Mobixim: A Framework for Devising Collaborative Algorithms

Alpha Diallo[1], Benoît Garbinato[1]

[1]University of Lausanne, Switzerland

Abstract

Indoor Tracking Systems (ITS) are becoming increasingly popular due to their affordability and the services they provide, such as indoor navigation and asset tracking. Most of the ITS presented in the literature or deployed in the real world rely on dedicated infrastructures that are costly and difficult to deploy and maintain. An alternative lies in infrastructure-free ITS based on inertial sensors embedded in each mobile device, but such ITS suffer from an accumulation of errors degrading their accuracy. To mitigate this accuracy degradation, an approach consists in fostering collaboration between moving devices. However, a major limitation of this approach lies in the complexity of devising such collaborative inertial-based algorithms. In this paper, we address this limitation by proposing MobiXIM, a framework that unifies the processes of devising, evaluating, and fine-tuning collaborative inertial-based algorithms. In addition, we propose a midpoint algorithm devised using our proposed framework. Through this experiment, we highlight each of the components of MobiXIM that operate separately and communicate with each other, thus allowing researchers to focus solely on developing specific parts of their solutions.

Keywords

Indoor tracking, framework, reproducibility, collaboration, positioning, distributed systems.

1. Introduction

Over the past years, the demand for location-based services has grown tremendously. These services, such as store locators, proximity marketing, or mobile games, mostly rely on Global Navigation Satellite Systems (GNSS), such as the GPS or Galileo, which operate in outdoor environments. There is also a need for such services in indoor environments as we observe a growing demand for location-based marketing, mobile navigation, assets tracking, etc. [1].

Indoor location-based services rely on Indoor Tracking Systems (ITS) that are not yet ubiquitous because there is no standard ITS, and most of them require the deployment of costly infrastructure that is difficult to deploy and maintain. To overcome the need for infrastructure, some solutions rely on embedded sensors available in most mobile devices, such as the accelerometer, the gyroscope, and the magnetometer. These sensors are defined as inertial as they measure quantities based on physical laws of motion (i.e., by indirectly measuring specific

Proceedings of the Work-in-Progress Papers at the 13th International Conference on Indoor Positioning and Indoor Navigation (IPIN-WiP 2023), September 25 - 28, 2023, Nuremberg, Germany

*Corresponding author.

†These authors contributed equally.

alpha.diallo@unil.ch (A. Diallo); benoit.garbinato@unil.ch (B. Garbinato)

0000-0003-4894-1750 (A. Diallo); 0000-0002-3952-9273 (B. Garbinato)

forces) [2]. In doing so, the accelerometer measures the acceleration on 3-axis, the gyroscope measures the angular velocity, and the magnetometer measures the Earth's magnetic field. Coupled with mobile devices possessing some computing capabilities, inertial-based ITS can operate without infrastructures. The drawback of inertial-based ITS is the negative impact of accumulated errors on their accuracy. In the literature, some authors propose collaboration between devices to improve the accuracy of these systems [3, 4, 5, 6, 7].

Practically, this collaboration consists in leveraging interactions with nearby devices to correct the accumulated errors of positioning caused by noisy sensors. More precisely, mobile devices exchange their respective estimated locations and apply some aggregation methods to compute their new and hopefully more accurate location estimate. In fact, most mobile devices can interact with nearby devices by exchanging data or signalling their presence.

Yet, devising, evaluating, and fine-tuning collaborative ITS is a time-consuming process due to various reasons. First, most of the collaborative algorithms presented in the literature are evaluated in simulated environments and with virtual data [8, 9, 10, 11, 12]. Such environments fail to capture the complexity of real-world scenarios. Secondly, it is difficult to conduct experiments due to the significant number of synchronous interactions between participants that need to be considered. Finally, collaborative algorithms are built on top of existing algorithms used by traditional ITS, such as filtering algorithms for signal processing and positioning algorithms to estimate a device's location. Such structure makes it difficult for researchers to focus only on devising collaborative algorithms.

The problem we address in this paper is how to facilitate the process of devising, evaluating, and fine-tuning collaborative algorithms. In doing so, we propose MobiXIM, a novel framework that offers a mechanism for aggregating real-life data from multiple sources with the aim of running collaborative algorithms in a controlled environment. By using a plugin architecture, we lay the foundation for code reuse, and we aim to accelerate the time for developing new collaborative algorithms. Our contribution is intended as a step forward in setting standards for building ITS and allowing for the easy reproducibility of algorithm evaluations. The code source of MobiXIM is available on GitHub[1].

The rest of the paper is decomposed as follows. In Section 2, we discuss the system model and the problem statement by highlighting the challenges encountered by researchers and the problems addressed in this paper. Section 3 presents the framework. In Section 4, we devise a collaborative algorithm using the framework and evaluate its performance. Section 5 discusses its architecture. In Section 6, we list works related to ours. In Section 7, we discuss our findings and present what we intend to do in the future. We conclude the paper in Section 8.

