$ and $5.6m$ for *First Fit*.

D. Discussion

The previous results show *FootPath* performs accurate enough to help the user find her way in indoor environments. The closest related approach, *CompAcc*, depends heavily on GPS for resynchronization and its difficult to create indoor maps for use with it.

PDR on the other hand also works without the use of GPS, but depends on escalators or elevators to reset an erroneous position. Dead Reckoning also suffers from high sensitivity with respect to compass inaccuracies and step length variations, which are very common in indoor navigation.

We summarize these characterizations in the following table with respect to the aspects indoor deployability, outdoor deployability, the need for additional infrastructure, initial setup or maintenance, and how they reset errors and the source of the map data:

Feature	*FootPath*	*CompAcc*	*PDR*	*GPS*
Indooor	✓	– (GPS needed)	✓	–
Outdoor	✓	✓	✓	✓
No Infrastructure	✓	– (GPS needed)	✓	–
No Initial Setup	✓	✓	✓	–
No Maintenance	✓	✓	✓	–
Error Resetting	✓ (route)	✓ (route,GPS)	✓ (elevators, stairs)	–
Map basis	✓ (OpenStreetMap)	– (proprietary)	– (proprietary)	– (proprietary)

Comparing *FootPath* to the other localization systems, like GPS Pseudolites, WiFi fingerprinting and Google Maps, we see that they either depend on war-driving or significant calibration effort. Furthermore, their use of additional infrastructure can limit their area of applicability.

For instance in hospitals, the use of mobile wireless equipment is often forbidden, as it might interfere with medical equipment. The prohibitive cost of GPS Pesudolites in terms of equipment and calibration also make it feasible only for very specific use cases. *FootPath* on the other hand can easily deployed campus wide, in hospitals or on archeological sites, where additional hardware may be damaging. We summarize their characterizations in the following table:

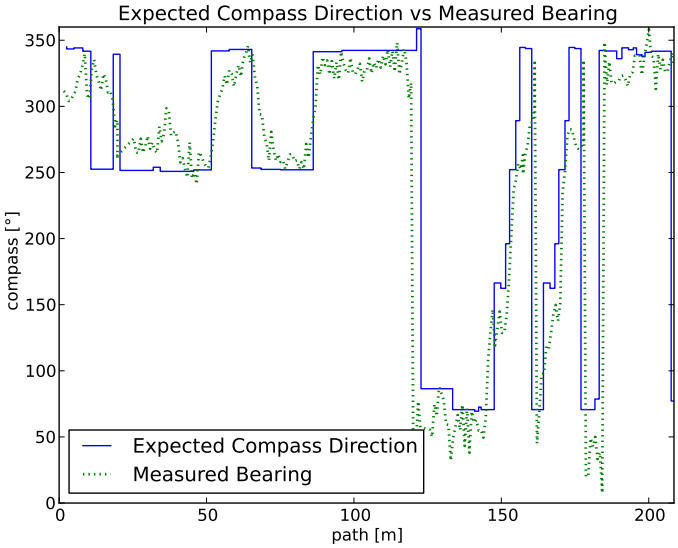

Fig. 12. Visualization of the compass direction of the path and the measured compass bearing along the path. Despite the flawed step heading we can see the resemblance of both graphs. This is exploited by both *First Fit* and *Best Fit*. The two consecutive staircases going around an elevator can be seen starting at about $150m$ to $180m$, with the expected and detected compass values moving clockwise.

Feature	Pseudolites	WiFi F.print.	Google Maps
Indooor	✓	✓	–
Outdoor	– (GPS availiable)	✓	✓
No Infrastructure	–	–	– (WiFi/GSM/GPS)
No Initial Setup	– (calibration)	– (war-driving)	– (war-driving)
No Maintenance	– (calibration)	– (war-driving)	– (war-driving)
Error Resetting	–	–	–
Map basis	– (proprietary)	– (proprietary)	– (proprietary)

To the best of our knowledge, none of the discussed approaches properly address the task of creating indoor maps in a generic way. Making use of OpenStreetMap and its associated tools allowed us to create maps over the course of two hours for a complete trade fair, making *FootPath* almost instantly deployable.

We implemented our system designed for infrastructureless indoor navigation as an Android application.[1] Android is one of the most popular mobile phone operating systems and allows easy prototyping, while the devices typically feature all sensors needed for our algorithms. Figure 14 shows screenshots of our prototype. However, porting *FootPath* to other operating systems running on mobile devices, such as the iPhone running iOS or other devices running Windows Phone, is easily possible.

Finally, we currently do not deal with the user deviating from the displayed route. The algorithms will always try to map the user onto the calculated route. This leads to undefined behavior, if a user wanders off. In the future, we

[1]Source code available at https://github.com/COMSYS/FootPath

Fig. 13. We created a map to navigate in an indoor environment consisting of two trade fair halls at an exhibition center covering 20 000m^2. The process of aligning reference pictures, drawing the paths and labeling nodes to be selected by the user, took less than two hours.

