Final Report

Mert Bayraktar

202151075008

Introduction

Indoor localization refers to the process of determining the location of an object or person inside a building or enclosed space using various technologies and techniques. While GPS (Global Positioning System) is commonly used for outdoor positioning, it is often unreliable or inaccurate indoors due to signal attenuation and multipath effects caused by buildings and other structures.

Agent-based modeling (ABM) is a computational modeling technique used to simulate the behavior and interactions of individual agents within a system. When applied to indoor localization, agent-based modeling can provide a powerful framework for studying and understanding the dynamics of indoor positioning systems.

In an agent-based model of indoor localization, the system is represented as a collection of autonomous agents, each of which represents an object or person within the indoor environment. These agents possess certain characteristics, such as movement patterns, sensing capabilities, and decision-making algorithms, which allow them to navigate and interact with their surroundings.

Agents in an indoor localization model can be designed to mimic real-world entities, such as occupants in a building or mobile devices with positioning capabilities. They can have attributes like location, velocity, and orientation, and can perform actions like moving, sensing the environment, and transmitting or receiving signals.

Fingerprinting

Fingerprinting usually requires an environmental survey to collect features of the environment (fingerprints) and then determine the location of the target by matching real-time measurements with the closest fingerprints stored in the database. The fingerprint is a vector of statistical attributes of a signal received from multiple nodes, such as mean, variance, and histogram.

The fingerprinting technique consists of two stages: the offline stage and the online stage. In the offline phase, a field survey in an environment or a signal propagation model should be used to create radio maps. In the online stage, the localization algorithm uses the real-time signal vector and stored signal vectors in the database to estimate the device's location.

Dataset

The dataset is provided as "csv" files to facilitate data manipulation, and no special software is required to read "csv" files. The 'csv' file consists of the following columns:

1. Coordinates: The three columns in the dataset show the GPS coordinates (latitude, longitude, and floor) of the classes where the measurements were made. Example: for a measurement in the dataset (x, y, z) = (36.89672737982672, 30.649524638866378, 1).

2. RSSI and Mac Address: There are 85 columns as all MAC addresses seen during the total measurement period. These columns are sorted numerically and alphabetically. In each line, the RSSI information received from the Wi-Fi access point with that MAC address is displayed in the data set. Each row corresponds to one measurement. Inaudible access points are set to 0 dBm. For example, in the dataset, the Wi-Fi access point with the MAC address “x” was heard at -66 dBm. RSSI values are interpolated between the timestamps of each measurement.

3. Timestamp: The timestamp column represents the interpolated timestamps of each measurement in UNIX time format. The timestamps provided are saved on a mobile device between the start and end times of each measurement.

4. Room: The room column shows the ID number of the room where the measurements were made. Data collection was done in the classrooms of the Engineering Faculty building, and these room identification numbers were entered before the measurements were taken in the mobile application. There are a total of 20 classrooms in the data set.

Simulation Overview

The simulation model consists of two types of agents: IndoorAgent and MobileAgent. The IndoorAgents represent stationary elements in the indoor environment, such as access points or reference points. The MobileAgents represent mobile devices whose location needs to be estimated. These agents interact and simulate the localization process within the environment.

The IndoorAgent is responsible for simulating the behavior of stationary elements in the indoor environment. Each IndoorAgent keeps track of the received signal strength indicator (RSSI) values from the MobileAgents. This information is vital for estimating the location of the MobileAgents. The IndoorAgent calculates the RSSI values by determining the distance between itself and each MobileAgent. It utilizes a signal propagation model to calculate the RSSI based on the distance. These RSSI values are then stored for further use by the MobileAgents.

The MobileAgent represents a mobile device whose location needs to be estimated within the indoor environment. Each MobileAgent can move randomly within the environment, simulating the movement of a real mobile device. In the simulation, the MobileAgent updates its position on the grid by randomly selecting a neighboring location to move to. After moving, it predicts its location using the RSSI values received from the IndoorAgents. By comparing the RSSI values and using a signal propagation model, the MobileAgent estimates its distance from each IndoorAgent and predicts its location accordingly.