2. System Model & Problem Statement

This paper considers mobile devices as aware of their environment, capable of signaling their presence and exchanging data with nearby devices. Each of these devices has a physical location that may change over time. A location is a tuple representing coordinates in the form of (ϕ, λ) where ϕ represents a latitude and λ a longitude in a geographic representation system. A set of

[1] https://github.com/doplab?q=mobixim

locations is defined as a trajectory. As opposed to an inertial trajectory, approximated using inertial measurements, a groundtruth represents a pedestrian's actual trajectory or true path.

We define an ITS as a system that tracks people or assets in indoor environments where GNSS-based solutions fail to operate. We distinguish two types of ITS described hereafter.

- **Infrastructure-based ITS.** They rely on fixed equipment such as Bluetooth Low Energy (BLE) beacons or WiFi routers. These pieces of equipment are used as reference points or for communication purposes. In some cases, such as fingerprinting, infrastructure-based ITS offer decent accuracy. However, they are costly regarding equipment, deployment, and maintenance time.
- **Infrastructure-free ITS.** Such systems mostly use Inertial Measurement Units (IMU) embedded in most mobile devices to estimate their location. By removing the need for infrastructure, these systems are more affordable. However, inertial-based ITS are less accurate as they accumulate errors over time.

In an infrastructure-free ITS, mobile devices can collaborate with nearby devices to keep low costs and improve accuracy. In such systems, defined as *collaborative ITS*, devices are capable of signalling their presence to nearby devices using communication interfaces like BLE or WiFi.

To compute their position estimates, devices run two different types of algorithms.

- **Local algorithms.** They estimate a device's location using data collected by its embedded sensors. They comprise filtering (Low pass filter, High pass filter, etc.) and positioning (Pedestrian Dead Reckoning, fingerprinting, etc.) algorithms.
- **Collaborative algorithms.** These algorithms use the estimated location computed by the local algorithms and the distance between two devices to improve their accuracy.

Problem statement

Devising collaborative algorithms is a complex task requiring data from many devices interacting in a controlled environment. The literature addresses this challenge by relying on synthetic data generated during a simulation [13, 14, 12**?**]. However, synthetic data can be biased or unrealistic, thus failing to capture the real movements of people in an indoor environment. Other authors use real-life data to evaluate their algorithms. This approach is challenging as collaborative algorithms require synchronisation between multiple devices. For instance, studies show that collaborative algorithms perform better with a large number of participants [15, 13]. Achieving this can induce a high cost for running experiments.

In addition to this cost, another factor to consider is the complexity of the algorithms. Indeed, collaborative algorithms work on top of local algorithms. Each algorithm has some parameters. Therefore, fine-tuning each parameter has an impact on the accuracy of the system.

Coupled with a large number of participants each running their own local algorithms with predefined parameters, testing each scenario quickly ends up being a very complex task.

3. The MobiXIM Framework

The MobiXIM framework addresses the complexity of devising, evaluating, and fine-tuning collaborative inertial-based algorithms, by proposing a unified process supported by a software

platform. MobiXIM is an open-source framework to capture real-life data from individuals, reproduce their movements and combine multiple real executions with their respective groundtruths in a controlled environment.

3.1. Components

MobiXIM comprises two main parts described as follows.

Web platform. Developed in Python and Javascript, the web platform is accessible via a browser. It is used to set up experiments and fine-tune the algorithms.

Companion App. Also defined as a data collection app, it is built in Java and intended for deployment on Android devices. The companion app is used by participants to collect inertial data along a path designed using the Web platform.

3.2. Process flow

MobiXIM reproduces the typical iterative process followed by researchers. This process consists of three main stages: *devise* by using a plugin architecture to facilitate code reuse, *evaluate* by proposing a mobile app that collects real-life data and respective groundtruths, *fine-tune* by offering a web platform and an API to combine the data collected by participants and running algorithms. More precisely, MobiXIM requires researchers to follow a predefined process flow of iterative steps described hereafter.

1. *Planning a scenario.* It consists of defining a floorplan, drawing the groundtruths by adding landmarks on a map and assigning each groundtruth to a participant.
2. *Collecting data.* Once the groundtruths are planned, participants start walking along the landmarks. They can do it simultaneously or independently if preferred. During this process, inertial data is collected at a fixed time interval.
3. *Exporting data.* Data collected by each device is then exported to the web platform.
4. *Merging data.* The web platform aggregates the data collected by multiple participants in a single place.
5. *Parameterizing the algorithms.* Collected data are linked to local algorithms responsible for computing inertial-based estimate trajectories. In this step, researchers fine-tune the local algorithms for each trajectory and select a collaborative algorithm for the experiment. MobiXIM offers the flexibility to use existing algorithms or to integrate new ones as plugins.
6. *Reproducing the collected data.* On the web platform, researchers reproduce the trajectories followed by participants and those generated by the algorithms.
7. *Evaluating the algorithms.* The final step is to observe how the algorithms perform by using a benchmark.
8. *Repeating the process.* Researchers can repeat Steps 5 to 7 to fine-tune their algorithm until they obtain satisfactory results.

