DIOPTRA

Synthetic Data for Positioning Systems

Synthetic Data Generator for Indoor Positioning, Tracking and Navigation

User's Manual

v1.0, 24.11.2021

Dioptra is a tool to support Research and Development (R&D) activities in the field of Indoor Positioning, Tracking and Navigation (IPTN). It can be used to generate synthetic datasets representing the output of the several types of sensors used in IPTN.
One fundamental stage in the research process in IPTN is the evaluation of the proposed methods/algorithms/systems. The most common practice in this field is to evaluate the new systems in real-world conditions. This approach, however, has some disadvantages as a first stage of evaluation, including: (i) quite often, systems are evaluated in very limited spaces, of a single type (e.g. office building or university lab), and for a short period of time; (ii) setting up a complete evaluation environment is a very time-consuming and, many times, a costly task, in particular if long term experiments in large spaces are to be conducted.
Synthetic data, created using a as-realistic-as-possible simulator, can be used for the first stages of evaluation of IPTN systems with the following advantages: (i) data can be easily generated for long periods of time and for large spaces; (ii) a large set of different configurations can be easily tested; (iii) the same data can be used for benchmarking different IPTN systems in a large set of different conditions.

Dioptra is an open-source tool that has been developed to simplify the task of generating synthetic datasets for the evaluation of IPTN systems.

1. Dioptra main features

Dioptra was initially developed to help in the evaluation of Wi-Fi fingerprinting-based indoor positioning systems, and later extended to support other types of IPTN systems.
In its current public version (1.0), Dioptra's main supported features are:

- easy way for defining the geometry and topology of the space, based on the well-known OSM[1] format;
- several motion models for the moving actors;
- one radio propagation model for radio-based beacons;
- models for Wi-Fi Access Points (as beacon) and BLE beacons;
- models for Wi-Fi and BLE scanners;
- models for heading sensors (basic AHRS emulation);
- models for odometers and step-counters;
- a graphical user interface (GUI) to show and control the simulation progress;

[1] OSM – OpenStreetMaps (http://www.openstreetmap.org)

- a command-line interface to directly generate datasets faster;
- text-based (CSV) output files.

Planned to be an open-source tool, Dioptra is prepared to be easily extended to include additional models, such as alterative radio propagation models, motion models and sensor models.

2. Setting up a simulation environment

Generating a dataset using Dioptra is done in four main steps:

- create the geometry and topology of the physical space, or use one of the provided ready-to-use space models;
- configure all the actors, with their beacons and sensors, or use one of the provided use cases;
- set the main simulation parameters;
- run the simulation.

2.1. Geometry and topology of the physical space

The geometry of the physical space for which data is to be created is represented in an XML file based on the OSM format[2]. These files can be created manually, or by using one of the existing OSM editors (e.g. JOSM[3], Vespucci[4]). By using a standard for the representation of the space geometry and topology, Dioptra takes advantage of the existing editors and data format converters.

