Improving Pedestrian Dead Reckoning using Likely Paths and Backtracking for Mobile Devices

Fabian Hölzke, Johann-P. Wolff, Christian Haubelt
Institute of Applied Microelectronics and CE
University of Rostock
Rostock, Germany
Email: fabian.hoelzke2@uni-rostock.de

Abstract—**Pedestrian Dead Reckoning is a method estimating a persons path from a known starting point based on length and direction of all performed steps. Measuring these parameters, e.g. using inertial sensors, introduces small errors that accumulate quickly into large distance errors. Knowledge of a building's model may reduce these errors as it can be used to keep the estimated position from moving through walls and onto likely paths. Common indoor localization approaches like particle filters track, verify and re-sample several hundred positioning estimates with each user step, resulting in a comparably high computational load. In this paper, we use backtracking to improve an existing localization system tracking a single localization estimate using a foot-mounted inertial sensor and a smartphone. We show how backtracking a single localization estimate can improve the accuracy of indoor positioning systems and discuss restrictions and disadvantages of this approach. Our quantitative results show a reduction of the positioning error by up to 75% and an average endpoint accuracy of 1.91% of the travelled distance with an average computation time of 256.7μs on a 2014 Motorola G2 smartphone.**

I. Introduction

The utility of smart home applications is greatly enhanced by location based services (LBS). Today, LBS are most commonly provided by tracking a users position via GPS, providing meter accurate localization when moving outdoors. Indoors however, GPS fails to deliver an adequate estimation of user positions. In this paper, we propose a method for indoor positioning that works especially well in large buildings. One possible application could be retirement homes: Here the indoor positioning enables services like tracking of residents and staff and routing staff to the residents current position. Residents may also be assisted by the automatic activation of smart home features if they enter their living spaces or be relieved of turning off appliances and devices when leaving. There exist approaches for indoor localization based on the buildings existing infrastructure like WiFi fingerprinting [1], [2], [3] or based on additional infrastructure like Bluetooth beacons [4], [5]. These approaches require extensive setup and maintenance of a signal strength database and, in the case of beacons, monetary investment in hardware. These beacons however may suffer from unclear long term usability because of a lack of standardization regarding the transmitted information [6], [7].

An alternative to these approaches is Pedestrian Dead Reckoning (PDR) [8], [9], [10]. It uses wearable hardware to count steps and estimate the step length and orientation. The users position is then estimated by stringing together consecutive steps. As these measurements are never perfect, small errors add up with each step, degrading the long-term accuracy of this approach. To limit the accuracy degradation of PDR, additional positioning technologies may be used to fuse multiple independent localization sources [11], [12]. However, infrastructure based methods may not be cost effective because of the required setup and maintenance. Another approach is using a building map. For example, measured steps that violate a plausibility check (i.e. crossing walls, leaving a known path) are unlikely and may be corrected to a more plausible nearby position [13], [14].

Particle filter based PDR methods use this approach. A cloud of particles represents possible user positions and each particle is propagated semi-randomly with each step. The cloud of particles then models the uncertainty of the localization after each step. Particles are weighted according a given set of rules. For example, a particle that crosses a wall represents a highly unlikely user trajectory and may be discarded and re-sampled. This discards unlikely user trajectories and leaves only plausible positions. The users position is finally calculated as a function of the remaining weighted particles [15], [16]. A drawback of this method is the comparably high computational complexity due to the propagation and validation of several hundred particles per user step, limiting its practicality on mobile and wearable devices.

The approach presented in this paper is based on the work described in [17]. Here, the users position is directly propagated by the measured step length and direction. If a measured step is found as invalid (i.e. crossing a wall) the users position is corrected to the nearest plausible location based on the last step measurement. However, if this corrected position is not the true user position, the localization is actively degraded by the correction method. In extreme cases, this leads the user trajectory to a dead end, rendering any further user tracking useless. In this paper, we present a backtracking method to reverse faulty correction decisions and reexamine past measurements of the user trajectory to find the most plausible current user location.

