From Research to App: Personalized Inertial Navigation for the Visually Impaired

Tommy Moisan[1], Hanyuan Fu[2], Valérie Renaudin[3] and Mohamad Issam Sayyaf[2]

AME-GEOLOC, Univ. Gustave Eiffel, F-44344 Bouguenais, France

Abstract

The transition from research to real-world application is full of challenges. In this work, we present our efforts to integrate a personalized Pedestrian Dead Reckoning (PDR) framework, called MAPIN, into a real-time mobile application that displays the user's trajectory on a map using only inertial signals and map data. We have overcome the challenges of translating a Python implementation into a C++ implementation, assembling different modules from different research works and made compromises to fit the available hardware and computational resources in the mobile devices. Experimental results show that both the real-time and Python implementations produce very similar results, with execution times that are reasonable for a real-time application. Furthermore, the real-time MAPIN has been integrated with the guidance interface developed by Okeenea Digital, our industrial partner in the project, to create a proof-of-concept (POC) navigation application for demonstration purposes. Through this integration process, we have learned valuable lessons that will guide our future integration efforts.

Keywords

Smartphone, Pedestrian Dead Reckoning (PDR), Real-time implementation, Navigation aids for visually impaired people, Deep Learning

1. Introduction

For many visually impaired individuals, navigating unfamiliar environments alone can be stressful, particularly due to the fear of getting lost or not knowing how to reach a destination. While the proliferation of smartphones has enabled a growing number of navigation aids tailored for visually impaired users, many existing solutions rely heavily on external signals obtained through GNSS, beacons, or cameras. These signals are often unavailable, unreliable, or impractical in real-world settings, especially indoors or in crowded or dynamic environments.

In response to these limitations, inertial-based navigation has emerged as a disruptive approach. By leveraging motion sensors embedded in smartphones, these systems can estimate a user's position without relying on external signals or infrastructure. Our previous work introduced MAPIN (Mobility Adapted Pedestrian Inertial Navigation) [1], a personalized Pedestrian Dead Reckoning (PDR) framework that adapts to individual walking patterns to ensure optimal performance for all users. Unlike generic models, it was developed in collaboration with visually impaired people, incorporating their behaviors and preferences in the learning. While MAPIN showed strong performance in controlled, post-processed datasets, transitioning this into a real-time mobile application introduces new challenges: limited computational resources, variable sensor quality across devices, and the integration of independently developed research components.

This paper presents the real-time implementation of MAPIN on a smartphone platform (Fig.1). Our contributions include translating research algorithms into a C++ mobile application, the architectural integration of step detection and stride vector estimation modules, the evaluation of the system's performance in terms of execution time and the difference in positioning results between the real-time and post-processing versions. Furthermore, the real-time MAPIN has been integrated with the guidance

IPIN-WCAL 2025: Workshop for Computing & Advanced Localization at the Fifteenth International Conference on Indoor Positioning and Indoor Navigation, September 15–18, 2025, Tampere, Finland

tommy.moisan@univ-eiffel.fr (T. Moisan); hanyuan.fu@univ-eiffel.fr (H. Fu); valerie.renaudin@univ-eiffel.fr (V. Renaudin); issam.sayyaf@univ-eiffel.fr (M. I. Sayyaf)

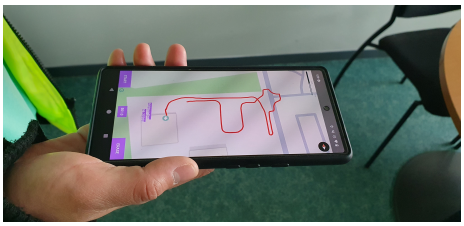

Figure 1: The MAPIN app interface displaying a real-time user trajectory

user interface developed by Okeenea Digital, our industrial partner in the project, to create a proof-of-concept (POC) navigation application for demonstration purposes. The POC app successfully guided a blind volunteer and a low-vision volunteer to their destination during an 8 to 10-minutes navigation. These contributions represent an important step toward bridging the gap between research prototypes and practical, inclusive navigation tools that operate reliably in unconstrained environments.