4. Experiment & Evaluation

In this section, we explain how we use the framework to build an inertial-based collaborative ITS, and then we introduce and evaluate a collaborative algorithm. For this experiment, 5 participants collected 14 groundtruth trajectories, plus their corresponding inertial measurements on a single floor of a university building covering an area of up to $8400m^2$. First, we store all the inertial data collected from the different real mobile devices on the web platform; then, the local algorithm computes the approximate inertial trajectories before running the collaborative algorithm.

4.1. Experiment

The experiment consists of tracking participants using Android tablets embedding inertial and wireless sensors capable of exchanging data with a remote server. For the experiment, we use a single floorplan composed of multiple rooms interconnected by corridors.

Constructing the floorplan. The first step before running the experiment is to design the floorplan. This step is optional if the researchers do not plan on using the characteristics of a floorplan (walls and objects detection, convoluted boundaries, etc.). We propose *GeoJSON* to design floorplans. It is a standard format easy to manipulate with many Geographic Information Sytems (GIS) tools such as QGIS[2], GeoJSON.io[3] or Mapbox studio[4]. An advantage of using such a format is the flexibility for adding properties to its features. Features are polygons that compose a deployment environment. The properties associated with these features can be names, types (for example, interior wall, door, furniture, etc.), floor numbers, or elevation. The properties can also indicate, for example, the possibility for a device to cross a feature or not (open or closed door, restricted areas, etc.).

Planning a scenario. Prior to the data collection, groundtruths must be defined and assigned to scenarios. As described in Section 3, a groundtruth consists of multiple landmarks placed on a map.

Data collection. After designing the groundtruths, participants scan a generated QR code to obtain the scenario they should execute. Using the companion app, they walk through each point of the groundtruth while holding their devices at chest level. After executing their paths, participants send the collected data to the web platform for processing.

Setting up the simulation. Before running the simulation, we can filter the raw inertial data and fine-tune some parameters, such as the participants' step length, initial heading, and detection range. After setting up these parameters, we should select a positioning algorithm to generate each inertial trajectory.

[2]https://qgis.org
[3]https://geojson.io
[4]https://www.mapbox.com/mapbox-studio

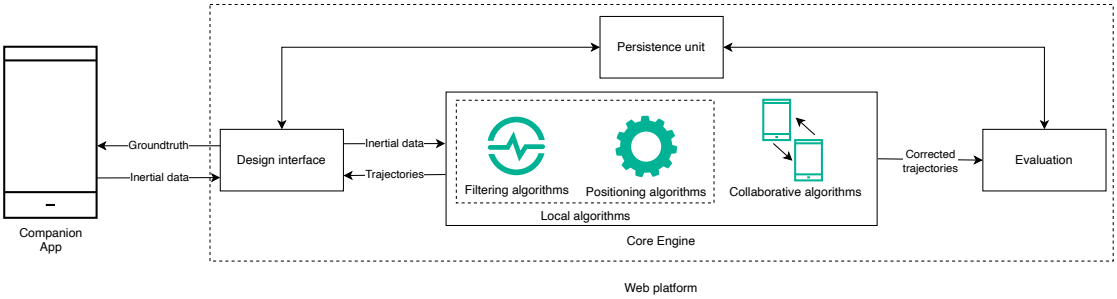

Figure 1: Components of MobiXIM

Running the simulation. The simulation consists of selecting inertial trajectories generated by the participants and reproducing their walking paths in a controlled environment. Before proceeding, we can select a threshold for collaboration. The threshold is expressed in terms of elapsed time either since the last accurate location estimate (typically when entering a building) or since the last collaborative exchange of location with a nearby device. This approach is based on the idea that the accuracy of inertial-based tracking degrades with time. We distinguish two types of thresholds:

- *Lower threshold.* It is a value representing accumulated errors above which a device's estimated location can be updated during an encounter.
- *Upper threshold.* It is a value representing accumulated errors above which a device stops collaborating because its accumulated error is too high and can negatively impact other devices.

For this experiment, we propose a *Midpoint algorithm (MP)* to correct the positioning error when devices' detection ranges overlap. The MP algorithm uses a low-complexity geographic computation technique consisting of drawing a straight line between two devices when their detection ranges overlap and positioning them in the middle of the line. Each device is represented as a point with its estimated location computed by a local algorithm. Figure 2 illustrates two overlapping circles representing the detection ranges of two devices. In this Figure, the lines represent the trajectories of each device illustrated by two distinct colors. The straight lines are the groundtruths, and the dashed lines are the inertial trajectories obtained after running a Pedestrian Dead Reckoning (PDR) algorithm. We can observe how the inertial trajectories diverge over time from the groundtruths due to accumulated errors.

After correcting the errors in positioning, the MP algorithm resets the accumulated errors of the devices. The MP algorithm is described in Algorithm 1 where the *mid point*() function returns a point in the middle of two locations.

After selecting a collaborative algorithm, researchers can decide on the starting time interval and the simulation speed before running the simulation.