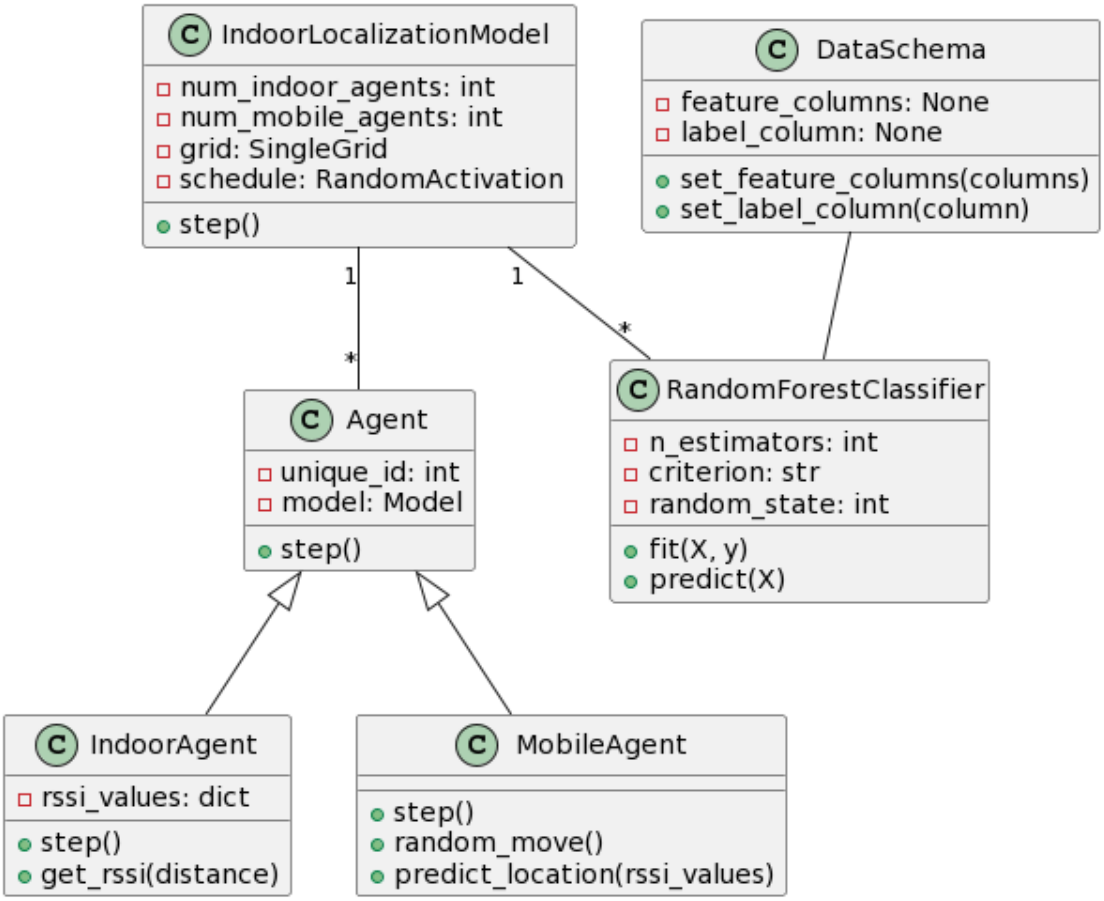

Figure 1. UML diagram of agent-based indoor localization model for classification problems.

Figure 1 represents a UML class diagram describing the classes and their relationships in a simulation model related to indoor localization, as well as some additional classes. Explanation of the different elements present in the diagram are given below:

Agent: This class serves as the base class for other agent classes in the simulation. It has private attributes unique_id (an integer) and model (an instance of the Model class) and a public method step().

IndoorAgent: This class represents an indoor agent in the simulation. It has a private attribute rssi_values (a dictionary) and public methods step(), calculate_distance(mobile_agent_pos), and calculate_rssi(distance).

