Developing an indoor localisation and wayfinding app for a University Library

Dimitris Christodoulou

MInf Project (Part 2) Report
Master of Informatics
School of Informatics
University of Edinburgh

2022

Abstract

The Main Library at the University of Edinburgh houses an extensive book collection and provides access to multiple resources such as study spaces and electronics workshops. Library users, especially new students, can sometimes find it tricky to navigate, and would benefit from the provision of better location-based information.

In this report, we describe our work towards building a localisation and wayfinding Android smartphone application, with the aim of helping library users find what they are looking for quickly and easily. The app can detect a user's location using signal strength measurements from nearby Wi-Fi access points, and give them directions to points of interest of their choice.

Localisation is done using the fingerprinting approach. Different preprocessing and machine learning algorithms were compared and a Gaussian process based approach was found to produce the best results.

The pathfinding algorithm uses topological maps generated in advance, using the Zhang-Suen thinning algorithm in order to efficiently find paths from the user's current location to any location in the library.

The proposed localisation and pathfinding approaches were evaluated by conducting a user study, where participants were asked to use the app in a section of the library to complete a sequence of tasks, which received a positive response overall.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 2021/59600
Date when approval was obtained: 2022-03-14
The participants' information sheet and a consent form are included in the appendix.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Dimitris Christodoulou*)

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Petros Papapanagiotou for his guidance and support throughout this project.

I would also like to thank my friends and family for their continuous support.

Table of Contents

Chapter 1

Introduction

1.1 Motivation and Goals

The University of Edinburgh's Main Library is located at the University's George Square campus and is used by thousands of students and staff daily. It houses a massive book collection spanning across six floors and provides hundreds of study spaces, electronics workshops, among other resources. It can sometimes be tricky to navigate the building, especially for students new to the University. Even for regular users of the library, finding a specific book can sometimes be difficult. Finding an empty study space can also be challenging, especially around busy periods such as the exam season.

We seek to improve access to library resources and help users of the library find what they are looking for quickly and easily. In last year's project, we focused on improving the occupancy information that can be provided to students, in order to make finding an empty study space more straightforward, as discussed in Section 1.4.

This project has the goal of building an indoor localisation and pathfinding system, accessed by users through an Android smartphone application. It will help guide users around the building to find books, study spaces or any other library resource.

More specifically, the app will show their current location on a map of the building, allowing them to orient themselves. Location-based services can then be provided through the app. In particular, this project also aims to implement pathfinding functionality that can display a path to users in order to reach different points of interest in the building, such as a study space or a book.

While this project is focused in the context of the University Library, the findings can be applied in a variety of different environments. For example, a similar app could help people navigate railway stations, airports or even provide guided tours in museums.

1.2 Approach

Our first step is to explore what the common approaches for indoor localisation are, and how these techniques have been used so far in related work (Chapter 2). After

deciding our general localisation approach, we collect the data we need and experiment with and compare different preprocessing approaches, data augmentation techniques and machine learning algorithms, in order to build our localisation model (Chapter 3). Then, we describe our approach for indoor pathfinding, which involves generating a topological map for the space (Chapter 4). We set up a server that provides access to the system through a Web API. We developed an Android app which provides localisation and pathfinding results by interacting with the API (Chapter 5). The localisation model and our pathfinding approach were evaluated with the help of a user study, where participants were asked to complete a sequence of tasks using the app (Chapter 6).

1.3 Contributions

The main contributions of the project can be summarised as follows:

- An Android application that can be used to collect Wi-Fi signal strength data for localisation in indoor environments.
- Wi-Fi signal strength datasets that can be used for localisation, collected at the University Library and from a computing lab at the University.
- A localisation model for a section of the University Library, built after evaluating different preprocessing and data augmentation approaches.
- An implementation of a pathfinding algorithm that can provide directions to users to reach different points of interest in an indoor environment.
- An Android application which can be used for localisation and pathfinding to different points of interest within the University Library.
- A critical evaluation of the localisation and pathfinding systems based on a user study, where participants were asked to perform a sequence of actions using the application in the University Library.

