2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 29 Nov. – 2 Dec. 2021, Lloret de Mar, Spain

Dioptra – A Data Generation Application for Indoor Positioning Systems

Cristiano Pendão*, Ivo Silva*, Adriano Moreira*, Joaquín Torres-Sospedra†
*Algoritmi Research Center, University of Minho - Portugal
†UBIK Geospatial Solutions S.L., Castellón, Spain

***Abstract*—Indoor Positioning Systems (IPSs) based on different approaches and technologies have been proposed to support localization and navigation applications in indoor environments. The fair benchmarking and comparison of these IPSs is a difficult task since each IPS is usually evaluated in very specific and controlled conditions and using private data sets, not allowing reproducibility and direct comparison between the reported results and other competing solutions. In addition, testing and evaluating an IPS in the real world is difficult and time-consuming, especially when considering evaluation in multiple environments and conditions. To enhance IPS assessment, we propose Dioptra, an open access and user-friendly application to support research, development and evaluation of IPSs through simulation. To the best of our knowledge, Dioptra is the first application specially developed to generate synthetic datasets to promote reproducibility and fair benchmarking between IPSs.**

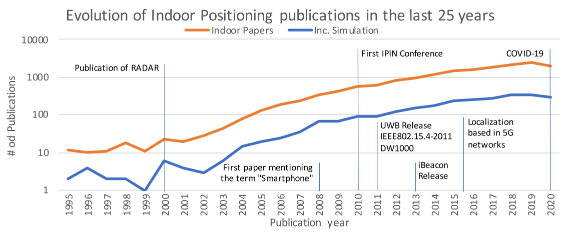
***Index Terms*—Simulator; Indoor Navigation; Evaluation**

I. Introduction

The position estimation of a subject on land and seas has been of utmost importance since ancient civilizations. Astrological devices such as the Dioptra, the Planisphere or the Astrolab were conceived not only for identifying celestial bodies, but also to determine the current user's location, especially on seas where reference landmarks are missing. Nowadays, GNSS systems tackle localization with a constellation of satellites. However, they show severe problems operating indoors and none of the available technologies stands out as they present different features (e.g., cost, accuracy, reliability, scalability, latency and power consumption) that depend on the scenario.

Research on indoor positioning has boosted up in the last 25 years. Fig. 1 shows the works indexed in *Scopus* where the terms related to indoor positioning appear on the title, keywords or abstract. Publications increased from 23 papers in 2000, when RADAR was proposed [1], to almost 2500 papers in 2019. In the last two decades, Indoor Positioning has been consolidated as a solid research area as many new positioning technologies have emerged (based on IEEE 802.11 Wireless LAN (Wi-Fi), Ultra-wide band (UWB), Bluetooth Low Energy (BLE), Visible Light Communications (VLC), etc.), the computational resources have improved (either locally or in the cloud), and new portable computing devices (Smartphones, Raspberry Pi, Arduino) have broaden new applications.

Corresponding Author: Cristiano Pendão (`cpendao@dsi.uminho.pt`)
This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020 and the PhD fellowship PD/BD/137401/2018. J. Torres-Sospedra acknowledges funding from Programa Torres Quevedo (PTQ2018-009981).



```
Q1:TITLE-ABS-KEY(( "indoor posit*" OR "indoor loca*" OR "indoor navig*" ))
Q2:TITLE-ABS-KEY(( "indoor posit*" OR "indoor loca*" OR "indoor navig*" ) AND (
"simulated" OR "simulation" )))
```

Fig. 1. Retrieved documents from *Scopus* using the search queries.

Evaluating the performance of Indoor Position Systems (IPSs) in the real-world is a difficult and time-consuming task. Generally, IPSs are evaluated under very controlled conditions being the obtained performance results not directly comparable to those of other competing IPSs for the same application domain. Use of simulated data is a common practice in the research field (15% of works in Fig. 1).

Despite simulated data do not fully mimic the conditions of real scenarios, it is useful when the hardware elements are not available (e.g. assessing positioning with 5G [2]) or in exploring IPS limitations (e.g. a very large number of targets [3]) without extensive manual labour data collection. Simulated data has also been used in preliminary steps to setup optimal parameter values [4]) and IPS validation before an empirical assessment with real data [4], [5]. However, the review of very recent publications [4]–[8] shows that none of them relies on a single generic stand-alone simulation tool. Instead, simulations are mainly based on own MATLAB signal model implementations [2], [6], commercial network simulators, or simulated experiments through external datasets [8].

Another aspect making it difficult to benchmark different IPSs is that most authors do not share their datasets, whether real-world collected or generated through simulation. This practice undermines the reproducibility of the proposed systems as other researchers cannot replicate others' works to compare with theirs. Fortunately, an increasing number of researchers are sharing their datasets in open access through platforms such as Zenodo and Crawdad. While donated datasets are beneficial, most of them suffer from the limitations identified above for real-world evaluations.

This paper addresses the challenge of evaluating IPSs by proposing an open access tool, Dioptra, that researchers can use to generate synthetic datasets and evaluate their IPS:

- in similar conditions to other previous systems, therefore enhancing reproducibility and fair benchmarking;
- in large spaces, in contrast to evaluations in small labs;
- in a variety of different conditions, such as different types of buildings and rooms;
- by considering multiple sets of parameters' values;
- for scalability, e.g. with several simultaneous users.

Previous attempts to provide simulation tools have either focused only on signal propagation prediction [9], [10], time differences in communication protocols [11], or magnetic field positioning [7]. Other works just rely on simple models to synthetically generate data without presenting a dedicated simulation tool, namely Log-Distance Path Loss (LDPL) model [12]–[15], ray-tracing [9], [16] or radiosity [17], [18].