MobileAgent: This class represents a mobile agent in the simulation. It has public methods step(), random_move(), predict_location(), and calculate_distance(rssi_value).

DataSchema: This class represents the data schema used in the simulation. It has private attributes feature_columns and label_column (both initially set to None) and public methods set_feature_columns(columns) and set_label_column(column).

RandomForestClassifier: This class represents a random forest classifier used in the simulation. It has private attributes n_estimators (an integer), criterion (a string), and random_state (an integer) and public methods fit(X, y) and predict(X).

IndoorLocalizationModel: This class represents the simulation model for indoor localization. It has private attributes num_indoor_agents and num_mobile_agents (both integers), grid (an instance of the SingleGrid class), schedule (an instance of the RandomActivation class), and a public method step().

The associations in the diagram show the relationships between the classes. Specifically:

Agent is the base class for IndoorAgent and MobileAgent.

DataSchema is associated with RandomForestClassifier, indicating that the data schema is used by the random forest classifier.

IndoorLocalizationModel has associations with multiple Agent instances and multiple RandomForestClassifier instances.

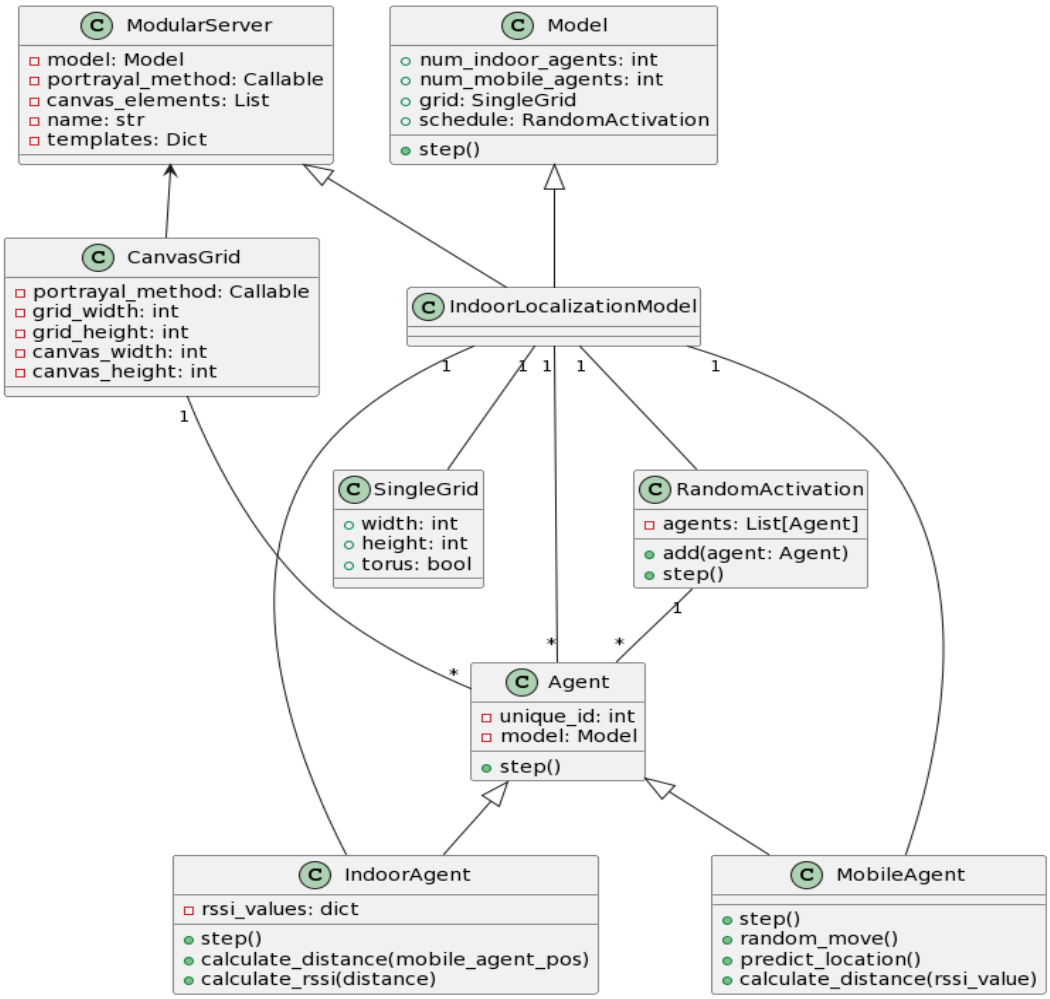

Figure 2. UML diagram of agent-based indoor localization model for the simulation.