The paper is structured as follows: In Section II, related work is explored and the previous work, on which the presented method is based, is described and discussed. Following in Section III, a method for backtracking of past navigation decisions is presented and explained in detail. Section IV then describes the experiment used to validate the method, results and implications are given in Section V. In Section VI, conclusions are given and future work is outlined.

II. Related work

In addition to the related work already presented in Section I, we want to describe a similar approach in more detail: The use of backtracking in conjunction with a particle filter is presented in [15]. In this approach, every particle inherits a predefined number of preceding positions. If a particle crosses a wall, it is re-sampled and re-evaluated a few times. If it is still invalid, it and its predecessors are deleted. Afterwards, the past user positions are computed again and the deleted particles are re-sampled at the new position. The authors use 2000 particles, save 20 to 60 past particle positions and report a 50$\%$ decrease in positioning error compared to traditional particle filtering. The average error at the end position is given as 1.50m with an escape plan map compared to 2.01m without backtracking. The propagation and repeated validation of 2000 particles suggests a high computational complexity of this approach. Additionally, multiplying past positions and particle count, the management of 40.000 to 120.000 particle positions is required. This may limit the methods practicality on mobile devices. Finding a valid trajectory may be a matter of chance with particle filters: Although highly likely, it is not guaranteed that a particles trajectory resembles the users path through a building. Errors or losses in measurement, for example, may introduce large deviations between the sampled and true user trajectory. A less probabilistic approach may lead to more reliability in indoor positioning.

A. Correcting PDR using likely and unlikely paths

In [17], we describe a method of step wise PDR correction by map data. The correction algorithms include Orientation Angle Correction (OAC), redirection when intersecting walls (Correction using Wall Information - CWI) and forcing the user position on known paths (Correction using Path Information - CPI). These methods chose likely map features to correct the users position, e.g. the nearest door if a wall is intersected or the nearest path. A schematic showing errors that are corrected by CWI is shown in Figure 1: Intersections with walls in shallow angles are reflected back into the room, intersections with acute angles are either lead around walls or, if no opening is nearby, set in front of the intersection. If a minor fraction of the step vector is intersecting the wall in an acute angle, the user position is also set in front of the wall.

The results presented in [17] show that the combination of the correction methods work well regarding short routes with a length of up to 100 m and low complexity. On longer paths however, the accuracy is on average severely degraded. A detailed analysis of the data reveals that a fraction of sampled user trajectories show a significant deviation from ground truth at specific points of the route. There are 5 and 11 such trajectories on Routes 3 and 4 respectively. In these instances

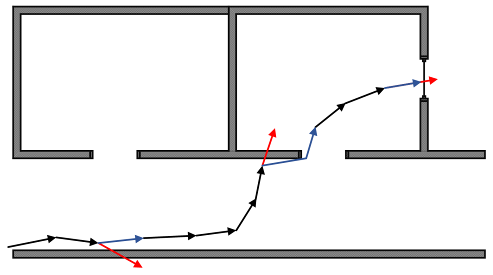
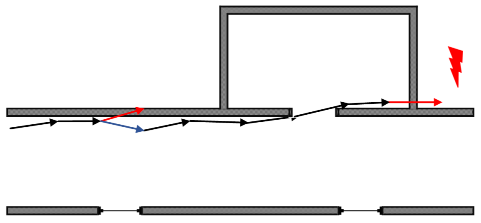


Figure 1: Reflecting, redirecting and shortening steps by CWI

the correction algorithms chose the most likely but wrong map feature to correct a sampled step - subsequent user steps are severely misplaced.

The benefit of combining CWI and CPI is not obvious from the accuracy plots. However, as illustrated in Figures 2a and 2b, we argue that a combination of both methods decreases the possibility intersections with walls due to positioning drift and therefore reduces the chance of erroneously entering spaces leading to dead ends. Disallowing the user trajectory to drift through walls reduces the chance of erroneously selecting paths that may not be located in the users vicinity. For this reason we use a combination of both methods in conjunction with backtracking.