The rest of the paper is structured as follows. Section 2 reviews existing mobile navigation applications for visually impaired users. Section 3 introduces the MAPIN framework. Section 4 presents the challenge of implementing MAPIN in real-time. In section 5, we evaluate the real-time MAPIN in terms of computational time and accuracy in real-life setting. Section 6 discusses the lessons learned from this project. Section 7 concludes the paper.

2. Existing smartphone-based mobility aids for visually impaired users

Smartphones now offer a variety of mobility aids for visually impaired users, using various technologies, such as GNSS, cameras, beacons, and inertial sensors. Since this work focuses on the real-time implementation of an inertial-based navigation solution, we categorize existing smartphone-based mobility aids into two groups: Non-inertial-based and inertial-based.

2.1. Non Inertial-based Mobility Apps

Non-inertial mobility apps use external signals to estimate the user's position. The benefit of these solutions is that the user's position is estimated at each epoch independently from the previous estimate without positioning error accumulation. Most of these solutions rely on one of these three technologies: GNSS, vision, and beacon networks.

GNSS-based technologies support outdoor navigation, with specialized apps offering tailored features beyond standard accessibility tools. Examples are BlindSquare, Ariadne GPS and Seeing Eye GPS that offer voice guidance and tactile user interfaces tailored to visually impaired users. Key features include real-time position updates, street announcements and points of interest (POI). In dense urban areas, signal reflections and obstacles affect GNSS accuracy, and the signal is further attenuated indoors.

Smartphone apps such as GoodMaps and NaviLens use visual markers or spatially anchored codes to facilitate indoor positioning and navigation. GoodMaps uses a combination of LiDAR-based mapping and camera recognition to achieve meter-level indoor localization, while NaviLens uses high-contrast, color-coded QR-like codes that are readable from wide angles and distances. Both systems provide reliable guidance in GPS-denied environments. Since the navigation instructions are obtained from the scene analysis, the user must point the phone towards their surroundings and therefore hold it in their hand. This is a major limitation as many visually impaired people prefer to have their hands free [2]. These vision based approaches are also severely hampered by obstacles such as crowds or changing light conditions.

Beacon technology is only deployed in certain places (railway stations, bus stops, museums, shopping malls, etc). Positioning relies on radio signals like UWB (Ultra Wideband), BLE (Bluetooth Low Energy) and WiFi transmitted from the known locations of the beacons. Evelity, the android application developed by Okeenea, our industrial partner, utilizes a combination of BLE and inertial signals to provide accurate turn-by-turn indoor navigation. Routes can be pre-optimized to account for structural obstacles like furniture or columns. Step-Hear processes BLE signals with the interaction of the RFID (radio frequency identification) wristband to provide contextual information as users approach

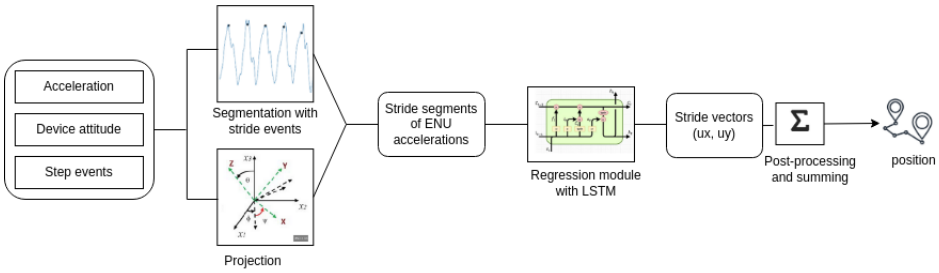

Figure 2: MAPIN framework overview [1]

predefined zones. This type of technology comes with high deployment costs for equipment and surveying the terrain to estimate either the accurate coordinates of the beacons or the radio fingerprints of indoor map grids to apply fingerprinting techniques.