Dioptra is designed to support multiple types of actors (pedestrians, robots, etc.), sensors, motion models, and propagation models. Moreover, the Dioptra Project can contribute to increasing the number of datasets shared by the community, since being synthetic: First, there is no effort related to preparing a real-world setup, collecting and preparing the data to be shared. Second, there are no data privacy issues or share restrictions related to funding entities. The actual datasets do not need to be shared, only the configurations since anyone can replicate the dataset using the same configuration file. It is important to note that Dioptra, in its current version, is designed to generate synthetic data to test and evaluate IPSs, not to simulate IPSs.

To the best of our knowledge, this is the first open access tool developed specifically for generating synthetic data to be used for IPS development and evaluation, with a modular architecture allowing future extensions.

II. Dioptra Description

Dioptra is a cross-platform application or tool designed to simplify the process of generating synthetic data that can be used, e.g., to support the development and testing of an IPS or to fairly benchmark several IPSs in the same conditions. It was designed with three main practical aspects in mind:

- **Simple and easy to use:** Dioptra is coded in Java, therefore it can run on multiple platforms, such as Windows, Linux or MacOS. A Graphical User Interface (GUI) was developed to allow users to follow the progress and control most aspects of a work session (simulation). Work sessions are based on configuration files that provide a simple way to reproduce results and can be easily adapted to implement other use cases. Dioptra already includes several examples ready to use, and more will become available in the website open repository [19].
- **Allow easy extension and integration:** Dioptra was built from the ground up to simplify the integration of new modules and features. Dioptra's framework and open API allow new modules, developed by our team or by the community, to be easily plugged into the application. The new modules can be anything from propagation models, to new implementations of sensors or transmitters. To ensure abstraction and compatibility between new modules and the Dioptra engine, the new modules must follow the defined Java Interfaces[1]. Dioptra will become open-source soon.
- **Use of standard or well-known formats:** By using standards for input/output, e.g. OpenStreetMap (OSM) for the floor plan, or the IPIN logfiles [20] as output, the integration in the users' current workflows is simplified.

A. Main Architecture

The main architecture of Dioptra is shown in Fig. 2, where the current modules and their interactions are depicted.

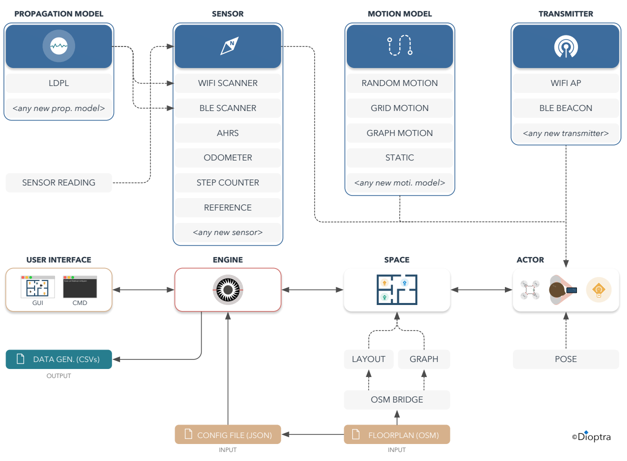


Fig. 2. Dioptra Main Architecture

The Engine is the module responsible for the main setup, integration and control operations in a work session.

The Space includes the layout and navigation topologies (that can be used in the graph-based motion model) that are loaded and created by the Engine, based on the information from the input floor plan file (OSM).

Actors are created by the Engine, based on the input configuration file (JSON), and added to the Space. The Actor module allows to create different types of actors, currently supporting, e.g., the representation of Fixed Transmitters, Pedestrians, Robots, Industrial Machines, Drones, among others.

Motion Models define the movement behaviour that an Actor will follow during a work session. A motion model must be configured for each actor. Different parameters can be configured for each type of motion model, and each instance of a Motion Model, that is associated with an Actor, can have different values for the available parameters.

The Sensor module allows to implement different types of sensors, that can be added to an Actor. Each type of sensor has its own set of parameters that can be configured. Sensors such as Wi-Fi and BLE Scanners measure virtual signals from the transmitters based on a Propagation Model, that can also be configured. It is important to note the difference between a Fixed Transmitter - Actor, and a Transmitter (e.g. Wi-Fi AP, BLE Beacon). Transmitters can be added to any actor,

[1]https://docs.oracle.com/javase/tutorial/java/concepts/interface.html

therefore one can create a Fixed Transmitter - Actor, that can include multiple Transmitter instances. This flexibility allows, e.g., to create Fixed Transmitters that include transmitters in different frequency bands or different radio technologies.

The User Interface allows the interaction with the application using the GUI or the Command (CMD) Line. In both interfaces, users select/specify a JSON-based configuration file as input in order to setup a work session. The path to the floor plan file is configured in the JSON configuration file. The floor plan file (OpenStreetMap format) is loaded by the Engine using the OSM Bridge Module, creating a new Space. Additional details (warnings, errors, info) are logged to a file that can be accessed by the user.

At the end of a work session, the data generated by each Actor will be saved as an independent output file. The space layout and the transmitters' positions can also be exported.

B. Actors

An actor represents a fixed or moving entity, such as a fixed device or a moving pedestrian, vehicle or robot. The current supported Actor types are: Pedestrian, Smartphone, Robot, Industrial Machine, Drone, Probe (e.g. Fixed Scanner) and Fixed Transmitter. The predefined types of Actors improve the visualisation and allow an easy distinction between multiple devices in the GUI, since each type of device is represented by a different representative image.