The starting time interval indicates the time of departure of each device. A fixed time interval indicates that each device starts moving at a predefined time interval after the previous one. On the other hand, each device could be assigned a predefined timestamp for departure without considering other devices.

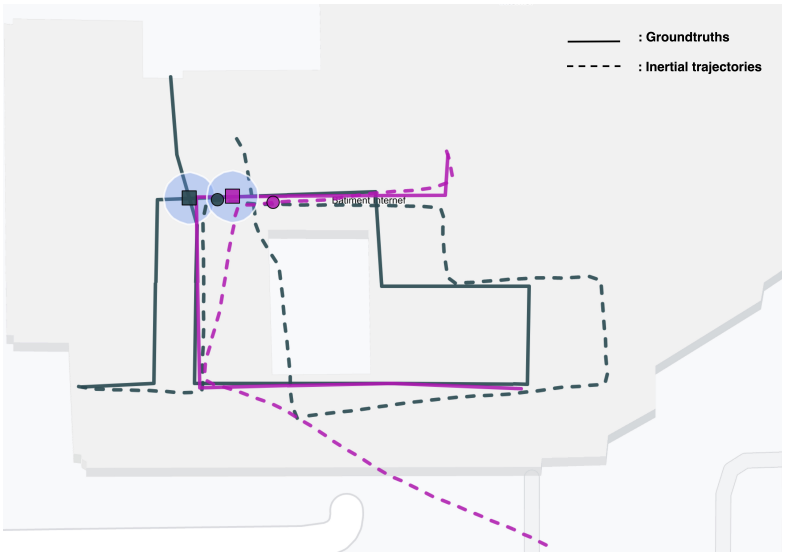

Figure 2: Drift generated by the errors accumulated by the PDR algorithm

Algorithm 1 Midpoint algorithm (MP)

Input: devices: A and B, lower-threshold t
Output: A
$midpoint \leftarrow midpoint(A.location, B.location)$
if $t < A.errors$ **then**
    $A.errors \leftarrow 0$
    $A.location \leftarrow midpoint$
end if
return A

The simulation speed defines the time rate for reproducing the execution of participants' paths. MobiXIM offers the advantage of controlling the speed of execution of the trajectories.

4.2. Evaluation

After running the simulation, MobiXIM outputs a dataset containing three trajectories for each device. These trajectories are as follows.

- *Groundtruth*, designed before the data collection process,
- *Inertial trajectory*, obtained after running the local algorithms,
- *Corrected trajectory*, improved inertial trajectory obtained after running a collaborative algorithm.

For the evaluation, we consider two metrics: the third quartile of point distance error and the Discrete Fréchet Distance (DFD). The third quartile is computed using the ground distance between the measurement and estimated points. Potorti et al. demonstrate that the third quartile

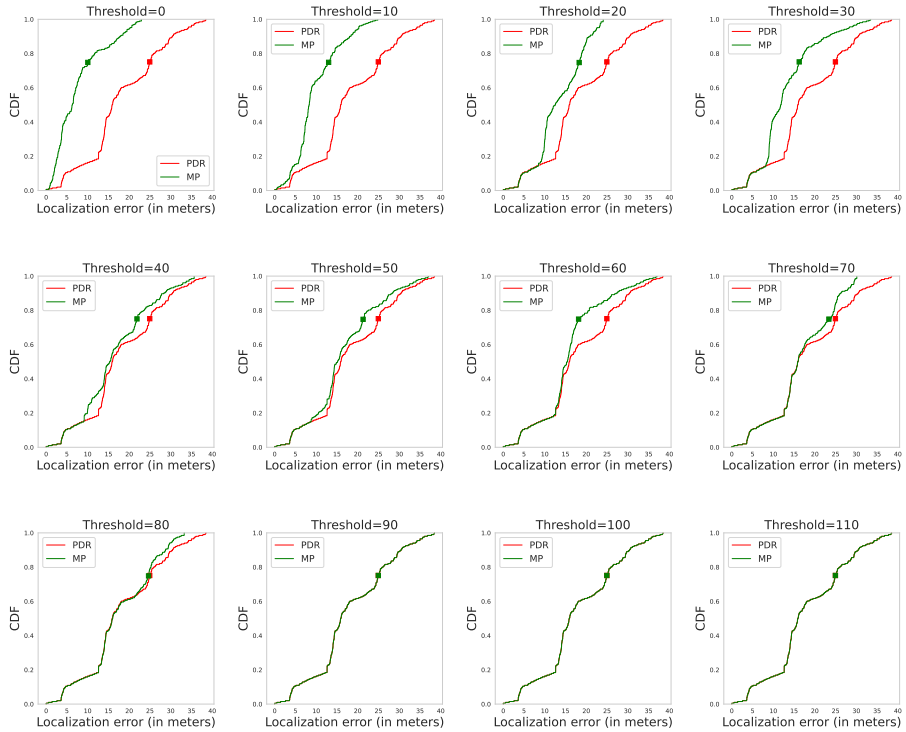

Figure 3: Cumulative Distribution Function of localization errors for a single device

measures more precisely the perceived results of an experiment in comparison with the mean error commonly used in the literature [16]. This metric is extracted from the Cumulative Distribution Function (CDF) of the localization errors. CDF plots illustrated in Figure 3 show a more visual representation of the localization errors of each algorithm for a single device.