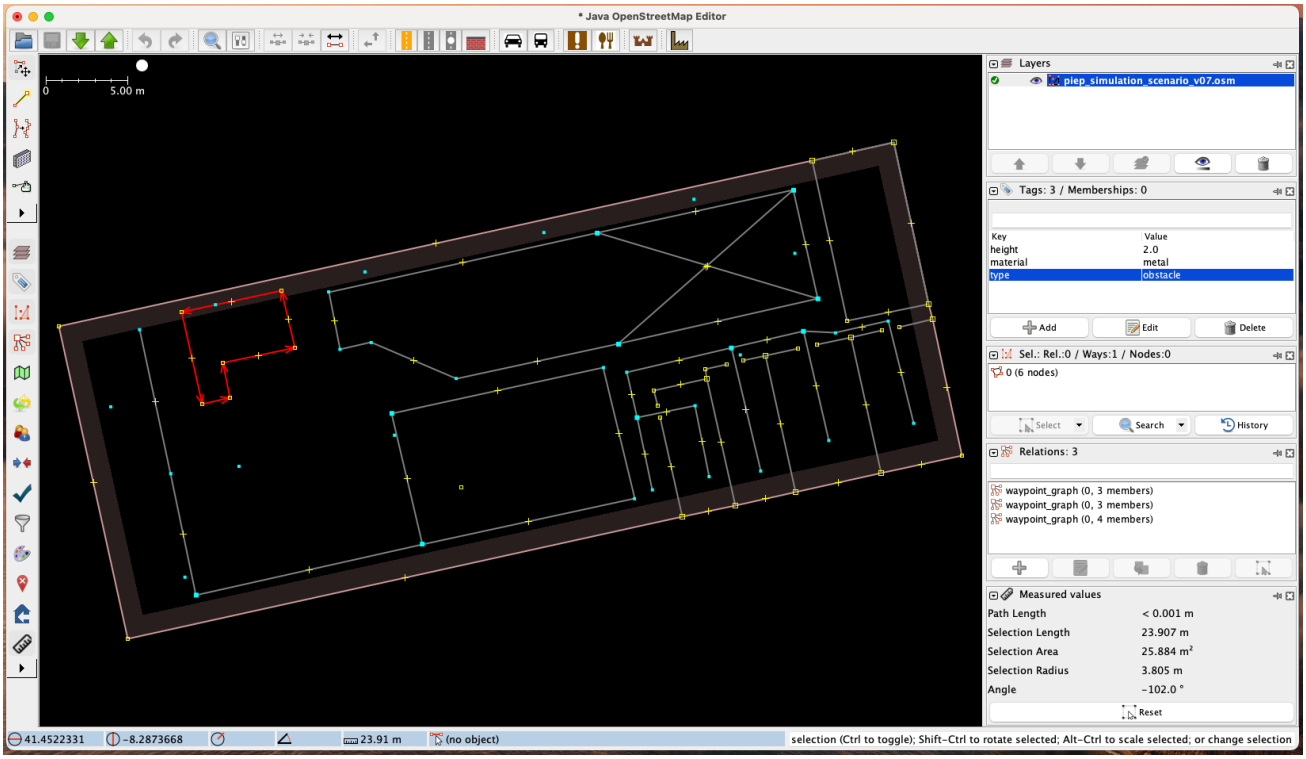

Figure 2.1 The JOSM editor.

[2] https://wiki.openstreetmap.org/wiki/OSM_file_formats
[3] https://wiki.openstreetmap.org/wiki/JOSM
[4] http://vespucci.io/

A very simple example of a file representing a rectangular space is shown below.

```
<?xml version='1.0' encoding='UTF-8'?>
<osm version='0.6' generator='JOSM'>
<!-- GEOMETRY -->
  <node id='-169406' action='modify' visible='true' lat='41.45244434043' lon='-
8.28748857758' />   <- represents a node; these are used later to define geometric objects
…
  <way id='-165502' action='modify' visible='true'>  <- a multi-line
    <nd ref='-169406' />
    <nd ref='-169409' />
    <nd ref='-169420' />
    <nd ref='-169410' />
…
    <nd ref='-169406' />
    <tag k='building' v='yes' /> <- represents the outer shell of the building
    <tag k='height' v='6' />     <- floor height (distance from the floor to the ceiling)
    <tag k='lat0' v='41.4524784' />     <- origin latitude (for conversion to XY)
    <tag k='lon0' v='-8.2872281' />     <- origin longitude (for conversion to XY)
    <tag k='rotation' v='-12.4' />      <- rotation, in degrees, measured from North
    <tag k='material' v='cinder_block' />      <- building shell material
    <tag k='width' v='0.4' />    <- building shell wall width [m]

  </way>                                    anti-clockwise (for conversion to XY)
…
  <way id='-165504' action='modify' visible='true'>
    <nd ref='-169418' />
    <nd ref='-169419' />
    <tag k='material' v='cinder_block' />      <- the wall material
    <tag k='type' v='wall' />           <- a multi-line representing a wall
    <tag k='height' v='3' />            <- height of the wall [m]
    <tag k='width' v='0.3' />           <- width of the wall [m]
  </way>
…
  <way id='-165503' action='modify' visible='true'>
    <nd ref='-169413' />
    <nd ref='-169414' />
    <nd ref='-169415' />
    <nd ref='-169407' />
    <nd ref='-169413' />
    <tag k='material' v='wood' />       <- obstacle material
    <tag k='type' v='obstacle' />       <- a multi-line representing an obstacle; must be
                                           a closed polygon
    <tag k='height' v='2.5' />          <- height of the obstacle [m]
  </way>
…
<!-- TOPOLOGY -->
<node id='-169448' action='modify' visible='true' lat='41.4523636105' lon='-
8.28740693495'>
    <tag k='type' v='graph_node' />     <- represents a node on a graph (topology)
  </node>


…
  <way id='-165520' action='modify' visible='true'>
    <nd ref='-169453' />
    <nd ref='-169454' />
    <tag k='type' v='graph_edge' />     <- represents a set of graph edges (one or more)
  </way>
…
  <relation id='-164789' action='modify' visible='true'>    <- a graph representing the
    <member type='way' ref='-165616' role='graph_edge' />      topology of the space
    <member type='way' ref='-165518' role='graph_edge' />   <- one edge of the graph
    <member type='way' ref='-165613' role='graph_edge' />
    <tag k='graph_id' v='stacker' />    <- the graph ID; many graphs can be defined
    <tag k='type' v='undirected_graph' />
```

```
  </relation>
</osm>
```