Figure 2 represents a UML class diagram describing the classes and their relationships in a simulation model related to indoor localization. . Explanation of the different elements present in the diagram are given below:

Agent: This class serves as the base class for other agent classes in the simulation. It has private attributes unique_id (an integer) and model (an instance of the Model class) and a public method step().

IndoorAgent: This class represents an indoor agent in the simulation. It has a private attribute rssi_values (a dictionary) and public methods step(), calculate_distance(mobile_agent_pos), and calculate_rssi(distance).

MobileAgent: This class represents a mobile agent in the simulation. It has public methods step(), random_move(), predict_location(), and calculate_distance(rssi_value).

SingleGrid: This class represents a single grid in the simulation. It has public attributes width and height (both integers) and a torus attribute indicating whether the grid has a torus topology.

RandomActivation: This class represents the random activation of agents in the simulation. It has a private attribute agents (a list of Agent objects) and public methods add(agent: Agent) and step().

CanvasGrid: This class represents a grid visualization using a canvas. It has private attributes portrayal_method (a callable) and grid_width, grid_height, canvas_width, and canvas_height (all integers).

ModularServer: This class represents a modular server for visualization in the simulation. It has private attributes model (an instance of the Model class), portrayal_method (a callable), canvas_elements (a list), name (a string), and templates (a dictionary).

Model: This class represents the simulation model. It has public attributes num_indoor_agents, num_mobile_agents, grid (an instance of SingleGrid), schedule (an instance of RandomActivation), and a public method step().

IndoorLocalizationModel: This class represents a specific type of model related to indoor localization. It is a specialization of the Model class. It has associations with multiple Agent instances, a SingleGrid, RandomActivation, IndoorAgent, and MobileAgent.

Simulation Progress

The simulation progresses over time by executing the step function of each agent. The agents' steps are scheduled and managed by the Mesa library, which ensures the synchronization and interaction between the agents.

During each step, the IndoorAgents calculate the RSSI values from the MobileAgents based on their distances, and the MobileAgents update their positions and predict their locations based on the received RSSI values. This iterative process simulates the dynamic nature of the indoor localization process.

The simulation model is visualized using the Mesa library's visualization capabilities. The grid visualization provides a visual representation of the indoor environment, with each agent represented by a specific shape and color.

In the visualization, the IndoorAgents are displayed as red rectangles, representing their stationary nature within the environment. The MobileAgents are represented as blue circles, symbolizing their mobility. Additionally, the current position of each MobileAgent is displayed as text within the visualization.

Figure 3. Initial state of the simulation environment and mobile agent.

Figure 4. Simulation environment and mobile agent's position after one step.

```
Mobile agent 10 is moving to 6 18
Mobile agent 10 predicted location: (4, 14, 4.12310562561766)
```

Figure 5. Predicted location of mobile agent after one step.

Conclusion

In this report, we have explored an indoor localization simulation using a model-driven approach. By simulating the behavior and interactions of IndoorAgents and MobileAgents, we can gain insights into the dynamics of indoor localization and optimize localization algorithms. The simulation model, visualization, and demonstration have provided a practical understanding of how the model-driven approach can contribute to indoor localization research and development.

While the presented model-driven approach provides valuable insights into indoor localization, incorporating real-time data into the simulation can further enhance its accuracy and practicality. By integrating real-time data into the indoor localization simulation, we can bridge the gap between simulation and real-world scenarios, enabling more accurate analysis, optimization, and validation of localization algorithms.