(a) Using CWI without CPI may lead to positioning errors

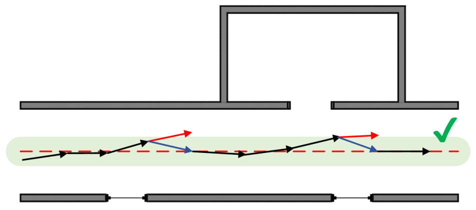

(b) Combining CWI and CPI reduces CWI-specific errors

Figure 2: Benefit of combining CWI and CPI

CWI is also identified as a source of occasional erroneous corrections: Occasionally, the wall-opening chosen by CWI to correct the user position is incorrect, leading to large positing errors or dead ends. This highlights the need to backtrack past correction choices to find the true user position. Our backtracking method is presented in the following sections.

III. Method

The backtracking method is structured in three parts: Trajectory validation, finding alternative trajectories and selecting an alternative. The following sections III-A, III-B and III-C describe these steps in detail. The general backtracking control

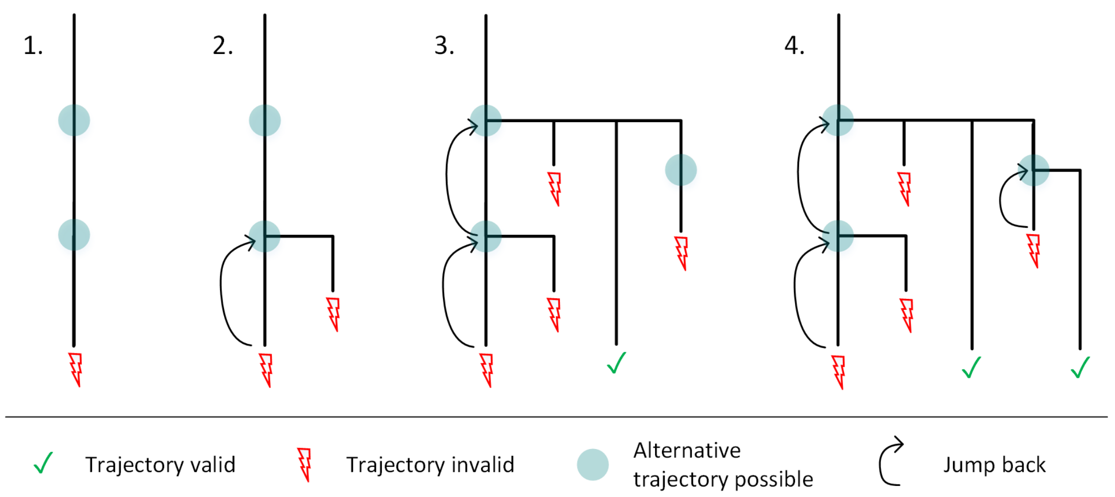


Figure 3: Exemplary backtracking of an invalid trajectory in four steps

flow is illustrated by an example in Figure 3. Here, a user trajectory through a building is shown as a black line from top to bottom or start to current position respectively. Alternative paths are indicated by blue dots and branches to the right. The resulting structure is similar to a search tree. The search for the correct trajectory may be understood as a depth-first search in this tree of possible trajectories.

The example of Figure 3 illustrates the search for a a valid user position in four steps:

1) A trajectory is recognized as invalid. The last step with a possible alternative correction choice is searched (e.g. another nearby door after an acute wall-intersection). This step is referred to below as the candidate step.
2) At this position, alternative corrections and the resulting trajectories are determined and validated with raw data of the following steps. If a valid alternative is found, backtracking stops.
3) No valid alternative is found yet. The candidate step with possible alternative corrections is searched further up the list of past steps. Here, three alternatives are found. One trajectory is valid and two are invalid. One invalid trajectory however contains another candidate step with other possible correction choices, thus allowing further backtracking.
4) The invalid sub trajectory is backtracked and another valid trajectory is found.

Two valid trajectories are found in this example. The procedure to choose the trajectory that is the most plausible is described in Section III-C. The following Section III-A describes the trajectory validation method.