2.2. Inertial-based Mobility Apps

Inertial-based solutions rely on the PDR technique, which uses body-centric sensing and movement models to recursively estimate the user's displacement. The user's position and orientation are inferred by accumulating these estimated displacements. These self-contained solutions can operate in any environment and tolerate a variety of device locations, such as handheld or carried in a pocket. However, without external signals for absolute position correction, these solutions suffer from growing position error as the travel distance increases. Consequently, the positioning estimates are often improved with the map matching technique.

Waymap [3] a free IOS/Android app, tracks the user's position by estimating the user's displacement on every step and map matching is used to constantly check the user's location and maintain accuracy over long distances. Similarly, WayFinding [4], an experimental iPhone app from UC Santa Cruz, uses two PDR techniques in parallel, a gait-driven PDR based on step count and a sampling frequency-driven [5] AI-based PDR RoNIN [6]. Their PDR models are fine-tuned with data collected from blind volunteers to improve their accuracy for this population. Map matching with a particle filter improves the overall positioning result.

Adapting PDR solutions to users is another way to reduce the accumulated positioning error, as demonstrated by WayFinding. Our work takes this customization to a new level. MAPIN is the first fully personalized PDR framework. As inertial signals reflect individual walking patterns, the PDR model can be personalized for optimal performance on a specific user. To validate this approach, we have implemented the MAPIN framework in an Android app. Real-world deployment is essential to evaluate the usability, robustness, and efficiency of the system under different conditions, considering the variability in smartphone sensor quality, user movement, and device placement. The remainder of this paper presents the app's design and implementation, along with an initial evaluation of its performance in real-world scenarios, laying the foundation for inclusive, adaptive, and autonomous navigation solutions.

3. Overview of the MAPIN framework

The MAPIN framework was developed in a participatory scientific project, in which the end users, i.e. visually impaired persons, are included in the research. During 5 years, more than 300 km of human gait data were collected with more than 30 volunteers, 20 of them being visually impaired. Different individuals have different walking gait fingerprints and the visual impairment introduces supplementary variability. In this context, a generic model might not be optimal for all users. In particular, visually impaired users require greater positioning accuracy and robustness than sighted individuals to ensure precise and reliable guidance. Building a personalized approach addresses these challenges. The MAPIN framework [1] adapts PDR models to individual user and device locations, based on the observation that the same person wearing the device in the same location results in recurring acceleration patterns observed at each gait cycle.