1.4 Previous work carried out

This project follows last year's work on occupancy monitoring in the library. Better occupancy information makes access to the library easier for students, and helps staff manage resources more effectively. In that project, we made use of Wi-Fi data collected centrally through the access points in the library building, which keep a timestamped log of when devices connect and disconnect from them.

First, we performed an analysis of the data to discover its key characteristics. We proposed a preprocessing pipeline based on our observations. Then, we developed and evaluated a model that detects users that are 'settled' anywhere in the library building. We defined that someone is 'settled' when they are stationary somewhere in the building for a significant amount of time - typically 5 minutes or more - and possibly using some resource of the library. This model can be used to get an estimate on the levels of use of different parts of the building, which can be provided to students and library staff.

Chapter 2

Background

In this chapter, we first outline what techniques can be employed for the task of indoor localisation (Section 2.1). In Section 2.2 we give some high-level information on the regression algorithms we experimented with for our localisation model, and in Section 2.3, we outline the search algorithms we used for our pathfinding implementation. Some related work is discussed in Section 2.4.

2.1 Indoor localisation techniques

Localisation is the task of determining a person's or an object's location in an indoor or outdoor environment. The Global Positioning System (GPS) is used extensively for applications outdoors, where it provides excellent accuracy, according to Mainetti et al. [22]. However, according to Al Nuaimi and Kamel [1], GPS does not provide the same level of accuracy indoors. As GPS is low-power, if the line of sight to the GPS satellites is blocked by obstacles such as walls or there is presence of noise caused by equipment indoors, its performance degrades significantly.

There have been multiple solutions proposed for the task of indoor localisation. For example, there has been extensive research in localisation involving RFID (Radio-frequency identification) technology, as described by Li and Becerik-Gerber [19]. However, the use of RFID requires specialised equipment (readers & tags), making it less suitable for applications where the localisation service is intended for use by the general public. As Wi-Fi networks have become ubiquitous in indoor environments, use of data from Wi-Fi access points for the purpose of indoor localisation is increasingly common in the literature, as noted by Mainetti et al. [22]. The use of Wi-Fi signals for localisation typically does not require installation of specialised equipment, helping keep costs low and making access to the system simpler, according to Retscher and Leb [27].

There are different approaches used for localisation. There is no single better approach, as the choice depends on the requirements for the localisation system, the characteristics of the environment where it is carried out and the type of data or equipment that is available. Retscher and Leb [27] summarised the most common approaches, some of

which are the following:

- **Lateration:** A geometric approach where the signal strength from at least three access points is used to calculate the distance from each of them. These distances are then used to calculate the estimated location. It is easy to implement as it does not require any training or data collection. However, it requires line of sight to the access points which is difficult in environments with many obstructions, such as the University Library. Prior knowledge of the locations of access points is also needed. It can be done with RSSI (Received Signal Strength Indicator) or RTT (Round-trip time) data.
- **Angulation:** A similar approach to lateration, using the angle of the path from the current location to the access points instead of the distances. It requires line of sight to the access points and knowledge of the incidence angle.
- **Fingerprinting:** Signal strength readings at multiple locations in the indoor space are taken in advance to construct a *fingerprint database*. A new reading is compared against the readings in the fingerprint database to produce an estimated location. Its main benefit is not requiring line of sight to the access points. It has been found to perform better than the Lateration or Angulation techniques, according to El Ashry and Sheta [9].
- **Scene analysis:** Similar to fingerprinting, in the sense that a database is collated in advance. However, it uses image data from the environment collected using cameras. A new image is then compared against the images in the database to determine the location. It is computationally expensive, requires large storage for the database and involves larger data transfers than the RSSI-based fingerprinting.

The large number of Wi-Fi access points (30-40 per floor) already installed in the University Library, the lack of line of sight requirement, and the need to have sufficiently high accuracy to guide users to books make fingerprinting the most suitable approach for our project.