The corresponding pseudocode is listed in Algorithm III.1. The procedure receives four parameters: $unplausibleSteps$, $stepList$, $level$ and $holdOffCnt$. $unplausibleSteps$ is a list of invalidated steps, that need to be re-evaluated with alternative corrections. $stepList$ is the list of saved steps that may be used to backtrack further. $level$ is the current level of recursion. This parameter limits the extent of the search within sub-trees of alternative trajectories. $holdOffCnt$ sets the minimum number of steps that are re-evaluated. This parameter ensures a meaningful minimum number of steps are re-evaluated. The call FINDBTSTEP(...) returns the candidate step with alternative corrections. The call BTWALL(...) returns the best scoring alternative trajectory starting from the candidate step. This procedure will itself call BACKTRACK(...) to recursively search sub-trees as shown in step 4 of Figure 3. The deepest allowed recursion level is specified by the global parameter $maxBtLevel$. If an alternative is found or the $stepList$ is traversed too far back, backtracking stops. If no alternative is found, the next candidate step is searched.

Algorithm III.1 Backtracking: Search for alternative trajectories

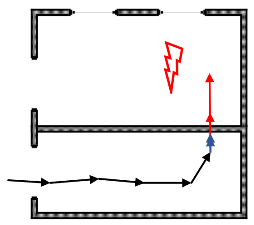
```
procedure BACKTRACK($unplausibleSteps, stepList,$
        $level, holdOffCnt$)
    $btCorrSteps \leftarrow \emptyset$
    $backtracking \leftarrow true$
    $btEntryIdx = stepList.indexOf(unplausibleSteps[0])$
    $btStepIdx \leftarrow$ FINDBTSTEP($stepList, btEntryIdx$)
    while $backtracking \wedge level < maxBTLevel$ do
        if $btStepIdx \geq holdoffCnt$ then
                                            ▷ Found a candidate step
                        ▷ Search the best alternative trajectory at this point
            $btCorrSteps \leftarrow$ BTWALL($stepList, btStepIdx, level$)
        end if
        if $btCorrSteps \neq \emptyset$ then
                              ▷ Alternative found, exit loop and recursion
            $backtracking \leftarrow false$
        else
                                            ▷ Find next candidate
            $nextBtIdx \leftarrow$ FINDBTSTEP($stepList, btStepIdx - 1$)
            if $nextBtIdx \geq holdoffCnt \wedge$
                    $(btEntryIdx - nextBtIdx) < maxBtSteps$ then
                $btStepIdx \leftarrow nextBtIdx$
                $backtracking \leftarrow true$
            else
                                            ▷ No more candidates
                $backtracking \leftarrow false$
            end if
        end if
    end while
    return $btCorrSteps$
end procedure
```

A. *Validating the current trajectory*

In order to provide useful indoor positioning it is essential to recognize trajectories that are unlikely early on and with low computational effort. We propose a trajectory validation for every new step vector with two easy to compute rules:

- Two corrected steps are consecutively intersecting walls.
- The difference in orientation between a new step vector from raw data and its correction is greater than 90°.

Intuition dictates that a trajectory that crosses a wall is invalid. However, the correction method CWI relies on intersections with walls and already handles these instances. Instead, multiple consecutive intersections with a wall indicate a need to backtrack. Here two scenarios that are anticipated: In the first scenario, there is no passage nearby: The first and second step are set in front of the wall. This case indicates a dead end: Past corrections need to be evaluated to find an alternative trajectory. An example of this is illustrated by Figure 4a. In the second scenario, there is a passage nearby: The first step is intersecting the wall with a small portion of the step length and is shortened by CWI to end just before the wall. The second step is intersecting the wall again and is then lead around the wall. In this case the first step is needlessly set in front of the wall instead of guiding the step around the wall. Although the introduced positioning error is minor, the accuracy may still be increased by reversing the erroneous first correction. This case is illustrated by Figure 4b.