This space is made of:

- the *outer shell of the building* (way, type `building`);
- *walls* (way, type `wall`);
- *obstacles* (way, type `obstacle`);
- a set of graphs (relation, type `undirected_graph`) representing the *topology* of the space: each graph is made of a set of points (`graph_node`) connected by a set of edges (`graph_edge`).

For *walls* and *obstacles*, the material can be specified by a keyword (e.g. "`wood`", "`red_brick`", "`glass`", etc.). The characteristics of each material are then specified in the configuration file (see section 2.2).

The topology of a space defines restrictions for traveling between pairs of points in the space, and is represented by one or more graphs. The topology representation can be used to define the movement of an object during the simulation (e.g. a vehicle can only move along the edges of one particular graph).

Multi-floor buildings must be represented by using one OSM file per floor. When these OSM files are used in the configuration file (see section 2.2), the `building_id`, `floor_id` and `floor_height` must be provided.
Dioptra 1.0 only supports single building, single floor configurations. Support for multi-building and multi-floor configurations is expected for future versions.

As shown in the example above, all coordinates used for defining elements in the physical space (geometry and topology) must be represented as (latitude, longitude) pairs of coordinates in the WGS84 datum in order to comply with the OSM format. Latitude and Longitude must be represented as real numbers in the interval (-180°,180°].

Once created, the OSM file must be referred to in the Configuration File.
Examples of files representing several indoor environments are provided with the tool.

2.2. Configuration File

The complete description of the simulation environment is represented in a single text file using the JSON[5] format. This configuration file defines:

- general simulation parameters, such as time duration, output directory, random seed, etc.;
- the space geometry and topology (link to the OSM file(s) described in section 2.1), as well as the characteristics of the materials used in walls or obstacles;
- a set of actors, fixed or mobile, each one of them hosting transmitters (e.g. a Wi-Fi AP) and/or sensors (e.g. a Bluetooth scanner or and odometer);
- the mobility models for each one of the moving actors;

[5] https://www.json.org/

- the radio propagation model used when measuring received signals in radio-based sensors.

Figure 2.2 illustrates the structure of the configuration file.