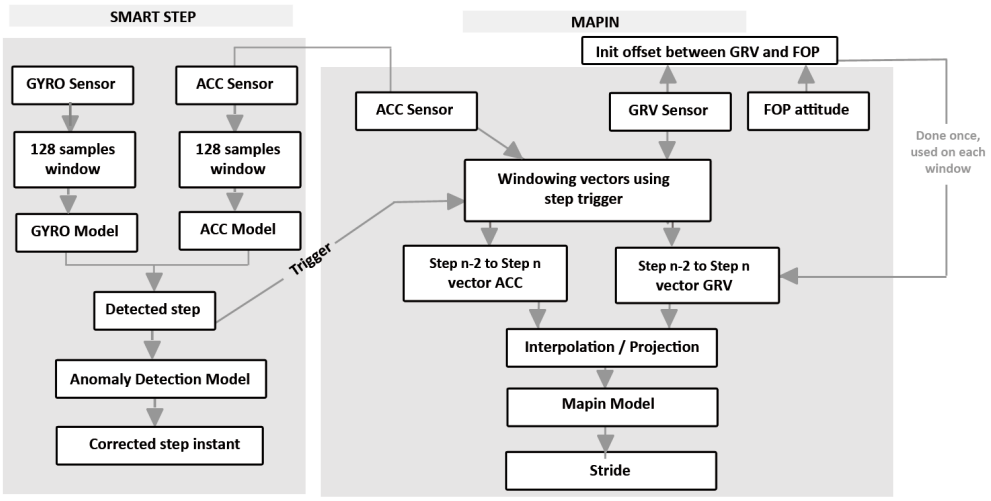

Figure 3: Algorithm of real-time implementation of MAPIN

Fig. 2 gives an overview of the MAPIN framework. Accelerometer readings, device attitude estimates and step events are the inputs to the framework. The device attitude is estimated by an Extended Kalman Filter (EKF) [7], and the step events are detected by Smartstep [8]. The raw accelerometer readings are segmented into stride intervals (containing two steps) and transformed into the navigation frame (North-East-Down, NED or East-North-Up, ENU) via the device attitude. We obtain stride intervals of different lengths, which we interpolate to a uniform length equal to 1.5 times the accelerometer sampling frequency, i.e. 100 Hz for the smartphone. The interpolated stride segments are fed into a personalized Long Short-Term Memory (LSTM) [9] module to estimate the user's stride vectors. They correspond to the displacements along the north and east axes of the user over the different stride intervals. The latter are smoothed and summed to obtain the user's position. To learn the realistic fingerprints of visually impaired walking, their training data was collected in real and varied environments, including urban neighborhoods with elements such as pedestrian crossings, obstacles, remote-controlled traffic lights and ramps.

4. Post-Processing to Real-Time: More Than Language Translation

MAPIN was first implemented in Python for our self-developed navigation device ULISS[10], in a post-processing way, i.e. the data recorded by ULISS was processed on a computer. When the version for ULISS was mature, we developed a MAPIN version that processes smartphone data, still in post-processing. Moving to a smartphone apps involved implementing the MAPIN framework in C++, which is more suitable for real-time operations on edge devices. Up to now, the personalized calibration of MAPIN walking gait models is still done offline. The transition from ULISS data to smartphone data and the switch from post-processing to real-time posed several challenges.

4.1. Interconnection between Smartstep and MAPIN

Figure 3 gives an overview of the architecture of our real-time implementation. The system comprises two main modules: the step detection model Smartstep [8] and the stride vector estimation model MAPIN. MAPIN is triggered by step detection and processes acceleration and attitude data within two step intervals. Reliable step detection is essential for MAPIN to function correctly, as missed steps are a major source of error. Initially, Smartstep and MAPIN operated at 200 Hz for ULISS data. When we implemented Smartstep on smartphones, we decided to reduce the sampling frequency to 100 Hz to lower the computational overhead. Consequently, MAPIN models were retrained with data sampled at 100 Hz to ensure compatibility with mobile devices.

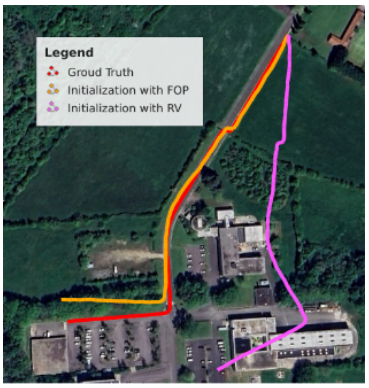

Figure 4: Different options for initializing the attitude angles: using RV (pink) or Google FOP (orange). The ground truth trajectory is shown in red.

Figure 5: Experimental setup

4.2. Device Attitude Estimation on Android: Challenges and Alternatives

The Android system provides several attitude tracking options [11], including the Rotation Vector (RV) and Game Rotation Vector (GRV). The RV is a quaternion representing the device's attitude, expressed in a coordinate frame with the Y axis pointing to the North and the Z axis pointing upward. It is estimated using acceleration, angular rate and magnetic field data. Whereas the GRV is a quaternion representing the device's attitude, computed using acceleration and angular rate but excluding the magnetic field. Thus, the Y axis does not point North but rather to an arbitrary reference, which may drift over time. The GRV is more robust to magnetic disturbances but does not align with the North, whereas the RV does. We decided to use the GRV for attitude tracking because of it is immune to noisy and perturbed magnetic field measurements, which are common in urban spaces with low-cost built-in magnetometers.