(a) Multiple wall-intersections indicating a dead end

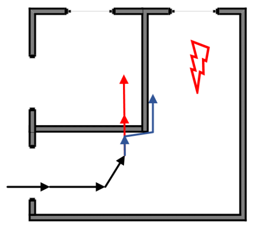
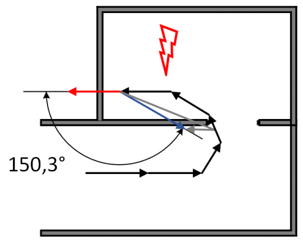

(b) Needles shortening of a step vector

Figure 4: Invalid trajectories indicated by multiple intersections

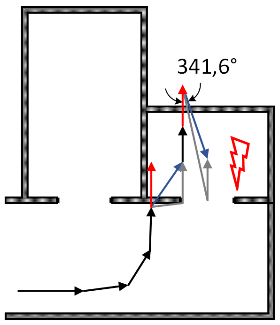
Another indication for an invalid trajectory is the change of step vector orientation by CWI. A correction through an opening that is located behind the user is unlikely and results in a change of direction that is greater than 90°, as illustrated in Figure 5.

B. *Finding alternative trajectories*

If a trajectory is found as invalid, an alternative needs to be found. Steps that potentially allow alternative trajectories are defined as follows: A step vector intersects a wall and is guided around that wall by CWI. In this case other openings are searched and collected along this wall. Figure 6 illustrates this case. A step vector that allows more than one possible trajectory around a wall is marked with a flag at runtime and then stored in the list of past steps. The search for steps with possible alternatives during backtracking is then done



(a) U-turn resulting in a dead end



(b) Nearby opening leading to a dead end

Figure 5: Scenarios of invalid trajectories indicated by invalid changes in direction

by checking if this flag is set. The actual computation of alternative step corrections is done by a modification of the CWI algorithm: Instead of finishing execution when a nearby opening is found, we collect all possible openings in a radius of 5 m. A candidate step that allows no valid alternatives is also marked by a flag. This step will not be re-evaluated in later backtracking efforts.

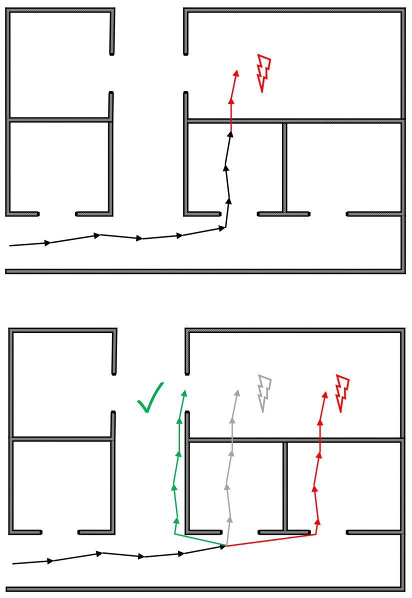


Figure 6: One invalid trajectory and two alternatives

C. *Choosing an alternative trajectory*

If the evaluation of one candidate step results in multiple valid trajectories, the most plausible trajectory is selected. For this purpose, each valid trajectory is scored according to its average corrected distance $\bar{s}$. It is defined as the arithmetic average over all corrected distances s_i within N steps, with $i \in \{1, 2, ..., N\}$. The corrected distance s_i is the distance between the endpoint of a corrected step vector and the endpoint from raw data as illustrated in Figure 7. Here, s_i and s_{i+2} have positive values while s_{i+1} is zero. The corrected distance $\bar{s}$ is larger for trajectories that are corrected often and significantly. Trajectories with a low $\bar{s}$ therefore fit well into the building geometry and are thus more plausible user trajectories. The trajectory with the lowest $\bar{s}$ is selected as the estimated user trajectory. To ensure comparability, the compared trajectories all have the same step count N.