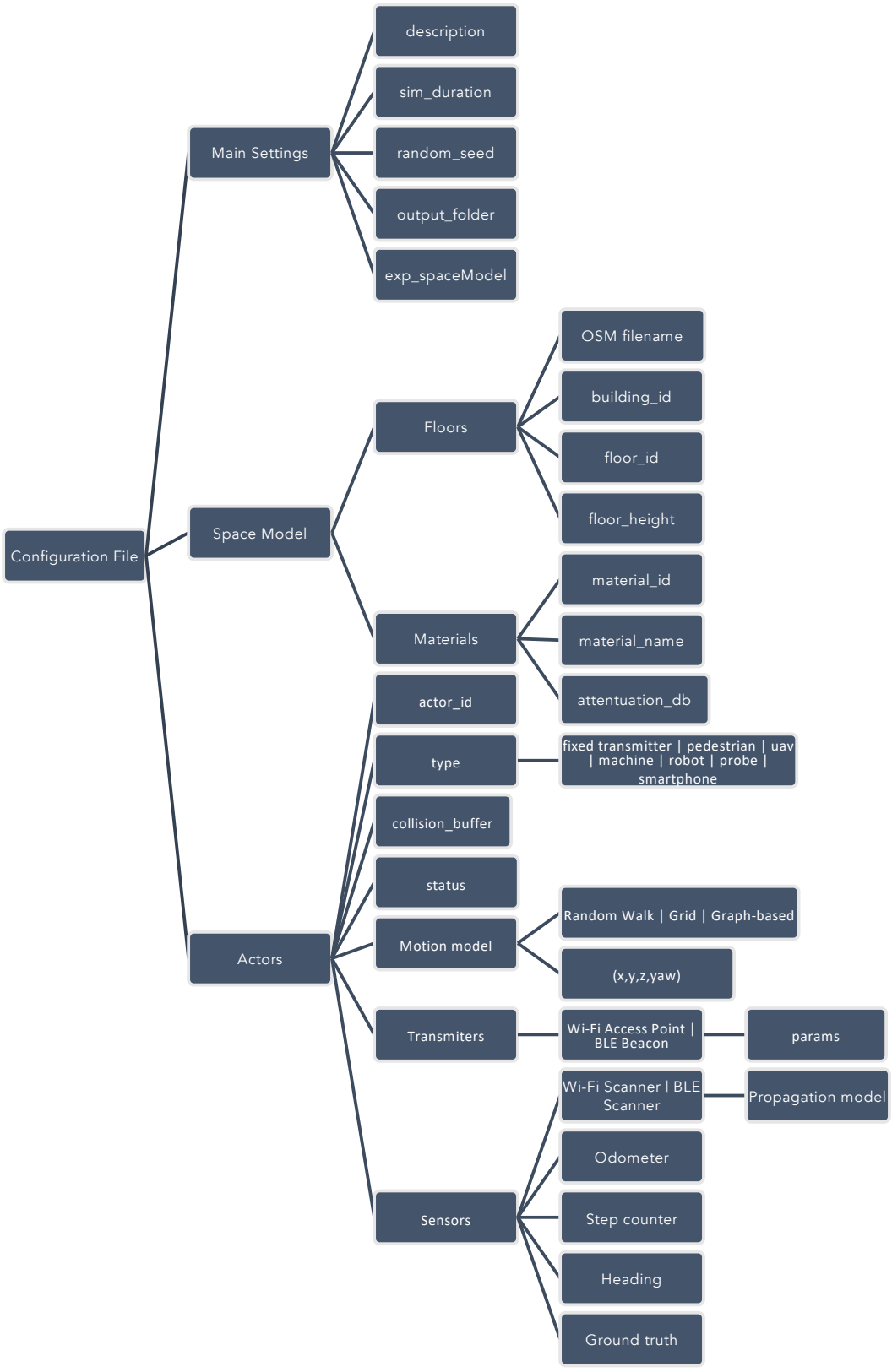

Figure 2.2 Structure of the Configuration File

An example of a configuration file is shown below.

```
{
    "settings": {                                       <- Main settings
        "sim_duration": 600,
        "random_seed": -1,
        "output_folder": "Dioptra Data",
        "output_space_model": false,
        "description": "This is an open text area used to describe the simulation."
    },

    "space": {                                          <- Space definition
        "materials": [
            {
                "material": "Cinder Block",
                "material_id": "cinder_block",
                "attenuation_db": 6.7
            },
            …
        ],
        "floors": [                                     <- reference to the OSM files
            {
                "building_id": "0",
                "floor_id": "0",
                "floor_height": 0.0,
                "osm_file_name": "Scenarios/roomB_v02.osm"
            }
        ]
       },

   "actors": [
           {
               "actor_id": "WIFI AP 1",                 <- one fixed actor with a Wi-Fi AP
               "type": "fixed_transmitter",
               "status": "enabled",
               "motion_model": {
                   "model_name": "static",
                   "x": 20.036,
                   "y": 4.489,
                   "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"
                   }
               ]
           },
   {
               "actor_id": "Device1",           <- one pedestrian carrying two sensors
               "type": "pedestrian",
               "status": "enabled",
               "collision_buffer": 0.8,
               "motion_model": {
                   "model_name": "random",
                   "z": 0,
                   "yaw": 0,
                   "max_speed":1.2,
                   "yaw_var":10
               },
               "sensors": [
```

```
                {
                    "sensor_id": "AHRS1",
                    "sample_frequency_hz": 20,
                    "sensor_type": "AHRS",
                    "noise_sigma": 0,
                    "sensor_drift": 1.0,
                    "status": "enabled"