On the other hand, the DFD is a common metric for computing the similarities between two trajectories. It measures the minimum separation between points in two trajectories. Therefore, the more the value tends towards 0, the more the trajectories are similar.

The DFD, described in Equation 1 was proposed by Eiter et Heikki [17].

$$dfd(i,j) = \begin{cases} d(P_i, Q_j) & \text{if } i = j = 1 \\ max \begin{cases} d(P_i, Q_j) \\ min \begin{cases} dfd(i-1, j) \\ dfd(i, j-1) \\ dfd(i-1, j-1) \end{cases} \end{cases} & \text{otherwise} \end{cases} \tag{1}$$

In this Equation, P and Q represent trajectories such as $P = \langle p_1, ..., p_m \rangle$ and $Q = \langle q_1, ..., q_n \rangle$,

Device ID	DFD PDR	DFD MP	3rd quartile PDR (m)	3rd quartile MP (m)
1	0.06	0.06	7.22	7.23
2	0.11	0.11	7.70	7.56
3	0.13	0.15	7.43	11.20
4	0.12	0.08	9.30	8.23
5	0.06	0.06	7.03	7.03
6	0.15	0.11	12.30	6.81
7	0.12	0.08	9.95	8.91
8	0.49	0.41	26.85	20.51
9	0.19	0.11	10.86	8.20
10	0.17	0.32	8.88	26.30
11	0.15	0.15	10.14	10.14
12	0.44	0.39	24.96	18.29
13	0.36	0.42	18.31	13.30
14	0.24	0.19	21.58	18.13

Table 1
Performance of the PDR and the MP algorithms

with p_i and q_i representing points on each of the trajectories. $d(P_i, Q_j)$ is the ground distance between points pertaining to their respective trajectories P and Q. The ground distance d between two points $p_l = (\phi_l, \lambda_l)$ and $p_k = (\phi_k, \lambda_k)$ is computed with the haversine formula defined in Equation 2.

$$d = 2R \arcsin \sqrt{\sin^2\left(\frac{\varphi_l - \varphi_k}{2}\right) + \cos(\varphi_k)\cos(\varphi_l)\sin^2\left(\frac{\lambda_l - \lambda_k}{2}\right)} \tag{2}$$

R is a constant representing the radius of Earth.

Figure 3 highlights the localization errors of a single device with different *lower thresholds*. In this figure, the x-axes represent the localization errors in meters; the y-axes are the empirical cumulative distribution functions for each algorithm. The points on the plots are the value obtained for each estimated trajectory with the local PDR algorithm (red lines) and the MP algorithm (green lines). The dots on each plot represent the third quartiles of each algorithm in meters. The thresholds represent *lower-thresholds* as our proposed MP algorithm does not consider an *upper-threshold*.

We observe in this Figure that a very high threshold decreases the accuracy of the MP algorithm. For example, the third quartile of the MP algorithm with a lower threshold set at 0 and no upper threshold is 10.12 m. Whereas with the same percentile, the results go beyond 23 m with a threshold of 70.

However, in the worst cases, the localization errors match the local PDR algorithm as devices struggle to reach a high accumulated error without encountering other devices.

Table 1 shows the performance of the MP algorithm compared to the PDR algorithm. The result shows how the algorithm performs with a *lower threshold* of 60, meaning that each participant runs the MP algorithm if they encounter another participant and their error is above 60. The MP algorithm improves the accuracy of 9 trajectories and the similarity of 7 trajectories regarding their groundtruths.

5. Architecture of MobiXIM

We design MobiXIM as a generic framework that is easy to deploy and extensible for different usages. To achieve these objectives, we built the framework by creating components associated with a generic collaborative ITS.

As we observe in Figure 1, we have main components that are interconnected. These components are described hereafter.

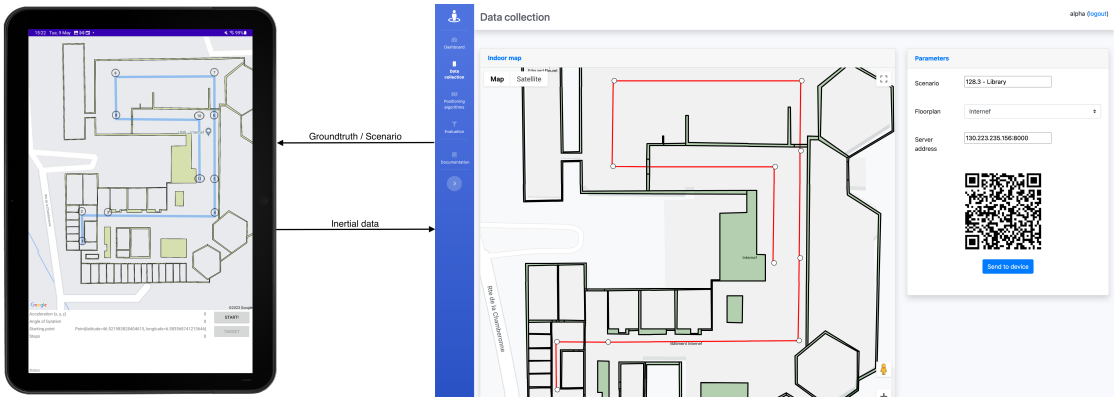

Figure 4: Companion App and Design Interface used for data collection and scenario planning