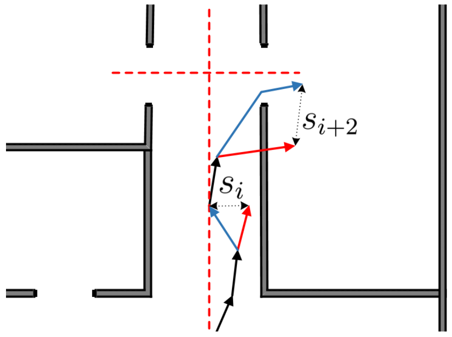


Figure 7: The corrected distance s_i of two steps

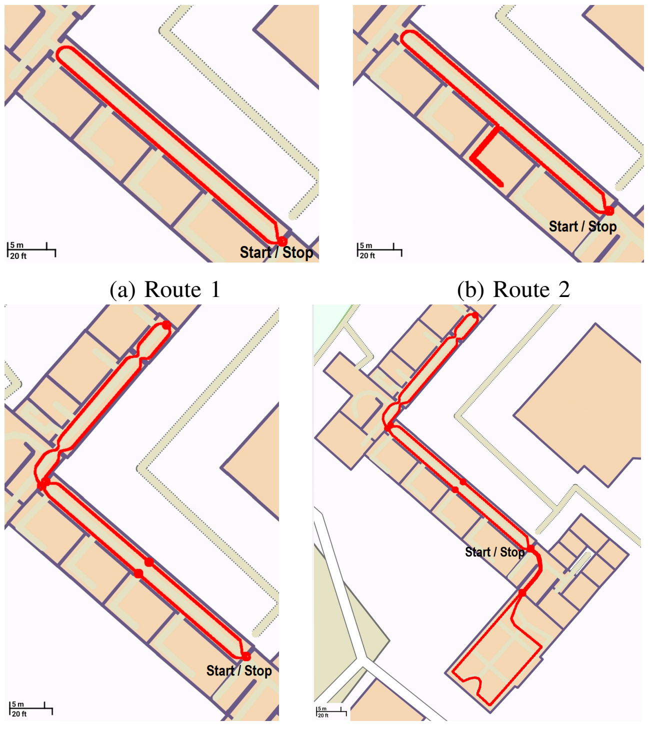
IV. Experiment

We use the same map and trajectory data as described in [17] to validate the described backtracking method. This includes four routes as seen in Figure 8. All routes have a singular start and end point in order to simplify accuracy evaluations. The distance between calculated start and end points is considered the overall estimation error.

While the localization works online, for this experiment the consecutive uncorrected step vector data from the Madgwick / ZUPT fusion described in [17] was recorded and the path estimation performed offline. The recordings were performed using a sensor node equipped with a Bosch Sensortec BMI160 inertial sensor and transmitting sensor signals to a connected Motorola Moto G2 smartphone using Bluetooth Low Energy. For the experiment, six participants each walked all four routes three times, resulting in 18 recordings per route and 72 recordings overall.

V. Results

As described in [17], the CWI correction method occasionally introduces significant errors in accuracy with increasing track length. Regarding short tracks like Route 1 and 2, the combination of CWI and CPI gains marginally better accuracy than CWI alone or no correction at all. This may be attributed to the effects of reduced positioning drift along straight track sections as illustrated in Figure 2. However, the combination of CPI and CWI still results in sporadic dead ends as seen by the large error bars in Figure 9. The described method corrects 18 out of 20 trajectories in which a dead end is encountered with one dead end remaining in Routes 1 and 2 each. As seen in Table I, an average accuracy of about 1.3% of the travelled distance is achieved on routes longer than $100m$, reaching the best accuracy of 1.22% on the longest route. The data suggests that longer, complex routes are reducing ambiguity regarding the users true path through a building and therefore



(a) Route 1 (b) Route 2 (c) Route 3 (d) Route 4

Figure 8: Routes used in the experiment as seen in [17]