To align the GRV with the navigation frame, we initially estimated the offset between the GRV and the RV at the beginning of the recording and during a static phase with minimal magnetic disturbances. This offset is added to the GRV to obtain the device's attitude expressed in the ENU (East-North-Up) frame. However, we have observed instability in the RV, which can lead to large errors in the initial walking direction. An alternative to RV is the attitude provided by the Google Fused Orientation Provider (FOP) [12], which exhibits less instability and rarely results in a huge initial walking direction error. Fig. 4 shows the impact of the choice of method for initializing the attitude on the initial walking direction. It is important to note that both RV and FOP rely on the magnetometer, which must be properly calibrated by waving the device in an "8" shape pattern several times in an environment free from strong artificial magnetic fields.

4.3. Aligning Data from Different Sensors

As the data from different sensors arrive at different times and are sampled at different frequencies, interpolation is required to align all data to a common timeline.

In a post-processing approach, we process cold data, i.e. the data for the entire run is available. Without real-time constraints, we adopted the simplest method to align acceleration and GRV, namely the interpolation of the GRV with the time stamps of the accelerations over the whole track.

In contrast, a real-time solution processes the data as soon as it arrives. We continuously collect sensor data and apply stride segmentation at each detected step instant. The alignment between acceleration and GRV is performed within each stride interval, instead of being performed on the entire recording.

The different processes of data alignment in the post-processing version and in the real-time version lead to slightly different input instances for the LSTM model to estimate the stride vectors.

4.4. Hardware Components

During this integration project, we tested our real-time solution with various wearable and found that their performance can be affected by the quality of the embedded hardware components. Smartphones

with low-quality inertial sensors may not be able to achieve the required sampling frequency of 100 Hz, resulting in a large number of missing step instants and degraded performance of MAPIN. For example, Samsung A23 accelerometer outputs readings at 80 Hz. In addition, some mobile devices do not include the necessary sensors to run our algorithm. For instance, Xiaomi POCO C65 does not have a gyroscope, without which the GRV is not available. To ensure compatibility, we have included a functionality that checks the presence of the required sensors to inform the user whether the application will be able to function with their device or not.

5. Experimental Assessment

This section evaluates the real-time MAPIN implementation through three experiments: one evaluating execution time for real-time suitability, one comparing the real-time stride vector results with those by the Python version to ensure that performance is preserved, and a final experiment in which MAPIN was integrated into a navigation guidance app and tested by two visually impaired volunteers.

Fig. 5 shows the experimental setup for the first two assessments. A volunteer walks with the smartphone Google Pixel 6 in a "texting" mode in outdoor and indoor environments, mostly on flat ground, ramps are also included, but stairs are excluded. The user's ground truth trajectory is provided by the foot-mounted reference device ULISS [13]. The ground truth stride vector is computed as the user's ground truth displacement between two stride instants detected by Smartstep. All recordings were started in an outdoor environment without strong artificial magnetic fields.

5.1. Execution Time

One of the real-time challenges is to obtain results within a limited period of time that allow us to estimate the user's position without significant delay, thus enabling the provision of navigation instructions at the right time.

Sensor data is processed in two steps: step detection and stride vector estimation. The average time delay between step detection and the corresponding stride vector computation on a Google Pixel 6 is 13.4 ms, with a standard deviation of 1.93 ms, calculated from a population of 50 steps. The delays were calculated using the device's timestamps (System.nanoTime()) at the moment each result is available. Given that the average duration of the step is approximately 0.5 s, the result shows that MAPIN operates efficiently and is well suited for real-time use. However, these results are specific to this device and may vary with other smartphone models due to differences in hardware performance.

5.2. Comparison of Python and Real-Time Implementation Results

Evaluated over 40 km with a Google Pixel, the post-processing version of the MAPIN framework reports an average **stride length error (SLE)** between 5 and 9 cm for all users, compared to 7 to 30 cm for different users with a generic PDR model [6].

To assess the accuracy of the real-time application, we compare the estimated stride vectors with those of the original Python implementation of MAPIN on the same dataset.