An actor can be equipped with a number of transmitters, such as Wi-Fi Access Points (APs) of BLE beacons, and several sensors collecting data. Actors are configured in the session configuration file. Users can use the presets provided, and easily adapt them. Adding or removing sensors or transmitters is simple and the configuration snippets provided can be used as example. Multiple instances of any sensor or transmitter can be added to any Actor.

For the radio-based sensors, multiple radio propagation models can be used. Each transmitter and sensor can be configured independently, e.g. by setting the transmit ID and advertising frequency. Similarly, sensors can be configured by setting, for instance, the sampling frequency and noise level.

Moving actors can adopt one among a set of alternative motion models, making the tool useful to simultaneously generate data for several types of entities, such as pedestrians, robots or vehicles. A special type of sensor can be used to record ground truth. Each motion model includes configurable parameters that can be tuned by the user.

C. Motion Models

The motion behaviour of each actor is governed by one among the several motion models described in this section (see Fig. 3). The motion model defines how an actor moves, including periods of immobility, and how it avoids obstacles.

1) Random Walk: The actor moves freely across the navigable space, with variable speed and avoiding walls and obstacles. The initial position of the actor is set randomly at

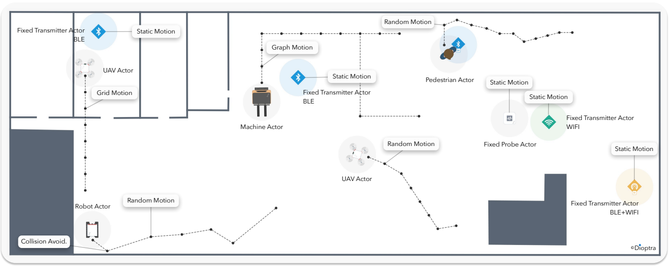


Fig. 3. Motion Models (Dioptra 1.0 - Light Mode)

the beginning of the simulation. From this initial position, the actor moves to a new position following the model:

$$px_i = px_{i-1} + v_i \times \Delta t \times cos(\phi_i) \tag{1}$$

$$py_i = py_{i-1} + v_i \times \Delta t \times sin(\phi_i) \tag{2}$$

$$pz_i = pz_{i-1} \tag{3}$$

where $(px_{i-1}, py_{i-1}, pz_{i-1})$ is the previous position, $v_i = U(0, v_{max})$ is the current velocity modelled as a random variable uniformly distributed between 0 and v_{max}, $\phi_i = \phi_{i-1} + U(-\theta, \theta)$ is the current heading, and Δt is the time step. The value of θ defines how sharp can a turn be.

Collisions with walls and obstacles are detected at each iteration of the model. If the new position (calculated from eqs.1–3) end in a non-navigable area, that position is discarded and a new one is calculated after adding a random increment to the heading: $\phi_i \leftarrow \phi_i + U(-\frac{\pi}{2}, \frac{\pi}{2})$; the process is repeated until the new position does not collide with an obstacle.

2) Grid: In the grid motion model, a matrix of 2D points is generated for the entire navigable space. The Euclidean distance between adjacent points in the grid is a model parameter, as is the height of the actor (z coordinate). No points are generated within walls or obstacles. An actor following the grid motion model "jumps" from one point to another, stays there for a certain amount of time (configurable) and then jumps to the next point. This motion model is particularly useful e.g. to create radio maps for fingerprinting-based IPSs.

3) Graph-based: The model of the space might include its topology described by one or more undirected graphs (see Section II-G). The motion of an actor adopting the Graph-based motion model is restricted to the edges connecting the nodes in the graph (the graph to use is a model parameter). The initial position is selected randomly from the set of all nodes in the graph. The actor then follows a trajectory over a straight line (graph edge) connecting the source node to a destination node. The destination node is randomly selected from the set of nodes directly connected to the source node, excluding the previous source node if there are other options (to avoid going back to the previous node). While traveling from source to destination node, the actor travels at a constant speed. The speed value is randomly selected for each edge: $v = U(v_{min}, v_{max})$, with $v_{min} = 0.5\,\mathrm{m\,s^{-1}}$ and v_{max} being a model parameter. The minimum speed is not allowed to be zero to prevent an actor to get locked in a node.

4) Static: The static motion model is used for actors that should remain static, as is the case of actors of the type "Fixed Transmitter" (an actor equipped with transmitters, e.g. Wi-Fi AP). The parameters of this model are the geometric coordinates (x, y, z).

D. Sensors

One or more sensors can be added to an actor. Radio interfaces (e.g. Wi-Fi, BLE) are treated as radio scanners and implemented as sensors, as they are used to measure the received radio signals. The current version of Dioptra supports sensors to measure displacement, orientation and radio data. The configuration parameters of each sensor allow to simulate different grades of sensors according to the characteristics of the sensor. All sensors share the status (enabled or disabled) parameter. When disabled, the sensor is not considered in the simulation. This allows users to easily enable/disable sensors without removing them from the configuration file.