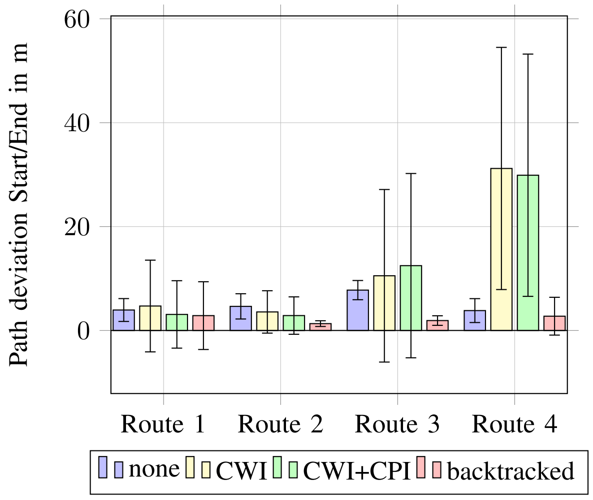
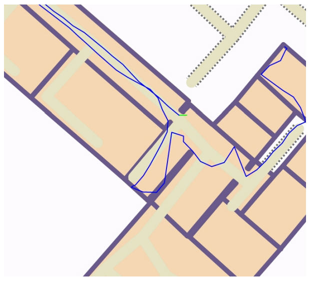
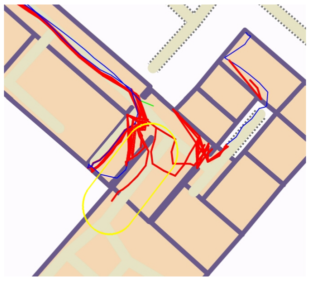


Figure 9: Localization accuracy: Path deviation at end of each route

increase the accuracy gained by backtracking. This behavior is the opposite of typical PDR behavior and shows that the use of building maps is hugely beneficial in improving PDR accuracy.

One case in which a dead end remains after backtracking is shown in Figure 10a. In Route 4, a room is erroneously entered after a right turn instead of entering a hall. The alternative trajectory into the hall, highlighted in Figure 10b, is found but rejected because it violates the validity rule illustrated


(a) Erroneous trajectory correction by CWI


(b) Rejected plausible trajectory highlighted

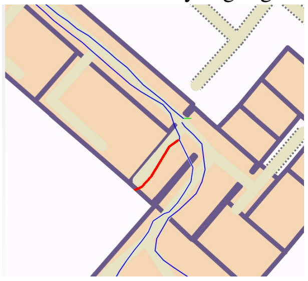

(c) Additional recursion recovers the trajectory

Figure 10: Increased reliability of localization through deeper backtracking recursion. Rejected trajectories are marked in red.

in Figure 4b by needlessly shortening a step vector before entering the hall. This is corrected by additional backtracking of this alternative trajectory, enabled by an increment of the allowed recursion level as shown in Figure 10c. This highlights the trade-off between reliable localization and increased computation by a broader search for alternative trajectories. The average computation time for a new raw step vector using a Motorola Moto G2 smartphone is 229.4μs for combined CWI and CPI. The average processing time including backtracking is 256.7μs, and therefore slightly increased. The maximum recorded time spend in a call of the backtracking procedure is 3.36ms. These measurements show that the backtracking method can provide real-time positioning on a mobile device.

VI. Conclusion

In this paper, we demonstrate an extension of PDR-based indoor navigation systems using a backtracking approach. More specifically, we show how the localization in a building resembles a depth first search through a tree of likely user trajectories. Our approach enables us to localize the user with comparably little initial effort, evaluating more complex trajectory adjustments by backtracking only if needed. The backtracking method presented here is modular: The trajectory validation, search for alternative corrections at a given part of the trajectory and scoring of possible valid trajectories may be replaced by other application specific methods. This enables backtracking for a variety of PDR approaches.

A future extension of this method would be the inclusion of trajectory shape matching to known paths and re-sizing of steps. Additional localization by coarse grained WiFi fingerprints, to distinguish in which room or corridor a user is located, may be used to assist the scoring and validation of alternative trajectories.

TABLE I: Accuracy results, measured as distance between start and end point, relative to route length

Route	Length in m	none	CWI	CWI+CPI	backtracked
Route 1	77.01	5.12%	6.12%	4.01%	3.71%
Route 2	99.44	4.67%	3.59%	2.89%	1.33%
Route 3	139.72	5.56%	7.54%	8.93%	1.37%
Route 4	225.97	1.69%	13.80%	13.23%	1.22%

References

[1] Z. Xiao, H. Wen, A. Markham, and N. Trigoni, "Robust indoor positioning with lifelong learning," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 11, pp. 2287–2301, 2015.