Fig. 6 shows that while the Python and real-time versions produce slightly different stride vectors, their results remain closely aligned. This small difference is due to the different data alignment methods used in the post-processing and real-time versions, as detailed in section 4.3.

Table 1 reports the average **or (SVE)** of the two versions on two walking tracks, computed by:

$$SVE = ||\vec{u_{gt}} - \vec{u_{es}}|| \tag{1}$$

where $\vec{u_{gt}}$ is the ground truth stride vector and $\vec{u_{es}}$ the estimated one.

The real-time result of MAPIN is considered acceptable if the difference between the SVE of the two implementations is below 1.4 mm, which is 1% of the average stride length of 1.4 m.

The difference between the SVE of the two implementations is below 1 mm for the two test tracks. Therefore, we consider that the deviation introduced by the real-time implementation has no significant impact on the accuracy of the model.

Table 1
Comparison between real-time processing and post-processing results (SVE)

Track	Walking distance (m)	Python error (m)	Real time error (m)
1	279	0.1967 ± 0.1643	0.1942 ± 0.1658
2	286	0.1683 ± 0.1905	0.1791 ± 0.1927

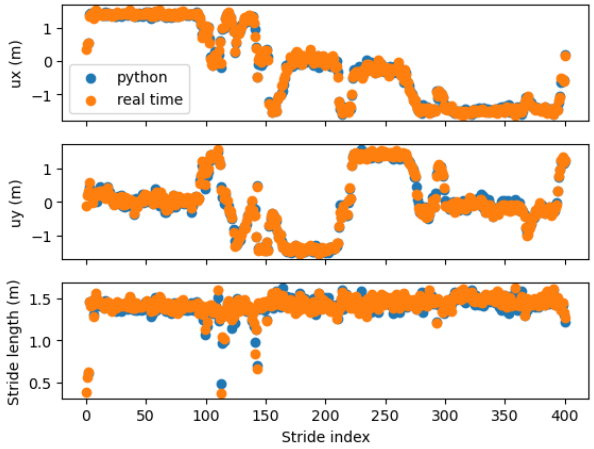

Figure 6: Comparison between stride vectors output by the real-time and Python implementation of MAPIN. **Top:** u_x (East component); **Middle:** u_y (North component); **Bottom:** Stride vector lengths.

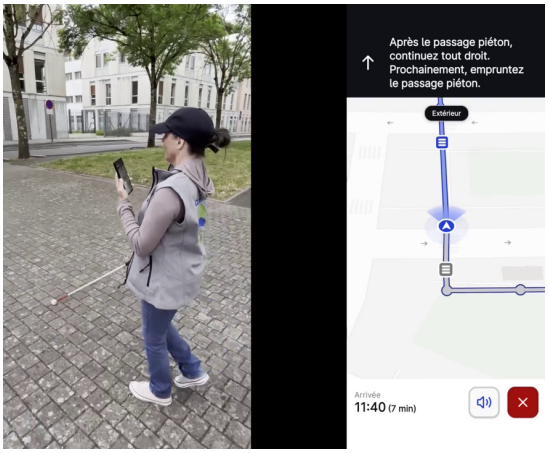

Figure 7: A blind volunteer using our POC navigation application and the application's user interface.

5.3. Real-life navigation tests

To evaluate the real-time MAPIN framework in the real world setting and to approach our final goal, which is to provide visually impaired users a mobile navigation aid that guides them to their destinations, we integrated MAPIN into the Evelity app, developed by our industry partner Okeenea Digital.

In this fusion, MAPIN provides the user's position estimates and Evelity provides the navigation graph, itinerary planning and the user interface with vocal guidance instructions. An instruction is triggered when the estimated position is within a 3 m radius of a direction changing point on the graph. In addition to navigation instructions, the app also announces crosswalks in advance to help the user anticipate them and reassures the user during long straight-line walking by saying "Go straight ahead."

Two visually impaired volunteers tested the proof-of-concept (POC) navigation app (Fig. 7). We have videotaped their navigation and these videos were presented during the project closing event held on June 5, 2025, and are available online [1] [2]. These experiments enabled us to validate the performance of MAPIN in real-life setting and address the aspect of user experience, which had not been explored until now.