2.2 Regression algorithms

Regression is the task of modelling the relationship between some independent and one or more dependent variables. For the localisation task, we seek to build a model mapping the Wi-Fi RSSI readings to the coordinates where the reading was taken. In this section we give some high-level information on the regression algorithms we used during our experimentation discussed in Chapter 3. These are Nearest Neighbours regression (Section 2.2.1), Random Forest regression (Section 2.2.2) and Gaussian processes (Section 2.2.3).

2.2.1 Nearest Neighbours regression

Nearest neighbours regression is a non-parametric model where the target variable is predicted to be equal to the mean or a weighted mean of the K nearest neighbours, as stated by Russell and Norvig [28], where K is a hyperparameter of the model. The K

nearest neighbours are found using some similarity metric on the feature vectors of the data points, such as the Euclidean distance. Using a weighted mean can ensure that closer points have larger influence on the predicted value. This can be accomplished by assigning to the target variable of each of the K neighbours, weight equal to the inverse of the distance to that neighbour.

2.2.2 Random Forest regression

Random Forest regression is an ensemble regression algorithm. It constructs multiple Decision Tree regressors, each on a different random subset of the features. The output is the average of the outputs of the individual Decision Tree regressors [14]. Hyperparameters such as the number of decision trees making up the random forest and the maximum depth of those trees can be controlled.

2.2.3 Gaussian process

A Gaussian process (GP) is a non-parametric Bayesian model that can model complex functions. Functions can be thought of as an infinite-dimensional vector, where each value is the output of the function for a particular input configuration. Gaussian processes model as a Multivariate Gaussian Distribution the output for a subset of the feature space, as discussed by Rasmussen and Williams [26]. For the regression task, it is sufficient to only consider the values of the function corresponding to the feature vectors within our training set and the testing set for which we want to make predictions.

The first part in constructing a Gaussian process is to define the prior distribution of the function values. Usually, the prior is defined as a zero-mean distribution - but any mean can be selected. A zero-mean prior for a function f is given by:

$$p(\mathbf{f}) = N(\mathbf{f}; \mathbf{0}, K) \tag{2.1}$$

where $f_i = f(\mathbf{x}^{(i)})$ and K is the covariance matrix. The covariance matrix is typically defined using a kernel function. One common kernel function used is the Radial Basis Function (RBF) kernel. With the RBF kernel, the covariance $K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ between the function outputs f_i and f_j corresponding to inputs $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ is given by:

$$k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\frac{||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||^2}{2l^2}\right) \tag{2.2}$$

where l is the length scale of the kernel. Generally, in the RBF kernel, as can be seen from Equation 2.2, the closer the feature vectors of between two data points are, the higher the covariance of their target variables.

Gaussian processes assume that each observed target value y_i from the training dataset is noisy, and is sampled from the distribution $N(f_i, \sigma_y)$ where σ_y is the noise level. Using these noisy observations, after some Linear Algebra, as described by Görtler et al. [16] and Schulz et al. [29], we can calculate the posterior distribution over the function values. The predicted output for an input feature $\mathbf{x}$ is normally distributed with

mean $\mu(\mathbf{x})$ and variance $\sigma_i^2(\mathbf{x})$ where $\mu(\mathbf{x})$ is the mean of the GP at $\mathbf{x}$ and $\sigma_i^2(\mathbf{x})$ the variance at $\mathbf{x}$, found during the calculation of the posterior.

As usual, the hyperparameters which include the noise level σ_y and any hyperparameters of the kernel, such as the length-scale l for the RBF kernel, can be optimised using cross-validation.

2.3 Path search algorithms

Path search is the task of finding a path from some source to a destination along some graph or a grid. We used two path search algorithms as part of our pathfinding approach described in Chapter 4. These are Dijkstra's algorithm (Section 2.3.1) and the A* search algorithm (Section 2.3.2), which we briefly describe in this section.

2.3.1 Dijkstra's algorithm

Dijkstra's algorithm is an algorithm that can find the shortest path between nodes on a graph [17]. The destination node does not need to be fixed. For example, the algorithm can terminate once it reaches a node satisfying a particular property. At each iteration, it expands the neighbours of the current node, keeping track of the paths traversed so far, as well as their associated costs. In the following iteration, the head of the path with the lowest cost so far is expanded first.