                },
                {
                    "sensor_id": "REF2",
                    "sample_frequency_hz": 20,
                    "sensor_type": "POSI",
                    "status": "enabled"
                },
            ]
        },
}
```

The Configuration File should map the structure shown in Figure 2.2.

Configuring the position of fixed elements, such as Wi-Fi Access Points or Bluetooth beacons, is easier if performed in three steps: (1) run Dioptra with a basic configuration file pointing to the OSM file and with the GUI enabled; (2) use the GUI (see section 3 below) to read the coordinates where those elements are to be placed; (3) edit the configuration file and add those elements.

2.3. Running Dioptra

Dioptra is an application developed in Java. To run it, users need to have the Java Runtime Environment[6] installed. Dioptra has been tested with Java version 8. No external libraries need to be installed as all the required libraries are included in the provided jar file.

Once the space geometry/topology and the configuration files are created, a dataset can be generated by running Dioptra in the command line:

```
java -jar Dioptra1.0.jar configs.json
```

where `configs.json` is one configuration file as described in section 2.2. The configuration file is optional. If it is provided, Dioptra runs without the GUI. If no configuration file is provided in the command line, the tool opens the GUI and offer the user the possibility to choose a configuration file from the menu.

Executable applications are also provided for OSX (Mac) and Microsoft Windows. However, be aware that security warnings might be issued by the operating system when opening the tool directly from the executable file.

The generated dataset is written to the directory defined in the Configuration File. The formats of the generated files are described in section 4.

[6] https://www.oracle.com/java/technologies/javase-jre8-downloads.html

3. Graphical User Interface

While running a simulation, the space geometry/topology, the position of the actors and their trajectories can be inspected through a Graphical User Interface (GUI). A view of the GUI is shown in Figure 3.1.