Design interface. As part of the Web platform hosted on a server, the design interface is an entry point to facilitate the interaction with the other components. It consists of a User Interface (UI) for creating data collection scenarios and running the experiments. The scenarios are made of groundtruths assigned to participants. Figure 4 shows a predefined groundtruth drawn on the design interface. The role of the QR-Code on the right panel of the design interface is to share the groundtruths with participants. Besides its role in designing groundtruths, the design interface is also used to edit some parameters of the proposed algorithms. These parameters can be fine-tuned by researchers to evaluate better how they affect the performance of their algorithms.

The design interface interacts with the other components by using a RESTful API. With the aim to make MobiXIM extensible, such architectural design facilitates the usage of MobiXIM in other applications.

Companion App. As illustrated in Figure 4, the link between the design interface and the companion app is made through a QR-Code that contains a link to the scenario details, the groundtruth, the floorplan, and the server address.

Core Engine. The set of algorithms for estimating a device's location is grouped into a core engine. This engine contains three types of algorithms: filtering, positioning, and collaborative algorithms. We design the core engine with a plugin architecture to make it extensible. Using this architecture, we conceive algorithms as independent entities to facilitate the collaboration

between researchers. The proposed algorithms, defined as plugins, must follow a predefined structure given as follows:

- *Name.* A plugin is identified by its name.
- *Slug.* It is used as a unique identifier for each plugin.
- *Category.* Plugins are grouped into three categories: *Filtering algorithm*, *Positioning algorithm* and *Collaborative algorithm.*
- *Display_name.* It is used in the design platform to identify plugins with a human-readable format.

The plugins should implement all the methods of predefined interfaces. For example, plugins for filtering algorithms should have at least a method called *get_filtered_data* that takes raw acceleration data as input and returns the filtered data.

Persistence Unit (PU). Also defined as a storage unit, it stores raw inertial data, groundtruth trajectories, and parameters obtained from the design interface. To facilitate the export of the results, the PU stores the links to the inertial and corrected trajectories generated by the local and collaborative algorithms.

6. Related work

In the literature, many attempts have been made to standardize the process of devising ITS. The heterogeneity of tracking technologies mainly justifies this. Poulose et al. take advantage of this heterogeneity to propose a sensor-fusion framework using smartphone sensors. Their proposed framework combines fingerprinting and trilateration using WiFi and PDR [18]. They present a flow chart of sequences for data collection and steps for building positioning algorithms. Their framework highly emphasizes fingerprinting, which is a cumbersome process that requires some equipment and generates a significant cost for collecting data.

Potorti et al. propose the EvAAL framework to evaluate ITS [16]. It is a well-proven framework that has been used for the past editions of the IPIN competitions. It offers tools for evaluating numerous algorithms on various use cases (pedestrian versus robotic navigation, real-time positioning, etc.). The EvAAL framework establishes several rules for evaluating positioning algorithms in a standardized way, such as the movement of participants, the space and time to execute scenarios, and the error metric.

In comparison with the EvAAL framework, MobiXIM offers a range of tools that goes beyond the evaluation of ITS. MobiXIM is a ready-to-use framework that allows researchers to devise, evaluate and fine-tune their algorithms. This is done by setting guidelines and providing with tools allowing researchers to control their experiments. By adding support for mapping using a GeoJSON format, we enable researchers to test their algorithms by considering the characteristics of their deployment environment. This way, researchers can devise algorithms integrating positioning techniques such as map matching, particle filtering, etc.

Recently, Van de Wynckel and Signer proposed the OpenHPS Framework for devising hybrid positioning systems composed of wireless and inertial-based technologies [19].

This framework addresses the lack of control in the devising process of ITS. It decomposes each step in modular layers of abstractions, such as symbolic spaces or high-level API endpoints. In their paper, they demonstrate the usage of their framework by developing an ITS with fingerprinting as a positioning algorithm. With the symbolic space abstraction, they represent rooms, corridors, lobbies and toilets as GeoJSON polygonal features.

Compared to the above-cited frameworks, the novelty of MobiXIM is the focus on the collaborative aspect of indoor tracking. By proposing this framework, we aim to stimulate researchers' interest in collaborative ITS and, facilitate the development of new collaborative algorithms.

7. Discussion & Future work

Proposing a framework that considers the most important aspects of an ITS is a difficult task, given all the different techniques used for Indoor Tracking. This paper proposes a framework that addresses the essential points needed to design an ITS. We extend the state of the art by adding a collaborative aspect that is missing in frameworks proposed in the literature.

