FootPath: Accurate Map-based Indoor Navigation Using Smartphones

Jó Ágila Bitsch Link, Paul Smith, and Klaus Wehrle
RWTH Aachen University/ComSys, Aachen, Germany.
Email: {jo.bitsch,paul.smith,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.**

Index Terms—**indoor navigation; step detection; mobile phones**

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

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.

II. Related Work

A multitude of indoor navigation solutions have been proposed. They can be categorized in several categories, however, in general, they make no assumptions about the route of a user. (1) Systems based on GPS pseudolites, e.g. [3], allow for a precision of $0.01m$ and better, they require very carefully placed transmitters and an exact calibration, making wide public deployment impractical and unfeasible for the time being.

Similarly, (2) 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 neither depends on war driving nor on additional RF infrastructure. Our OpenStreetMap integration makes incremental deployment possible and painless.

Finally, (3) 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.

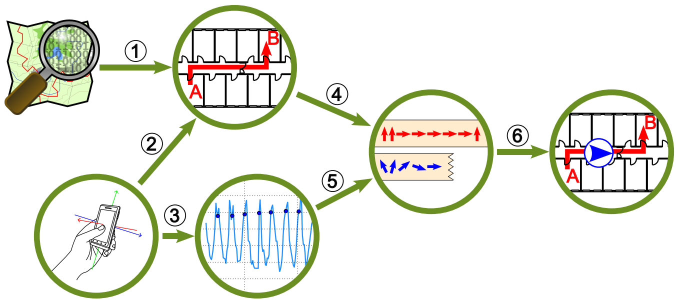


Fig. 1. Flow of information during navigation. (1) The application obtains map material from OSM. (2) The user selects her current position and her destination. The phone calculates the best route. (3) The mobile phone detects steps and directions. (4) The route is transformed into expected steps. (5)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

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 5 for screen shots of our current prototype.

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.

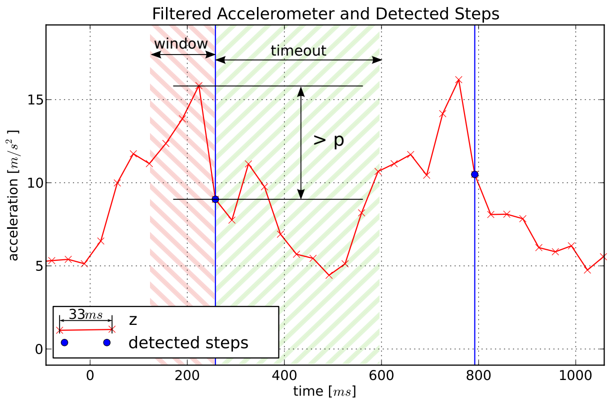


Fig. 2. Step detection with *FootPath*. A step is detected if there is a difference of at least $p = 2\frac{m}{s^2}$ on the low pass filtered z axis of the accelerometer. The difference has to occur during a window w of 5 consecutive readings, or $165ms$. After each detected step a timeout $t = 333ms$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

- **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.

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.

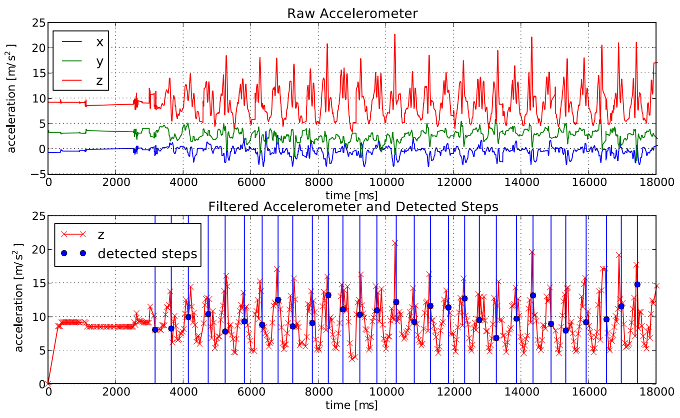


Fig. 3. Exemplary accelerometer raw data and detected steps. The upper plot depicts the raw data recorded from the phone's sensor, while a user first stands still for 2 seconds and then starts walking. The values display characteristic jiggling pattern. In the lower plot, we perform step detection on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

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. The best fit corresponds to a position on the route, which is in turn used for user feedback, see Figure 5. 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, this is *direct matching mode*. If the heading and the direction do not match for $k = 5$ consecutive detected steps, we try to find a new position on the path, using a *lookahead matching mode*.

A heading α_i and an edge direction β_j are *directly matching*, if the angle between them $\sphericalangle(\alpha_i, \beta_j) \leq 42°$. The position is then updated by step length l along the edge of the route.

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 detected by more detected steps in the same direction. (2) However, if the step length was underestimated, we obtain values not matching the current edge, because the user already walks into a different direction. These values are regarded as erors and the *lookahead matching mode* repositions the location to 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 steps consecutively matching to a segment on the path. We find these matches by searching further along the path trying to find a matching edge for the newest unmatched step headings. If a match is found, we proceed by backtracking along the previously unmatched steps trying to match them backwards to the matching and its preceding edges. If there is more than one consecutive match of a given minimum length, the longest