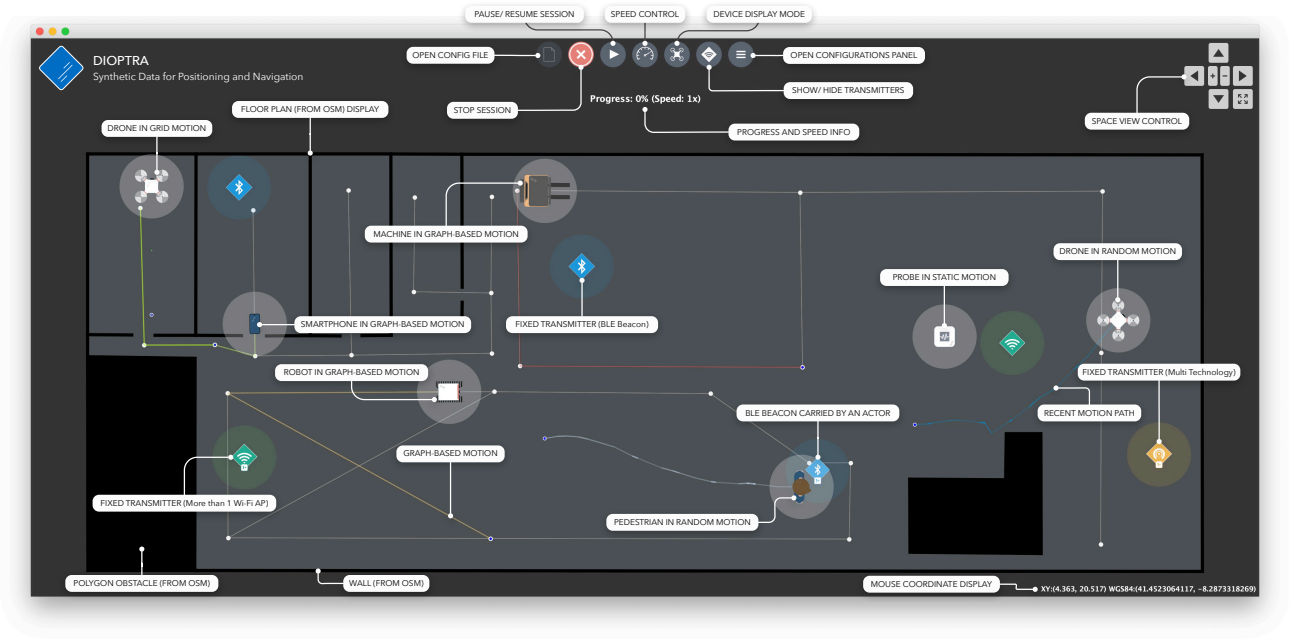

Figure 3.1 The Graphical User Interface

While in the current version the GUI cannot be used to interfere with the simulation parameters, it offers a set of functionalities (icons on top from left to right):

- load a configuration file (icon 1);
- launch/stop the simulation (icon 2);
- pause/continue the simulation (icon 3);
- change the simulation speed (icon 4);
- change how actors are displayed (icon 5);
- show/hide transmitters (icon 6);
- change some other configurations (icon 7);
- zoom in/out/centre (top right icons);
- panning (top right icons);
- see the coordinates of the cursor in (x,y) and (lat,long) coordinates (bottom right);
- read coordinates of specific points by moving the mouse cursor within the simulation area; this functionality is particularly useful to read the coordinates where to place fixed elements, such as BLE beacons.

Closing the GUI window while running a simulation terminates the simulation without exporting the data generated so far.

4. Output data format

Several files are generated in each session of the simulator. Some files are the output of the sensors and other information related to the session, while other files are about the tool itself (logging). These files are described in the following sections.

4.1. Session output files

Data generated by Dioptra is written to a set of text files. One file is generated per actor, containing all the data collected by the corresponding sensors.
These files are written to the directory specified in the configuration file ("`output_folder`"). The names of the files follow the format:

```
yyyyMMdd_HHmmss_actor_id.csv
```

The format of these files is as shown in the example below.

```
% This is a comment
AHRS;sensor_id;time;x;y;z;w;roll;pitch;yaw     <- an AHRS measurement

POSI;sensor_id;time;x;y;z;roll;pitch;yaw;floor_id;building_id
                                               <- a ground truth record (true pose of
                                                  the device

ODOM;sensor_id;time;displacement               <- a measurement from an odometer

STEP;sensor_id;time;step_length                <- a detected step and step length

WIFI;sensor_id;time;SSID;BSSID;channel;RSSI    <- a measurement from a Wi-Fi scanner
                                                  a complete scan is represented by
                                                  multiple lines like this one

BLE4;sensor_id;time;'dioptra';beacon_id;deployment_id;major_id;minor_id;RSSI
                                               <- a measurement from a BLE scanner
```

This format is similar to the one used in IPIN Competitions[7], in particular to the format used in Track 3 [2]. Note, however, that the two formats are not fully compatible.