One of the reasons for integrating a collaborative aspect to our proposed framework comes from the growing interest in solutions that use Device-to-Device (D2D) communications to improve positioning estimates. These solutions use an extensive range of technologies and tracking algorithms such as BLE with signals lateration [20], WiFi fingerprinting [21], PDR with particle filtering [12] or positioning with Visible Light [22].

Beyond the differences in technologies and tracking algorithms, these solutions use a similar approach to what we describe in this paper. Indeed, they integrate a local algorithm for computing local estimates, estimate the distance between users and use this data to improve the accuracy of a baseline algorithm. For example, Qi et al. proposed a collaborative ITS using WiFi RSS fingerprints [23]. Their paper mostly focuses on accurately estimating the physical distance between users using nearest-neighbour, random forest and a multilayer perceptron on the WiFi RSS fingerprints collected in advance. For devising such solutions, MobiXIM offers a high level of abstraction by allowing users to integrate existing trajectories generated by any positioning technique. In addition, our proposed framework allows users to define their own method for estimating inter-user distances and sending these data to a collaborative algorithm.

To facilitate the large adoption of our proposed framework in the future, it would be relevant to integrate some aspects commonly found in the literature. One of these aspects is the integration of a Cartesian spatial representation system. In this paper, we only consider a geometric spatial representation system. This design choice aims at facilitating the interconnection with existing outdoor positioning systems. As most location-based services rely on GNSS, we believe the research community and the industry will easily transition between both systems if they share the same spatial representation system.

Another major point that is debatable is the data collection procedure presented in this paper. We are aware that placing landmarks on a map could introduce errors, but we believe that it remains an easy way of enabling researchers to test their algorithm quickly. For those concerned about centimetre-level accuracy, MobiXIM offers an API that allows researchers to use their own datasets without going through the web platform.

Potential improvements

The MP algorithm presented in this paper serves as a mockup algorithm to illustrate how a collaborative algorithm works. This algorithm can be improved by considering other parameters, such as the upper threshold. The MP algorithm presented in this paper considers that errors accumulate over time and use an error counter that increments at a fixed time interval. In the literature, another way of doing it is through a Kalman gain which offers the tools for representing the reliability of a system estimates [18]. As this paper aims to present a generic framework for collaborative ITS, we are working on proposing more robust collaborative algorithms developed using our proposed framework.

8. Conclusion

This paper explores the stages of devising, evaluating, and fine-tuning collaborative ITS. After highlighting the laborious process of executing these stages, we propose a framework to collect real-life data, to combine them, and reproduce them in a controlled environment. Using a plugin architecture, we enable researchers to take advantage of the code reuse principle that boosted the development of complex pieces of software. Our goal is to ensure the extensibility of the framework and to facilitate its adoption by the research community. In future work, we will extend the framework to consider wireless-based ITS. We will also propose more complex collaborative algorithms built with MobiXIM.

References

[1] Y. Sartayeva, H. C. Chan, A survey on indoor positioning security and privacy, Computers & Security 131 (2023) 103293. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167404823002031. doi:10.1016/j.cose.2023.103293.

[2] M. Ramachandran, A. Veeraraghavan, R. Chellappa, CHAPTER 5 - Video Stabilization and Mosaicing, in: A. Bovik (Ed.), The Essential Guide to Video Processing, Academic Press, Boston, 2009, pp. 109–140. URL: https://www.sciencedirect.com/science/article/pii/B9780123744562000062. doi:10.1016/B978-0-12-374456-2.00006-2.

[3] Y.-T. Li, G. Chen, M.-T. Sun, An Indoor Collaborative Pedestrian Dead Reckoning System, in: 2013 42nd International Conference on Parallel Processing, IEEE, Lyon, France, 2013, pp. 923–930. URL: http://ieeexplore.ieee.org/document/6687434/. doi:10.1109/ICPP.2013.110.

[4] P. Strömbäck, J. Rantakokko, S.-L. Wirkander, M. Alexandersson, K. Fors, I. Skog, P. Händel, Foot-Mounted Inertial Navigation and Cooperative Sensor Fusion for Indoor Positioning, 2014.

[5] M. J. Abadi, L. Luceri, M. Hassan, C. T. Chou, M. Nicoli, A collaborative approach to heading estimation for smartphone-based PDR indoor localisation, in: 2014 International conference on indoor positioning and indoor navigation (IPIN), IEEE, 2014, pp. 554–563.

[6] R. Liu, C. Yuen, T.-N. Do, M. Zhang, Y. L. Guan, U.-X. Tan, Cooperative positioning for

emergency responders using self IMU and peer-to-peer radios measurements, Information Fusion 56 (2020) 93–102. Publisher: Elsevier.

[7] K. Kloch, P. Lukowicz, C. Fischer, Collaborative PDR localisation with mobile phones, in: 2011 15th annual international symposium on wearable computers, IEEE, 2011, pp. 37–40.

[8] V. Dehghanian, M. Lowe, RSS-INS integration for cooperative indoor positioning, in: 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), IEEE, Alcala de Henares, Spain, 2016, pp. 1–7. URL: http://ieeexplore.ieee.org/document/7743603/. doi:10.1109/IPIN.2016.7743603.

