Reykjavík University

Indoor Positioning and Indoor Navigation

T-785-IPIN

Indoor Positioning System

11 November 2020

Students:
William Paciaroni
Matteo Guerrini
Giacomo Menchi

Instructor:
Marcel Kyas

Contents

1 Introduction

The aim of this system is to track some person path inside Reykjavik University using data provided by Marcel Kyas recorded on a Pixel 3 mobile phone while walking.

This data contains two different types of information:

- One **.csv file** which is the log of all the sensors measurements used in this experiment. It contains data measured by the phones IMU, distances to Bluetooth beacons (using RSSI values) and WiFi access points (using fine time measurement).
- One **.xml file** containing info about the building map comprising walls, obstacles, beacons and access points locations. It also includes the ground truth positions location. In brief, it can be used to read the previous file.

The main project objective is to estimate the path created by a person following these steps:

- Write a script to **parse the data** file previously cited.
- Identify the **RSSI readings and locations** regarding Bluetooth beacons.
- **Estimate the distances** using these RSSI values.
- Use distances in order to **calculate device positions** and **track the path**.

The code to achieve this result is written in Python and can be downloaded at this link:

https://github.com/williampaciaroni/IPIN-System

2 Data Analysis Process

2.1 Understanding Data

The first thing to do is to understand all the data contained inside both files, in order to be able to parse it and use it.

- The **.csv file** info is found inside the "reader", where C++ code to parse data is provided. The reader code also has a link in one comment where all the different sensors data types are listed. The link is **this**.
- The **.xml file** is pretty straightforward since it contains data divided into sections, where each section has the name of the corresponding information type.

2.2 Parsing Data

Using the reader again, it is possible to select just the data types needed from the .csv file. In order to estimate the distances and approximate the device positions, the only two data types needed are Bluetooth beacons and WiFi FTM.

In this experiment, the focus is pointed on the Bluetooth beacons only, which correspond to the ID 9 (thus, it is the only ID used).

Once all the entries which have ID 9 are filtered, the data that can be obtained is the following:

- **MAC Address**: they are compared to the addresses inside the .xml file and, if they match, it is possible to gain the beacons locations from it and hence calculate distances.
- **RSSI**: it is used to execute the distance calculation. Since it is available, it is possible to calculate the distance using the Log-distance path loss model

$$d = 10^{\frac{P_t - P_0 - RSSI}{10 \cdot \gamma}}$$

Formula 1: Log-distance Path Loss Model

$P_t = 0$ and $P_0 = 56$dBmW

- **TX power**: it is parsed, but it is not used inside the experiment since is not always correct (some beacons manufacturers don't send a TX power value inside the transmitted packet, and thus this value may not be available, as in this case). A standard value is provided instead.

Data from the .xml file, instead, is used to get the coordinates for every type of data contained (listed above), which are then plotted inside a Cartesian plane.

Once again, since this experiment only uses Bluetooth beacons, WiFi access points and ground truth positions are ignored.

By parsing the data, it is possible to obtain a list of all the Bluetooth beacons which are also present in the .xml file, their distances from the supposed position, and the timestamps in which the distances/positions were calculated.

It is important to notice that timestamps are taken in consideration every 10 seconds, so we divided the timestamp value by 10 billion in order to group data in 10 seconds groups and reach to a better result (more data might bring to better accuracy).

2.3 Optimizing Data

The next step is to optimize the data gained in order to check which could be useful to estimate the device location and which could not.

Data optimization happens on two different levels:

- **Data Grouping:** in this step, all beacons obtained are grouped in different arrays, using the 10 seconds timestamp (as explained in the previous section), in order to facilitate the position estimation.

- **Data Refining:** for each array, it is verified if there are beacons that appear more than one time, and then a distance average is calculated preserving only one of them, in order to avoid duplication and make trilateration simpler.

2.4 Estimating Positions

Once data has been filtered, it is possible to use a position estimation algorithm to calculate the approximate device location.

At least three different approaches can be used here:

- **Simple Trilateration**: a trilateration algorithm is implemented which evaluates all the landmarks (the Bluetooth beacons) and tries to estimate a position. Its problems are that it needs at least three landmarks (hence the name "trilateration") and that it's not very accurate when giving results.
- **Trilateration using Maximum Likelihood Estimator**: same algorithm as before, but implementing a Maximum Likelihood Estimator, which finds the point in the parameter space that maximizes the likelihood function and thus better approximates the location. Its problem is that it still needs at least three landmarks to work correctly.
- **Clustering**: this last approach uses a different concept. The distances from the landmarks are used as radii to draw circles around them: by calculating the intersections between those circles and forming a cluster of these intersections, it is once again possible to estimate the true ground position. If circles never intersect, the ground truth position is instead calculated as the midpoint of the distance between the two circumferences. Its problem is that if circles are very close but don't intersect (less than , the clustering point can't be calculated.

By evaluating the results of this experiment, Trilateration using Maximum Likelihood Estimator seems to be the most reliable (has the fewest quantity of "impossible" locations, like those outside of the building perimeter), but all of them have their use case and thus could be useful in different situations, so it's always better to have them all implemented and evaluate how their behaviour changes in different experiments.

After implementing the algorithm, the next step is to use it once for each landmark/distance set which are under the same arbitrary time interval (which we set to 10 seconds in the section before), thus getting as a result the estimated device position at any point in time.

2.5 Plotting Data

Once all the estimated positions have been obtained, the last step is to plot them in a Cartesian plane one by one with a fixed delay, so that the complete device path can be shown one position at a time.

To achieve this, **matplotlib.animate** can be used to plot a sequence of locations with a fixed time delay (0.5 seconds is suggested to show them rapidly enough but not too much) and it also allows to connect the plotted points with a line (which aids in showing the followed path).

At **this link** there are some data plotting examples which are achieved using the three different algorithms and 10 seconds as time interval: as shown in the video it is clear that some of the points are not well approximated (some locations appear out of the building, which shouldn't happen), but the reasons behind this will be explained in the next section, alongside with the conclusions.

3 Results Evaluation and Conclusions

From this experiment, it is possible to see that different algorithms bring to radically different solution:

- Using **Simple Trilateration**, all 31 points are found (one for each timestamp), 23/31 points are plotted in the map and 20/31 are in "possible" locations. Some points are not plotted due to the fact that they are invalid for the reasons explained in the algorithm description, and thus can't be used.
- Using **Trilateration using Maximum Likelihood Estimator**, all 31 points are found (one for each timestamp), 29/31 points are plotted in the map and 25/31 are in "possible" locations.
- Using **Clustering**, all 31 points are found (one for each timestamp), 24/31 points are plotted in the map and 22/31 are in "possible" locations.

To summarize, all three methods work, but in different ways.

- **Simple Trilateration** finds few points and several of them are also not usable to estimate the track.
- **Trilateration using Maximum Likelihood Estimator** finds a lot more points but more of them are useless, making it better overall but not absolutely.
- **Clustering** finds few points but almost all of them are usable for plotting, meaning that it might get more accurate results even with less points.

Overall, as stated before, the best algorithm to use in this experiment is Trilateration using Maximum Likelihood Estimator, because it gives the highest number of results with the lowest amount of failures, but the results could change in other experiments and this is the reason why all three algorithms (and possibly even others) should be considered using when estimating a person track.