6. How the Integration Shaped the Research Methodology

The beauty of this research project lies in the opportunity to translate research findings into a real-world application. This transition has shaped our research methodology and will serve as a guide for similar projects in the future.

6.1. Lessons from real-time constraints

Without the real-time constraint, researchers have the freedom to use "future information" to simplify data processing and obtain optimal results. For instance, this could involve interpolating signals from different sensors or applying filters across the entire walking track. In a real-time implementation, only past and present information is available. Therefore, interpolation and filtering can only be applied

[1] https://www.youtube.com/watch?v=vLLbZiq5BUY
[2] https://www.youtube.com/watch?v=mPj79r-9Q_U

within the current sliding or segmented time window. As explained in section 4.3, aligning data within each segmented window and across the entire walk track yields slightly different results.

Without progressing to real-time integration into the end-user application, it is impossible to determine whether the simplifications or assumptions made during the research phase, in developing the scientific basis of the algorithms, are problematic for implementation in the final app. If this is the case, the research should be reviewed. Otherwise, minor discrepancies between the research results and the real-time implementation can be accepted, as was done in the case of data alignment.

6.2. The impact of combining different research outputs into one app

The research project includes several blocks, such as Smartstep and MAPIN, which were developed independently to answer different research questions. Every research task aims to perfect its own solution. However, the integration of several research works into one application implies certain adaptations and compromises.

To deal with the user's irregular movements that cause overdetection of steps, an anomaly detection model [14] based on a dense autoencoder has been integrated into the Smartstep module. The anomaly detection model can eliminate 80% of false step detection, but also 5% of true step detection. Accepting this tradeoff, Smartstep has been released on Google Play Store as a step counting application [15]. During integration, we observed that MAPIN has a certain "immunity" to irregular motion [16]. Since a MAPIN model is tailored to a user and a device location, it is trained with highly similar gait data. When MAPIN encounters unfamiliar gait patterns associated with irregular motion, it simply outputs tiny stride vectors that have a limited impact on position estimation. On the other hand, underdetection of steps is a major source of error for MAPIN, resulting in inaccurate distance estimation. Therefore, we disabled the anomaly detection model for MAPIN.

This adjustment led us to maintain two versions of Smartstep: the first runs without anomaly detection and is dedicated to MAPIN, and the second with anomaly detection for accurate step counting.

7. Conclusion

In this study, we integrated the personalized PDR framework MAPIN, developed with the participation of the visually impaired community, into a real-time smartphone application. This work brings us closer to our final goal, which is to provide visually impaired people with a precise and autonomous mobile navigation aid.

We have overcome the challenges of translating a Python implementation into a C++ implementation, assembling different modules from different research works, and making compromises to fit the available hardware and computational resources in the mobile device. Experimental results confirm that the real-time implementation maintains accuracy comparable to the original offline version, with execution times well suited to continuous pedestrian navigation. Additionally, we validated MAPIN's performance in real-life scenarios with visually impaired users using the POC navigation guidance app, developed in collaboration with Okeenea Digital. Due to the involvement of the industrial partner in the project, the source code of MAPIN is not publicly available.

This integration process revealed two key lessons. First, the successful translation of research results into applications relies on close and continuous collaboration between researchers and software engineers. Second, while individual components may perform well in isolation, assembling them into a cohesive real-time system often requires compromises to ensure overall stability and usability.

Let's note that without absolute position estimates to limit the error, the positioning error in PDR increases with walking distance. To mitigate this, the integration of a particle filter into our positioning application is already planned and in progress. This will increase the accuracy of the positioning results and improve user experience.

Declaration on Generative AI

During the preparation of this work, the author(s) used X-GPT-3.5 in order to traduce some sentences from french to english. After using these tool, the author(s) reviewed and edited the content as needed

and take(s) full responsibility for the publication's content.