[9] P. Agis, K.-K. Wong, Z. Zheng, Y. Zhang, Cooperative localisation with hybrid inertial navigation system/pedestrian dead reckoning tracking for GPS-denied environments, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM, Pisa Italy, 2016, pp. 675–681. URL: https://dl.acm.org/doi/10.1145/2851613.2851850. doi:10.1145/2851613.2851850.

[10] T. He, Q. Niu, N. Liu, GC-Loc: A Graph Attention Based Framework for Collaborative Indoor Localization Using Infrastructure-free Signals, Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 6 (2022) 1–27. URL: https://dl.acm.org/doi/10.1145/3569495. doi:10.1145/3569495.

[11] P. Pascacio, S. Casteleyn, J. Torres-Sospedra, E. S. Lohan, J. Nurmi, Collaborative Indoor Positioning Systems: A Systematic Review, Sensors (Basel, Switzerland) 21 (2021). doi:10.3390/s21031002.

[12] K. Kageyama, T. Miyazaki, Y. Sugaya, S. Omachi, Collaborative Indoor Positioning by Localization Comparison at an Encounter Position, Applied Sciences 13 (2023) 6962. URL: https://www.mdpi.com/2076-3417/13/12/6962. doi:10.3390/app13126962, number: 12 Publisher: Multidisciplinary Digital Publishing Institute.

[13] Y. Noh, H. Yamaguchi, U. Lee, Infrastructure-Free Collaborative Indoor Positioning Scheme for Time-Critical Team Operations, IEEE Transactions on Systems, Man, and Cybernetics: Systems 48 (2018) 418–432. doi:10.1109/TSMC.2016.2615652, conference Name: IEEE Transactions on Systems, Man, and Cybernetics: Systems.

[14] A. El-Naggar, A. Wassal, K. Sharaf, Indoor Positioning Using WiFi RSSI Trilateration and INS Sensor Fusion System Simulation, in: Proceedings of the 2019 2nd International Conference on Sensors, Signal and Image Processing, SSIP 2019, Association for Computing Machinery, New York, NY, USA, 2019, pp. 21–26. URL: https://doi.org/10.1145/3365245.3365261. doi:10.1145/3365245.3365261.

[15] J.-O. Nilsson, D. Zachariah, I. Skog, P. Händel, Cooperative localization by dual foot-mounted inertial sensors and inter-agent ranging, EURASIP Journal on Advances in Signal Processing 2013 (2013) 1–17. Publisher: SpringerOpen.

[16] F. Potortì, S. Park, A. R. Jimenez Ruiz, P. Barsocchi, M. Girolami, A. Crivello, S. Y. Lee, J. H. Lim, J. Torres-Sospedra, F. Seco, others, Comparing the performance of indoor localization systems through the EvAAL framework, Sensors 17 (2017) 2327. Publisher: MDPI.

[17] T. Eiter, H. Mannila, Computing discrete fréchet distance, Technical Report, Tech. Report CD-TR 94/64, Information Systems Department, Technical University of Vienna, 1994.

[18] A. Poulose, O. S. Eyobu, D. S. Han, An Indoor Position-Estimation Algorithm Using Smartphone IMU Sensor Data, IEEE Access 7 (2019) 11165–11177. doi:10.1109/ACCESS.2019.2891942, conference Name: IEEE Access.

[19] M. Van de Wynckel, B. Signer, Indoor positioning using the OpenHPS framework, in: 2021 international conference on indoor positioning and indoor navigation (IPIN), IEEE, 2021, pp. 1–8.

[20] P. Pascacio, J. Torres–Sospedra, S. Casteleyn, E. S. Lohan, A Collaborative Approach Using Neural Networks for BLE-RSS Lateration-Based Indoor Positioning, in: 2022 International Joint Conference on Neural Networks (IJCNN), 2022, pp. 01–09. doi:10.1109/IJCNN55064.2022.9892484, iSSN: 2161-4407.

[21] C. Zhou, B. Wang, Y. Mo, Z. Zeng, MOCLoc: Emerging Online Collaborative Localization Enhanced by Multidimensional Scaling, IEEE Transactions on Emerging Topics in Computational Intelligence 6 (2022) 751–761. doi:10.1109/TETCI.2021.3110260, conference Name: IEEE Transactions on Emerging Topics in Computational Intelligence.

[22] X. Liu, L. Guo, H. Yang, X. Wei, Visible Light Positioning Based on Collaborative LEDs and Edge Computing, IEEE Transactions on Computational Social Systems 9 (2022) 324–335. doi:10.1109/TCSS.2021.3109631, conference Name: IEEE Transactions on Computational Social Systems.

[23] T. Qi, C. Zhou, G. Ouyang, B. Wang, Multiuser collaborative localization based on inter-user distance estimation using wi-fi RSS fingerprints, in: 2022 18th international conference on mobility, sensing and networking (MSN), IEEE, 2022, pp. 679–686.