- **Odometer:** The odometer is a sensor that measures the displacement of a moving actor between two time instants. The current implementation has a configurable sampling rate and noise level. The noise is modelled as a Gaussian distribution Gaussian random process with null mean and configurable variance.
- **Step-Counter:** This sensor emulates a step-counter, detecting steps of a moving pedestrian and also providing an estimated step length for each step. The step length, the step missing probability, and the noise level are configurable. The noise is modelled as a Gaussian random process with null mean and configurable variance.
- **AHRS:** This sensor emulates the output of a soft-sensor that estimates the pose of an actor by fusing raw data from gyroscopes, accelerometers and magnetometers (a typical Inertial Measurement Unit (IMU)). The current implementation only provides the heading (yaw) of the actor. The sampling rate, as well as noise and drift levels can be configured.
- **Wi-Fi Scanner:** This sensor emulates a Wi-Fi radio interface/scanner and provides signal power measurements of the nearby Wi-Fi Transmitters. The sampling rate, noise and sensitivity can be configured. Each Wi-Fi sensor can use a configurable radio propagation model to estimate the signal level received from each transmitter. The advantage of using a configurable propagation model per sensor, instead of a single model for all sensors, is that this way it is possible to generate synthetic data considering different propagation models in the same session (see Section II-F).
- **BLE Scanner:** The BLE scanner has a structure similar to that of the Wi-Fi scanner. It provides raw measurements of the radio signal received from BLE beacons.
- **Reference:** The reference sensor is a virtual sensor, in the sense that it does not emulate a real sensor. Instead, this sensor simulates the process of obtaining highly accurate ground truth. In this case, this sensor provides a zero error ground truth pose, including a 3D position and orientation (roll, pitch and yaw) for each device at a configurable sampling rate. In the current implementation, only the yaw component of the orientation is provided.

E. Transmitters

Transmitters are devices that periodically broadcast a beacon. Transmitters can be attached to any actor, including fixed and mobile actors. The current implementation of Dioptra supports two types of radio-based transmitters: Wi-Fi APs and BLE Beacons.

1) Wi-Fi Access Points: The following parameters can be used to configure a Wi-Fi AP: transmitter_id; status (enabled or disabled); BSSID; radio channel; transmit power (rss_0; see Section II-F); SSID.

2) BLE Beacons: The following parameters can be used to configure a BLE Beacon: transmitter_id; status (enabled or disabled); advertising_interval; unique_id; deployment_id; major_id; minor_id; transmit power.

F. Radio Propagation Models

Radio propagation models connect transmitters (Wi-Fi Access Points and BLE Beacons) to the corresponding sensors (Wi-Fi and BLE scanners). They define how radio signals propagate through the physical space.

In a typical indoor positioning system, a radio scanner (sensor) performs scans periodically to detect incoming radio signals and measure some of their characteristics. As an example, the output of a Wi-Fi scan includes a list of detected APs and corresponding identification (BSSID), signal levels (RSS), frequency channel and network name (SSID). Fingerprinting or multi-lateration based methods can use these RSS measurements (scans) to estimate the position.

The radio propagation model defines the statistics of the radio signal measured by a scanner. The current implementation of Dioptra includes one single radio propagation model based on the LDPL model, however it is ready to include other propagation models and positioning techniques (e.g., AoA).

1) Enhanced LDPL radio propagation model: This model considers three characteristics of the propagation of radio signals indoors: (i) the attenuation as a function of the distance between transmitter and receiver; (ii) the additional attenuation introduced by obstacles (taking into account different materials) in the direct path between transmitter and receiver; (iii) the probability of a transmitter to be detected by a receiver.

The first two of these characteristics are modelled using the LDPL model [21], extended with a deterministic component to account for known obstacles between transmitters and receivers [12]. With this extended model, the strength of the received radio signal is given by:

$$rss = rss_0 - (10 \cdot \eta \cdot \log_{10}(d/d_0) + \mathrm{A} + \epsilon) \qquad (4)$$

where rss_0 is the signal strength measured at a distance d_0 from the transmitter, η is the path-loss factor (depends on the indoor environment characteristics) and $\epsilon = N(0, \sigma)$ is a Gaussian random variable with null mean and standard deviation σ, used to represent the channel noise and fading. The A component accounts for the additional attenuation introduced by known obstacles between transmitter and receiver. The η component of the LDPL model already tries to account for the different characteristics of the indoor radio channel. However, once the value of this parameter is set, it will be used independently of the transmitter and receiver positions. The A component takes advantage of the existing space model (see Section II-G) to estimate the true attenuation due to known obstacles made of known materials, and is calculated as:

$$\mathrm{A} = \sum_{i=1}^{k} a_i \tag{5}$$

where a_i is the attenuation introduced by each one of the obstacles intersecting a line that connects the transmitter and receiver (see Fig. 4).

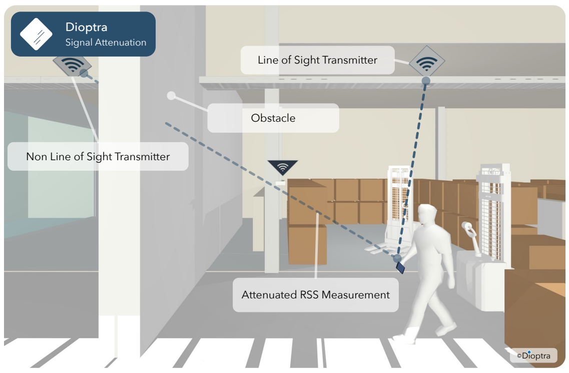

Dioptra
Signal Attenuation
Line of Sight Transmitter
Obstacle
Non Line of Sight Transmitter
Attenuated RSS Measurement
©Dioptra

Fig. 4. RSS Measurements: Obstacles Attenuation

As has been reported in some empirical studies, a beacon from a transmitter is not always detected by a receiver. To model this behaviour, we define a function representing the probability of receiving a beacon as:

$$f(rss) = \begin{cases} false, & rss < g(rss) \\ true, & otherwise \end{cases} \tag{6}$$

Function $g(rss)$ defines the minimum value of rss required for a beacon to be detected, and has two alternative implementations: $g_0(rss) = rss_{min}$ and $g_1(rss) = \rho \times rss_{min}$, where rss_{min} is the receiver sensitivity and $\rho = U(0,1)$ is a random variable uniformly distributed in the interval $(0,1)$. With function g_0, all beacons with rss above the receiver sensitivity are detected. Function g_1 gives weaker beacons a lower probability of being detected. Other functions might be implemented in the future.