References

[1] H. Fu, V. Renaudin, T. Bonis, N. Zhu, MAPIN: Mobility Adapted Pedestrian Inertial Navigation Using Smartphones for Enhanced Travel of the Visually Impaired, in: 2024 14th International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2024, pp. 1–6. doi:10.1109/IPIN62893.2024.10786129.

[2] R. Tachiquin, R. Velázquez, C. Del-Valle-Soto, C. A. Gutiérrez, M. Carrasco, R. De Fazio, A. Trujillo-León, P. Visconti, F. Vidal-Verdú, Wearable urban mobility assistive device for visually impaired pedestrians using a smartphone and a tactile-foot interface, Sensors 21 (2021). URL: https://www.mdpi.com/1424-8220/21/16/5274. doi:10.3390/s21165274.

[3] Waymap Ltd., Waymap – indoor, accessible navigation, 2025. URL: https://www.waymapnav.com/, accessed 23 June 2025.

[4] C. H. Tsai, F. Elyasi, P. Ren, R. Manduchi, All the way there and back: Inertial-based, phone-in-pocket indoor wayfinding and backtracking apps for blind travelers 17 (2024). URL: https://doi.org/10.1145/3696005. doi:10.1145/3696005.

[5] H. Fu, Y. Kone, V. Renaudin, N. Zhu, A Survey on Artificial Intelligence for Pedestrian Navigation with Wearable Inertial Sensors, in: 2022 IEEE 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2022, pp. 1–8. doi:10.1109/IPIN54987.2022.9918136.

[6] S. Herath, H. Yan, Y. Furukawa, Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, amp; new methods, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 3146–3152. doi:10.1109/ICRA40945.2020.9196860.

[7] V. Renaudin, C. Combettes, Magnetic, Acceleration Fields and Gyroscope Quaternion (MAGYQ)-Based Attitude Estimation with Smartphone Sensors for Indoor Pedestrian Navigation, Sensors 14 (2014) 22864–22890. URL: https://www.mdpi.com/1424-8220/14/12/22864. doi:10.3390/s141222864.

[8] N. Al Abiad, Y. Kone, V. Renaudin, T. Robert, SMARTphone inertial sensors based STEP detection driven by human gait learning, in: 2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN), IEEE, 2021, p. 8. URL: https://ieeexplore.ieee.org/abstract/document/9662513. doi:10.1109/ipin51156.2021.9662513.

[9] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[10] M. Ortiz, M. De Sousa, V. Renaudin, A New PDR Navigation Device for Challenging Urban Environments, Journal of Sensors 2017 (2017) 1–11. doi:10.1155/2017/4080479.

[11] Android Developers, Position sensors, 2024. URL: https://developer.android.com/develop/sensors-and-location/sensors/sensors_position, accessed: 2025-04-17.

[12] Google, Fused location provider api, 2024. URL: https://developers.google.com/location-context/fused-location-provider, accessed: 2025-04-17.

[13] N. Zhu, V. Renaudin, M. Ortiz, Y. Kone, C. Ichard, S. Ricou, F. Gueit, Foot-mounted INS for resilient real-time positioning of soldiers in non-collaborative indoor surroundings, in: 2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN), IEEE, 2021, pp. 1–8.

[14] M. I. Sayyaf, N. Zhu, V. Renaudin, T. Feigl, Step Detection Enhanced by Anomaly Filtering, in: Proc. Intl. IEEE Applied Sensing Conference (APSCON), 2025, pp. 1–4.

[15] M. I. Sayyaf, N. Zhu, V. Renaudin, Geoloc smart step, Mobile application software, https://play.google.com/store/apps/details?id=fr.univeiffel.geolocimu&hl=fr_CA, 2025. Last updated 17 January 2025; accessed 25 June 2025.

[16] M. I. Sayyaf, N. Zhu, V. Renaudin, Advanced step detection with anomaly filtering for enhanced positioning accuracy, in: Proceedings of the 2025 IEEE/ION Position, Location and Navigation Symposium (PLANS), IEEE, Utah, USA, 2025.