In summary:

- lines starting with % are comments;
- the decimal separator is the character '.';
- all timestamps are represented as the number of milliseconds elapsed since the beginning of the simulation;
- all other lines represent the output of a sensor or a ground truth pose (reference sensor):
  - AHRS: the output of a soft-sensor that estimates the heading, in the format: `AHRS;sensor_id;timestamp;x;y;z;w;roll;pitch;yaw`, where (x,y,z,w) is the orientation represented by a quaternion; (roll,pitch,yaw) are measured in degrees;
  - POSI: a ground true record of the pose of an actor, in the format: `POSI;sensor_id;timestamp;x;y;z;roll;pitch;yaw;floor_id;buiding_id`, with (x, y, z) represented in meters and (roll,pitch,yaw) measured in degrees;

[7] http://evaal.aaloa.org/2020/call-for-competitions2020

- ODOM: a measurement from an odometer, in the format: `ODOM;sensor_id;timestamp;displacement`, with displacement in meters;
- STEP: the output of a soft-sensor that detects a step (human walk), in the format: `STEP;sensor_id;timestamp;step_length`, with the step length in meters;
- WIFI: a measurement from a Wi-Fi scanner, representing one detected Access Point (AP), in the format: `WIFI;sensor_id;timestamp;SSID,BSSID;channel;RSSI`, where SSID is the Wi-Fi network name, BSSID is the MAC address of the AP, channel is an integer and RSSI is measured in dBm; the complete output of a scan might require multiple consecutive WIFI records, all sharing the same timestamp;
- BLE4: a measurement from a Bluetooth Low Energy scanner, representing one detected beacon, in the format: `BLE4;sensor_id;time;'dioptra';beacon_id;deployment_id;major_id;minor_id;RSSI`, where 'dioptra' defines the specific format used by Dioptra tool, beacon_id is a unique identifier of the BLE beacon, RSSI is measured in dBm, and deployment_id defines a set of beacons used for a particular application, and major_id and minor_id are values assigned to BLE beacons[8]; the complete output of a scan might require multiple consecutive BLE records, all sharing the same timestamp.

In addition to the files with the sensor measurements, two other files are generated:

- one file with the geometry of the obstacles (`yyyyMMdd_HHmmss_obstacles.csv`), where each obstacle is described by one or more lines with the following format: `obstacle_id;x1;y1;x2;y2`, representing lines (one single line) or polygons (multiple lines); this file is generated only if the flag "`output_obstacles`" is set to true in the configuration file;
- one file with the position of the fixed transmitters/beacons (`yyyyMMdd_HHmmss_transmitters.csv`), where each line has the format: `transmitter_type;transmitter_id;x;y;z`

4.2. Logging

For each session, a text file is also created in the directory specified in the configuration file ("`output_folder`") with logging data about the tool functioning.

5. Extending Dioptra

Dioptra will be available as an open-source project at GitHub. Researchers using Dioptra will be invited to contribute to the tool development by adding new features and/or new models (motion, sensors, radio propagation, etc.).
Further information on how to extend Dioptra will be provided in a separate document for developers.

6. Licensing

Dioptra is provided free of charge under the BSD Licence.

[8] https://support.kontakt.io/hc/en-gb/articles/201620741-iBeacon-Parameters-UUID-Major-and-Minor

7. Support

Being an academic project, the team that developed Dioptra does not provide support for this tool. However, we plan on continuing to use and further improve Dioptra in the future and in interacting with users/programmers aiming to contribute to its enrichment. Interaction is encouraged through GitHub (Issues).

References

[1] Pendão C., Silva, I., Moreira A., Torres-Sospedra, J. (2021) Dioptra – A Data Generation Application for Indoor Positioning Systems. In 2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN) IEEE.

[2] Jiménez, A. R., Seco, F., & Torres-Sospedra, J. (2019). Tools for smartphone multi-sensor data registration and GT mapping for positioning applications. In 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN) (pp. 1-8). IEEE.