The value for rss_0 is defined for each radio transmitter. The values for rss_{min}, σ and $g(rss)$ are defined for each receiver. The attenuation values associated to each obstacle are derived from the space model.

G. Space Models - Indoor Maps

Currently, there are some works trying to document and discuss the differences between different indoor map formats and standards [22], [23], however no universal indoor map standard has been adopted yet. The main reason is because when developing an application for indoor positioning, one considers the requirements and decides upon an indoor format that better fits that purpose. The OSM[2] format and the IndoorGML[2] are some of the most well-known indoor map standards, but there are also proprietary formats developed by several companies, namely, Google Maps Indoor, Nokia's Here Maps, Apple's Indoor Maps and ArcGIS Indoor.

OSM is a collaborative and free project that allows users to create maps. Users are capable of assigning key-value pairs to points (*nodes*), lines and polygons (*ways*) and groups of elements (*relations*). Key-value pairs allow to associate features to places, for example, the name of the place, type of amenity or type of building. Even though OSM maps were initially created for outdoor scenarios, they can also be used to represent indoor maps. There are several projects focused on developing proposals for the mapping of indoor spaces[2].

We choose OSM as the first supported format to represent indoor maps. OSM files are loaded by Dioptra's OSM Bridge module, therefore other formats can be easily supported in the future. OSM was our first option because it is vectorial, it includes information regarding indoor elements, it can be easily edited using an OSM editor[2], and since it is represented in XML format, it can be easily parsed and used by computer programs. One disadvantage of OSM is that it has some limitations in representing 3D elements. In contrast with IndoorGML, which has a lack of editor programs, there are several tools to edit OSM files, hence it is a better solution from an end-user standpoint.

We specified a custom format for the indoor elements because the aim of Dioptra is to enable the simulation of an indoor scenario with several indoor features, such as, walls, obstacles, the building materials of each element and the navigation topologies that describe the motion of different actors. The indoor features supported by the tool are:

- **Building**: A polygon is used to represent the exterior walls of the building. The wall width, height of the building, and building material can be configured.
- **Wall**: A line or set of lines may be used to represent a wall, which can be characterized by the height, width, and material (e.g. brick, glass or concrete).
- **Obstacle**: Defined by a polygon with an associated height and material.
- **Graph**: A set of nodes connected by edges is used to represent the undirected graph of the navigation topology. Each graph is uniquely identified with an id necessary to distinguish between different graphs.

Currently, the tool supports single-building, single-floor configurations, where devices can be placed in a 3D space, and the generated data is also 3D. The JOSM editor was used to create the floor plans provided with the tool. Although the tool comes with 4 space models that represent distinct scenarios (2 factories, 1 floor of an office building, and 1 floor of a

[2]https://wiki.openstreetmap.org/wiki/Indoor_Mapping; http://www.indoorgml.net; https://josm.openstreetmap.de/