[2] S. He and S.-H. G. Chan, "Wi-fi fingerprint-based indoor positioning: Recent advances and comparisons," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 466–490, 2016.

[3] H. Li, X. Chen, G. Jing, Y. Wang, Y. Cao, F. Li, X. Zhang, and H. Xiao, "An indoor continuous positioning algorithm on the move by fusing sensors and wi-fi on smartphones," *Sensors*, vol. 15, no. 12, pp. 31 244–31 267, 2015.

[4] X. Li, J. Wang, and C. Liu, "A bluetooth/pdr integration algorithm for an indoor positioning system," *Sensors*, vol. 15, no. 10, pp. 24 862–24 885, 2015.

[5] Y. Zhuang, J. Yang, Y. Li, L. Qi, and N. El-Sheimy, "Smartphone-based indoor localization with bluetooth low energy beacons," *Sensors*, vol. 16, no. 5, p. 596, 2016.

[6] W. Sakpere, M. Adeyeye-Oshin, and N. B. Mlitwa, "A state-of-the-art survey of indoor positioning and navigation systems and technologies," *South African Computer Journal*, vol. 29, no. 3, pp. 145–197, 2017.

[7] R. F. Brena, J. P. García-Vázquez, C. E. Galván-Tejada, D. Muñoz-Rodriguez, C. Vargas-Rosales, and J. Fangmeyer, "Evolution of indoor positioning technologies: A survey," *Journal of Sensors*, vol. 2017, 2017.

[8] A. Correa, M. Barcelo, A. Morell, and J. L. Vicario, "A review of pedestrian indoor positioning systems for mass market applications," *Sensors*, vol. 17, no. 8, p. 1927, 2017.

[9] R. Harle, "A survey of indoor inertial positioning systems for pedestrians." *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1281–1293, 2013.

[10] W. Kang and Y. Han, "Smartpdr: Smartphone-based pedestrian dead reckoning for indoor localization," *IEEE Sensors journal*, vol. 15, no. 5, pp. 2906–2916, 2015.

[11] H. Zou, Z. Chen, H. Jiang, L. Xie, and C. Spanos, "Accurate indoor localization and tracking using mobile phone inertial sensors, wifi and ibeacon," in *Inertial Sensors and Systems (INERTIAL), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–4.

[12] Y. Li, Y. Zhuang, H. Lan, Q. Zhou, X. Niu, and N. El-Sheimy, "A hybrid wifi/magnetic matching/pdr approach for indoor navigation with smartphone sensors," *IEEE Communications Letters*, vol. 20, no. 1, pp. 169–172, 2016.

[13] J. Shang, X. Hu, F. Gu, D. Wang, and S. Yu, "Improvement schemes for indoor mobile location estimation: A survey," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[14] S. H. Shin, C. G. Park, and S. Choi, "New map-matching algorithm using virtual track for pedestrian dead reckoning," *ETRI journal*, vol. 32, no. 6, pp. 891–900, 2010.

[15] Widyawan, M. Klepal, and S. Beauregard, "A backtracking particle filter for fusing building plans with pdr displacement estimates," in *Positioning, Navigation and Communication, 2008. WPNC 2008. 5th Workshop on*, March 2008, pp. 207–212.

[16] J. Qian, L. Pei, J. Ma, R. Ying, and P. Liu, "Vector graph assisted pedestrian dead reckoning using an unconstrained smartphone," *Sensors*, vol. 15, no. 3, pp. 5032–5057, 2015.

[17] J.-P. Wolff, S. Stieber, F. Hölzke, D. Gis, C. Haubelt, P. Lepidis, and J. Meier, "Improving pedestrian dead reckoning using likely and unlikely paths," in *International Conference on Ubiquitous Positioning, Indoor Navigation, Location-Based Services (UPINLBS)*. IEEE, 2018, in press.