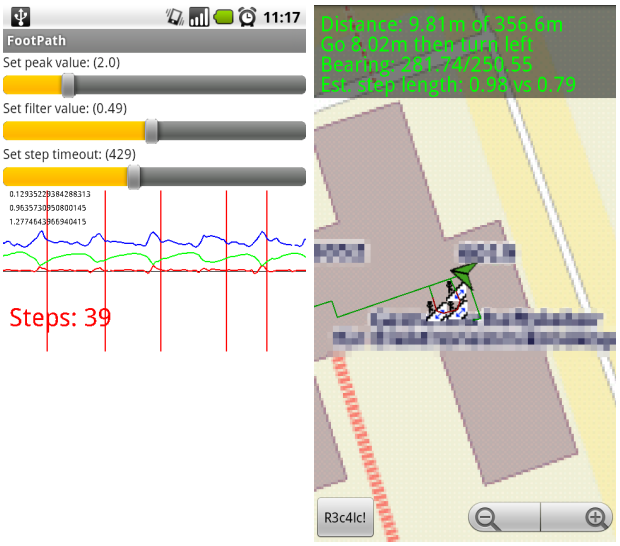

Fig. 14. Screen shots of the calibration screen and during navigation. While calibration parameters allow to tune step detection parameters, path matching itself is robust enough, even without tweaking. The current Android prototype during navigation displays background tiles from OpenStreetMap with an overlay of the calculated path and the user position. In addition we show internal status information, like the exact measured bearing, and the currently assumed bearing.

plan to address this issue by matching the user to several possible paths at the same time and opportunistically switching between those using bioinformatics inspired multi-alignment. Integrating other localization methods, our approach will further enhance robustness of our approach.

V. Conclusions

We presented *FootPath*, a self-contained, map-based indoor navigation system and demonstrated its feasibility in terms of indoor localization accuracy and incremental global deployability. We demonstrated this in three experiments: an outdoor experiment to measure accuracy, an indoor run to show robustness against typical indoor hazards, and a trade fair scenario.

Furthermore, we showed the feasibility of sequence alignment for localization on a route in our *Best Fit* algorithm. This scheme is easy to implement and has only minimal memory requirements, making it appropriate for smart phones and other highly embedded devices.

References

[1] OpenStreetMap community, "OpenStreetMap, The Free Wiki World Map," March 2011. [Online]. Available: http://www.openstreetmap.org/

[2] ——, "Indoor Mapping – OpenStreetMap Wiki," March 2011. [Online]. Available: http://wiki.openstreetmap.org/wiki/Indoor_Mapping

[3] C. Kee, D. Yun, H. Jun, B. Parkinson, S. Pullen, and T. Lagenstein, "Centimeter-accuracy indoor navigation using GPS-like pseudolites," *GPS WORLD*, vol. 12, no. 11, pp. 14–23, 2001. [Online]. Available: http://www.tik.ee.ethz.ch/~beutel/projects/picopositioning/gps_pseudolites.pdf

[4] P. Prasithsangaree, P. Krishnamurthy, and P. Chrysanthis, "On indoor position location with wireless LANs," in *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*, vol. 2. IEEE, 2002, pp. 720–724. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1047316&tag=1

[5] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm, "Accuracy characterization for metropolitan-scale wi-fi localization," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, ser. MobiSys '05. New York, NY, USA: ACM, 2005, pp. 233–245. [Online]. Available: http://portal.acm.org/citation.cfm?id=1067195

[6] D. Gusenbauer, C. Isert, and J. Krösche, "Self-contained indoor positioning on off-the-shelf mobile devices," in *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, 2010, pp. 1 –9. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5646681

[7] I. Constandache, R. Choudhury, and I. Rhee, "Towards mobile phone localization without war-driving," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1 –9. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5462058

[8] I. Scholz and OpenStreetMap community, "Java OpenStreetMap Editor," March 2011. [Online]. Available: http://josm.openstreetmap.de/

[9] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *J. ACM*, vol. 21, pp. 168–173, January 1974. [Online]. Available: http://portal.acm.org/citation.cfm?doid=321796.321811

[10] A. Abbot and A. Tsay, "Sequence analysis and optimal matching methods in sociology," *Sociological Methods & Research*, vol. 29, no. 1, pp. 3–33, 2000. [Online]. Available: http://smr.sagepub.com/content/29/1/3.abstract

[11] R. Barzilay and L. Lee, "Bootstrapping lexical choice via multiple-sequence alignment," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, ser. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 164–171. [Online]. Available: http://dx.doi.org/10.3115/1118693.1118715

[12] A. Prinzie and D. V. den Poel, "Incorporating sequential information into traditional classification models by using an element/position-sensitive sam," *Decision Support Systems*, vol. 42, no. 2, pp. 508 – 526, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/B6V8S-4G0YT6D-2/2/35c336180f44a2938df02b39c8417909