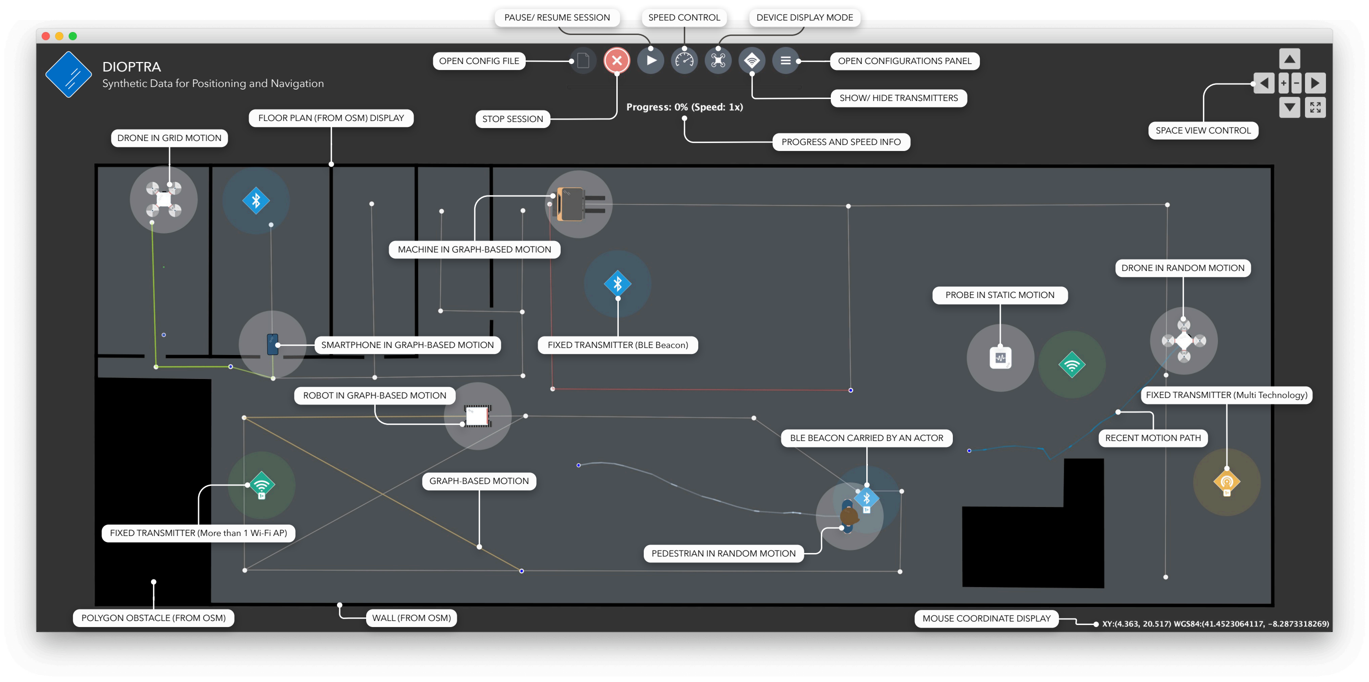

PAUSE/ RESUME SESSION
SPEED CONTROL
DEVICE DISPLAY MODE
DIOPTRA
Synthetic Data for Positioning and Navigation
OPEN CONFIG FILE
OPEN CONFIGURATIONS PANEL
SHOW/ HIDE TRANSMITTERS
Progress: 0% (Speed: 1x)
STOP SESSION
FLOOR PLAN (FROM OSM) DISPLAY
DRONE IN GRID MOTION
PROGRESS AND SPEED INFO
SPACE VIEW CONTROL
MACHINE IN GRAPH-BASED MOTION
DRONE IN RANDOM MOTION
PROBE IN STATIC MOTION
SMARTPHONE IN GRAPH-BASED MOTION
FIXED TRANSMITTER (BLE Beacon)
ROBOT IN GRAPH-BASED MOTION
FIXED TRANSMITTER (Multi Technology)
BLE BEACON CARRIED BY AN ACTOR
RECENT MOTION PATH
GRAPH-BASED MOTION
FIXED TRANSMITTER (More than 1 Wi-Fi AP)
PEDESTRIAN IN RANDOM MOTION
POLYGON OBSTACLE (FROM OSM)
WALL (FROM OSM)
MOUSE COORDINATE DISPLAY
XY:(4.363, 20.517) WGS84:(41.4523064117, -8.2873318269)

Fig. 5. Dioptra GUI: Simulation in Factory 1 Space (Diopta 1.0 - Dark Mode)

shopping mall), users may develop their own space models to test scenarios according to their preference.

When creating the indoor map of the space, users can define the materials of indoor elements such as walls and obstacles. In order to map the material type into an attenuation, users should specify the types of building materials and the associated attenuation caused by each material in the configuration file. Several research works have studied how radio signals are attenuated by distinct building materials [1], [24]. The attenuation values of different building materials for the 2.4 GHz frequency range, used by Wi-Fi and BLE transmitters are extracted from [24]: stucco (14.6 dB), glass (0.5 dB), cinder block (6.7 dB), red brick (4.4 dB), and wood (2 dB). These values are considered when the signal is obstructed by a wall or obstacle. Since they are based on the typical thickness of the materials, the tool does not consider the thickness of the material as defined in the indoor map. Dioptra supports any type of material and respective attenuation value, as long as it is defined in the configuration file. In order to match the materials used in the indoor map with the materials specified in the configuration file, the *material_id* value is used.

H. User Interface

As mentioned before, many of the available simulation tools are private, lack of a proper GUI, require advanced coding expertise or have dependencies to third party libraries, which makes their usage and installation a complex process. Dioptra is designed to be user friendly and ready to install/use as any other common application. Most of the controls for a simulation session are available through the GUI (Fig. 5).

When the user launches the application the first step is loading a configuration file by clicking the respective button and choosing the file. The floor plan of the space is parsed and displayed after the configuration file is loaded. The floor plan dimensions are automatically adjusted to fit the screen. The transmitters in the configuration file are loaded and displayed after the floor plan. A button in the main toolbar allows to show and hide the transmitters.

A view control tool is available to pan, zoom, and reset the space view. In the bottom right corner, the coordinates of the current mouse position in both cartesian and WGS84 referential are displayed. This allows the user to quickly get the coordinates of a specific point in the space.

Devices are not displayed until the simulation session is launched. Before launching the simulation the user can also configure some other parameters, such as the simulation speed (using the toolbar button). The simulation is started by pressing the launch button in the toolbar.

During a work session, users can pause, resume or end the simulation, control the speed and change the view parameters. Fig. 5 shows a work session running with six transmitting devices, three fixed Wi-Fi APs, two fixed BLE Beacons and a BLE Beacon carried by a moving actor. The session includes seven actors, with distinct parameters and sensors including: two pedestrians (one in random motion and another in graph-based motion), a machine and a robot in graph-based motion, and three drones (one in grid motion, one in random motion, and a Probe (fixed scanner) in static mode).

When an actor moves the most recent trajectory (t-s (sec)) is displayed with a different color for each actor. The GUI also displays the current simulation progress (percentage and progress bar) and the simulation speed. The work session ends when the simulation duration (defined in the configuration file) ends or if the user chooses to end the simulation at any time. All generated data is saved automatically if the simulation ends by it self. When a work session ends, users run a new simulation with the same conditions or to load a new configuration file.

The configuration panel, accessible by pressing the respective button in the toolbar, giving access to extra options, between others show/hide a grid inside the space area, change the GUI mode (dark or light mode), set the simulation speed to max, which will run the simulation at the maximum speed supported by the computer, this option disables the visualisation delays. In addition to the GUI, users can opt to use the command line to generate data. The command line can be useful e.g. to automate the data generation and integrate into the user workflow, or for a long session where the user does not want to use the GUI.

In the GUI and command line are shown possible errors, e.g. detected in the configuration file or OSM file. It raises errors whenever a required parameter is missing in an actor, transmitter or sensor.

III. Using the Tool and Contributing

A. How to generate a Radio Map for pedestrian tracking

An example of an use case for Dioptra is the generation of synthetic radio maps and test datasets for assessing fingerprinting methods. Below, we describe the steps to generate a radio map which are reflected in the illustrative configuration file (see Fig.6).