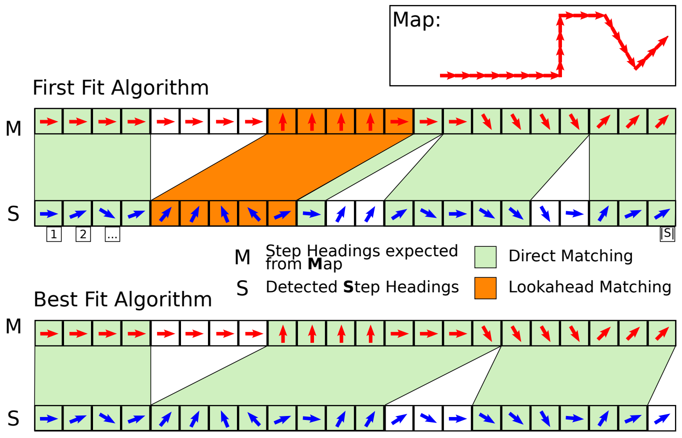


Fig. 4. Matching sequences of detected steps onto sequences of expected steps. *First Fit* detects direction changes and finds the next possible match in a lookahead. *Best Fit* uses sequence alignment to find the best match between the sequences. Unmatched parts correspond to overestimated and underestimated step lengths.

match is chosen. If we do not find a match along the path, we restart the matching process upon the next detected step.

2) Best Fit: By adapting sequence alignment algorithms [9] from the field of bioinformatics, we align the detected steps with the expected steps extracted from a map. 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 $S(j) : 1 \leq j \leq |S|$. Additionally, we define a matrix D, with $D(i,j) = d_{i,j} : 1 \leq i \leq |M|, 1 \leq j \leq |S|$. We initialize the 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:

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

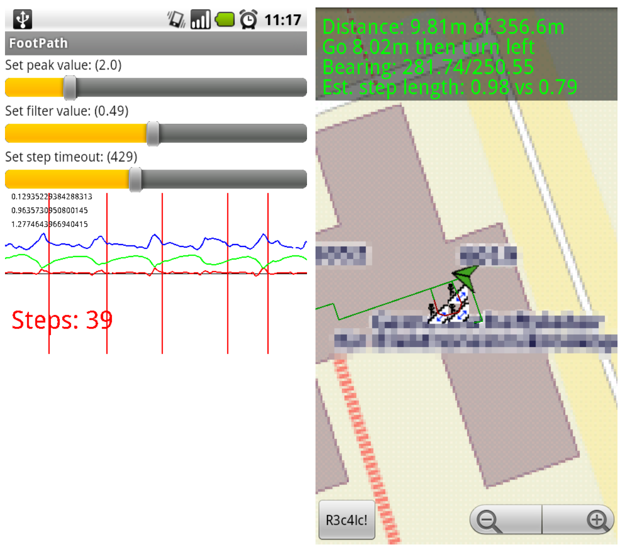
score returns the added penalties depending on how far the two given directions from the step and the considered location differ:

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

The expected position after j detected steps is therefore at map step pos_j with:

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

As we are only interested in the current location pos_j, the calculation of column $D(\_,j)$ only depends on the previous column $D(\_,(j-1))$. This makes the implementation very space efficient.



Set peak value: (2.0)
Set filter value: (0.49)
Set step timeout: (429)
0.12935229384288313
0.9635730950800145
1.2774643966940415
Steps: 39

Distance: 9.81m of 356.6m
Go 8.02m then turn left
Bearing: 281.74/250.55
Est. step length: 0.98 vs 0.79

Fig. 5. Screen shots of the calibration screen and during navigation.

IV. Evaluation

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 7. 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.

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. Furthermore, we showed the feasibility of sequence alignment for localization on a route.

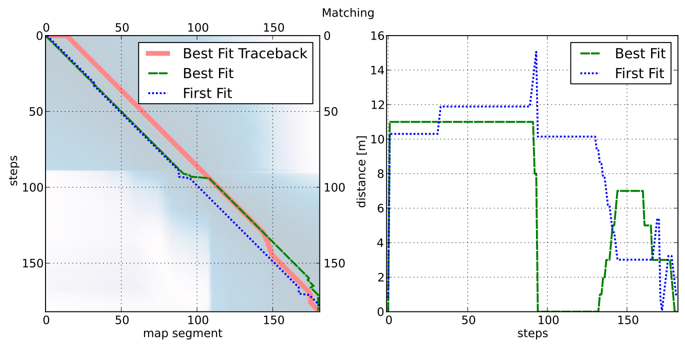


Fig. 6. Left-hand side: Displayed are the detected steps and their matched position on the path. Best Fit and First Fit are our matching algorithms. Best Fit Traceback is the best trace run on the matrix D, see Section III-C2, which is visualized in the background. Right-hand side: The absolute differences of our matching algorithms to the Best Fit Traceback match.

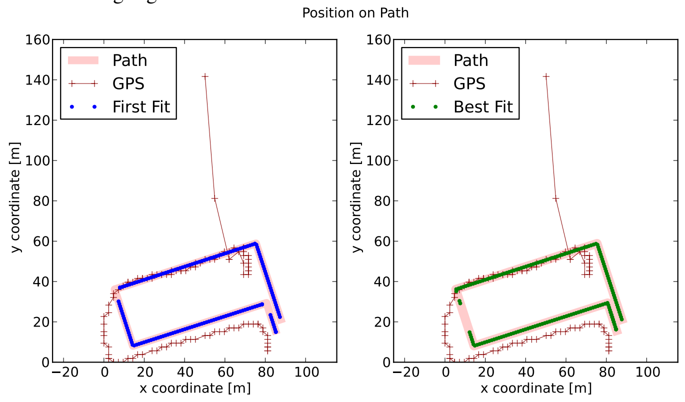


Fig. 7. During the experiment each user walked along a predefined path. In both figures we see the detected steps and their position corresponding to the matching algorithms. GPS was tracked for comparison and is displayed in both figures.

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.

[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.

[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.

[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.

[7] I. Constandache, R. Choudhury, and I. Rhee, "Towards mobile phone localization without war-driving," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1 –9.

[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.