– Setting the simulation parameters: First, the general parameters for the simulation must be set, including: the simulation duration, the random seed to allow reproducibility/repeatability, the path to the folder where all data will be saved, the simulation title/description and the inclusion of an output space model for advanced multi-building scenarios.

– Setting the space: Second, the features of the target environment have to be identified, namely the building and floor identifiers ($[0, \ldots, n]$), the floor height (in m) and the corresponding floor plan (path to the OSM file) must be provided. Additionally, the parameters for the materials generating NLOS conditions in the provided floor plans are provided. For instance, a red brick wall has an attenuation factor of $4\,\mathrm{dB}$ [24].

– Defining the actors: Third, all actors involved must be identified and properly configured. It is worth noting that actors involve from fixed transmitters and autonomous vehicles to pedestrians. We suggest introducing first all the actors involved in the infrastructure providing indoor localization (e.g., Wi-Fi APs) and later the actors involved in the data collection (e.g., a drone collecting the Wi-Fi fingerprints in the reference points distributed over a regular grid).

For an actor acting as an emitter, we need to identify it, provide the type of signals (with parameters) it broadcasts and set its corresponding motion model. For Wi-Fi APs (`type: "wifi_ap"`), the static motion model is preferred

```
{
"settings": {
   "sim_duration": 1,
   "random_seed": -1, //With -1 a random seed will be used
   "output_folder": "Dioptra Data",
   "output_space_model": false,
   "description": "Generate a Wi-Fi Radio Map."},
"space": {
   "floors": [
      {...// Only the first space/floor is parsed
         "building_id": "0","floor_id": "0","floor_height": 0.0,
         "osm_file_name": "scenarios/factory_scenario.osm"
      }...// Multi-building Multi-floor will come in future versions
   ],
   "materials": [
      { "material": "Red Brick", "material_id": "red_brick",
        "attenuation_db": 4.4},... // More materials
   ]},
"actors": [
   { // Actors emitting Wi-Fi probes (APs)
      "actor_id": "WIFI AP 1",
      "type": "fixed_transmitter",
      "status": "enabled",
      "motion_model": {"model_name": "static",
         "x": 20.036, "y": 4.490, "z": 3.0, "yaw": 0},
      "transmitters": [
         {  "transmitter_id": "AP 1",
            "type": "wifi_ap", "status": "enabled",
            "mac": "00:00:00:00:00:00", "channel": 1,
            "power": -21, "ssid": "Network X"
         },...// More networks if allows Virtual nets
      ]
   },...// More Wi-Fi APs
   { // Actor collecting the radio map
      "actor_id": "RadioMap",
      "type": "uav",
      "status": "enabled",
      "motion_model": {
         "model_name": "grid", "z": 1.0,
         "grid_size": 3.0, "time_per_point": 10.0,
         "mode":  "mapping_mode"},
      "sensors": [
         {  "sensor_id": "WIFI1",
            "sample_frequency_hz": 1,
            "sensor_type": "WIFI",
            "noise_sigma": 4, "sensitivity": -90,
            "beacon_rec_prob_function": "g0",
            "propagation_model": {
                "model_name":"ldpl", "fs_path_loss": 2},
            "status": "enabled"},
         {  "sensor_id": "REF1",
            "sample_frequency_hz": 1, "sensor_type": "POSI",
            "status":"enabled"}]}]}
```

Fig. 6. Configuration file for Radio Map collection

as the APs are usually fixed to walls or ceilings. Technology specific parameters include the MAC address, the channel, the transmission power, and the wireless network SSID. An actor can emit multiple signals emulating, for instance, an AP broadcasting multiple virtual APs with different parameters.

For an actor acting as a receiver, we also need to identify it, provide the type of signals (with parameters) it receives and set its corresponding motion model. To generate a radio map we need an actor performing a data collection over, for instance, a regular grid. At every reference point, the actor will collect data for a few seconds. The propagation model is set on the sensor's side to allow the generation of synthetic data with different propagation scenarios in the same work session.

B. How to contribute

Dioptra has been built to allow users to simulate their own indoor locations and positioning scenarios. Through the configuration file, the users can select the corresponding indoor map in OSM format and set all the simulation parameters needed to generate their data as done in Section III-A. Additionally, some examples of use are provided with the tool.

The main website [19] does not only provide the links to download Dioptra and the technical information (user's manual), but it also acts as a repository for the configuration files (including OSM maps) as well as the output data generated by the research community. Having a common place to host the datasets as well as the configuration files would allow researchers to reuse them and perform comprehensive evaluations from diverse sources. Therefore, researchers are encouraged to submit their configuration and output files to Dioptra's website to enrich the repository and provide relevant evaluation scenarios for the community.

IV. Conclusions and Future Work

This paper introduced Dioptra, a synthetic data generation tool aiming to fill the existing gap in the domain of tools for the development and evaluation of indoor positioning systems. Although simulations are commonly used, each research team often uses custom simulators, which impairs fair and reproducible benchmarking of different IPS in the same conditions.

Dioptra is provided as an open-access cross-platform application, planned to become open-source soon. Its architecture is prepared to be easily extended in the future, through contributions from the research community, to integrate new scenarios, new propagation and mobility models, and new sensors. The current version is packaged with a set of pre-build operational scenarios that include the geometric and topological models of the space, fixed infrastructure and a set of sensors to collect data. These scenarios can be used to generate data immediately or be used as a base for creating other operational scenarios.

The architecture of Dioptra is ready and the tool already supports some IPSs technologies. However, new models are required to extend to other variants of IPSs. For instance, new models can be created for new sensors (e.g. barometer or LiDAR), as well as new technologies (e.g. 5G-NR, UWB or infrared and visible light), and to generate data to allow testing IPSs based on Angle of Arrival (AoA), Time of Arrival (ToA), etc. The models of some sensors are still quite simple, such as the model for an AHRS sensor that is limited to provide only the yaw (heading). Therefore, the contributions from the community will be valuable.

Plans for the near future include: i) Compare the performance of IPSs with synthetic data generated by Dioptra and with real-world data for a real-world and virtual setup with similar characteristics. ii) extend the main engine to support multi-building and multi-floor scenarios, as well as elevators, stairwells and ramps; iii) add alternative radio propagation models (e.g. ray tracing); iv) add a simple model for the propagation of optical signals; v) add new mobility models; vi) add support for more advanced sensors, such as IMUs, atmospheric pressure and LiDAR), as well as new radio technologies such as UWB. One major step towards extending the current capabilities of Dioptra is to provide the code open-source to anyone aiming to contribute with the implementation of new models, which is planned as the next task to perform.

References

[1] V. Bahl and V. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *Proceedings of IEEE INFOCOM 2000*, IEEE, Mar. 2000.

[2] Z. Li, Z. Tian, Z. Wang, *et al.*, "Multipath-assisted indoor localization using a single receiver," *IEEE Sensors Journal*, vol. 21, no. 1, pp. 692–705, 2021.

[3] R. Raj, K. Saxena, and A. Dixit, "Passive optical identifiers for vlc-based indoor positioning systems: Design, hardware simulation, and performance analysis," *IEEE Systems Journal*, pp. 1–12, 2020.

[4] K. SongGong and H. Chen, "Robust indoor speaker localization in the circular harmonic domain," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3413–3422, 2021.

[5] L. Cheng, Y. Li, M. Xue, *et al.*, "An indoor localization algorithm based on modified joint probabilistic data association for wireless sensor network," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 63–72, 2021.

[6] Z. Ezzati Khatab, A. Hajihoseini Gazestani, S. A. Ghorashi, *et al.*, "A fingerprint technique for indoor localization using autoencoder based semi-supervised deep extreme learning machine," *Signal Processing*, vol. 181, p. 107 915, 2021.

[7] L. Ferrigno, M. Laracca, F. Milano, *et al.*, "Magnetic localization system for short-range positioning: A ready-to-use design tool," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, 2021.

[8] X. Cui, M. Wang, J. Li, *et al.*, "Indoor wi-fi positioning algorithm based on location fingerprint," *Mobile Networks and Applications*, Jan. 2021.

[9] D. Plets, W. Joseph, K. Vanhecke, *et al.*, "Development of an accurate tool for path loss and coverage prediction in indoor environments," in *Proceedings of the Fourth European Conference on Antennas and Propagation*, 2010.

[10] M. Werner, "Indoorsim : Simulation of wireless-lan-based indoor positioning systems incorporating cad-floorplans," 2011.

[11] T. Jankowski and M. Nikodem, "Smile - simulator for methods of indoor localization," in *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2018, pp. 206–212.

[12] C. Pendão, "FastGraph - Unsupervised Location and Mapping in Wireless Networks," Ph.D. thesis, Univ. Minho, Aveiro and Porto, Braga, 2019.

[13] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," *16th Annu. Int. Conf. Mob. Comput. Netw. - (MobiCom '10)*, p. 173, 2010.

[14] J. Koo and H. Cha, "Unsupervised locating of WiFi access points using smartphones," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 6, pp. 1341–1353, 2012.

[15] C. Pendão and A. Moreira, "Automatic RF Interference Maps and their relationship with Wi-Fi Positioning Errors," in *2019 Int. Conf. Indoor Position. Indoor Navig.*, 2019.

[16] J. Gante, L. Sousa, and G. Falcao, "Dethroning gps: Low-power accurate 5g positioning systems using machine learning," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 2, pp. 240–252, 2020.

[17] O. Belmonte-Fernández, R. Montoliu, J. Torres-Sospedra, *et al.*, "A radiosity-based method to avoid calibration for indoor positioning systems," *Expert Systems with Applications*, vol. 105, 2018.

[18] M. Ayadi, N. Torjemen, and S. Tabbane, "Two-dimensional deterministic propagation models approach and comparison with calibrated empirical models," *IEEE Transactions on Wireless Communications*, vol. 14, no. 10, pp. 5714–5722, 2015.

[19] C. Pendao, I. Silva, J. Torres-Sospedra, *et al.* (2021). "Dioptra's official website." http://dioptra.dsi.uminho.pt.

[20] J. Torres-Sospedra, A. Jiménez, A. Moreira, *et al.*, "Off-Line Evaluation of Mobile-Centric Indoor Positioning Systems: The Experiences from the 2017 IPIN Competition," *Sensors*, vol. 18, no. 2, p. 487, 2018.

[21] J. S. Seybold, *Introduction to RF Propagation*, 1-3. Hoboken, NJ, USA: John Wiley & Sons, Inc., Sep. 2005, vol. 14, pp. 145–173.

[22] K. J. Li, S. Zlatanova, J. Torres-Sospedra, *et al.*, "Survey on indoor map standards and formats," *2019 Int. Conf. Indoor Position. Indoor Navig. IPIN 2019*, 2019.

[23] H. G. Ryoo, T. Kim, and K. J. Li, "Comparison between two OGC standards for indoor space - CityGML and IndoorGML," *Proc. 7th ACM SIGSPATIAL Int. Work. Indoor Spat. Awareness, ISA 2015*, 2015.

[24] R. Wilson, "Propagation Losses Through Common Building Materials 2.4 GHz vs 5 GHz," *Magis Network, Inc.*, pp. 1–28, 2002.