2.3.2 A* search

A* is a best-first search algorithm, which means that it attempts prioritise expanding nodes it estimates being closer to the destination. It follows the same overall approach as Dijkstra's algorithm. The difference is that when choosing which path to expand, it considers the sum of the cost of the path so far and an estimate of the cost from the current node to the destination. This estimate is calculated using a heuristic function, such as the Euclidean distance between the current node and the destination. Because of the use of the heuristic, the precise destination must be determined in advance. A* can find the shortest path faster compared to Dijkstra's algorithm, especially on a large graph, as discussed by Rachmawati and Gustin [25].

2.4 Related work

In recent years there has been increased interest in providing location-based services in indoor environments such as libraries, hospitals, airports or museums.

There is ongoing work to develop a smartphone-based navigation and information service at the library of the Vienna University of technology, by Retscher and Leb [27]. Their app involves localisation, using the fingerprinting approach with signal strength readings from nearby access points. Their published work so far focused on experiments with signal strength data collection. They compared the signals received when the device receiving the signal is static or in motion, and found very small differences. They also

compared the results from using different devices. They attributed small differences in RSSI results, to the different Wi-Fi scan duration of the different devices. The differences were more pronounced when the scan was made while the device was in motion.

A campus-wide localisation and navigation system, called MazeMap was developed by Biczók et al. [5] and was deployed at the Norwegian University of Science and Technology. They reported that 10% of the University's students and staff were interacting with the app on a daily basis, indicating that there is demand for location-based services. Their focus was navigation across the campus on a room-to-room basis. They used trilateration for localisation, as they considered it sufficient for location estimation at room-level precision. It was easier to implement on a large scale, as it does not require offline training of a model, as in the case of fingerprinting. In this case the system works across all 60 buildings of the campus. They achieved localisation accuracy between 5 and 10 metres.

Van Haute et al. [31] explored the use of a localisation system in the context of a healthcare environment, at the 'Sint-Jozefs kliniek Izegem' hospital in Belgium. They expect such a system to provide multiple benefits to a hospital. For example, it can give vulnerable patients more freedom to move around the space, as staff can keep track of their location, and it can enable implementing a system that can notify the nurses or doctors who are closer to a nearby emergency. They explored different approaches to using RSSI data and explored the impact of different access point configurations in the space. They also evaluated the use of different technologies, including Wi-Fi access points and Zigbee nodes. They found that fingerprinting with Wi-Fi signals provided the best results, due to the larger range of Wi-Fi access points. Their system provided mean error of 1.21 metres.

Wang et al. [32] conducted a user study to evaluate the usability of a navigation system in Vienna's central railway station. In particular, they focused on the ability of different population groups to complete certain navigation tasks using the system. They found that while younger people could use the app with ease, older participants struggled to follow the instructions. They conclude that a navigation system to be successful and accessible to all, it should by complemented by physical navigational aids in the space.

Comparing results from different localisation models is difficult. Despite efforts, such as the fingerprint dataset published by Montoliu et al. [23], there is no common widespread approach or datasets for evaluating and benchmarking localisation algorithms. The score achieved depends on the the environment, the presence of obstructions and whether these obstructions are stationary. Nevertheless, as we will discuss in Chapter 6, the data collected can have a major impact on the results, which means that a different model might be suitable for different contexts. In addition, it is not always desirable to strive for the highest accuracy possible. Depending on the requirements, sometimes it is better to accept lower accuracy in favour of easier deployment on a larger scale, as in the example from Biczók et al. [5]. For our project, high accuracy is desirable to achieve guiding users through tight spaces such as library bookshelves.

Chapter 3

Localisation

The most important piece of functionality of the app is displaying and estimating the current location of the user in the building. Other features of the app depend on a reliable and accurate localisation model. For our localisation model, we follow the fingerprinting approach as it does not require line of sight and has been proven to perform better in the literature, as discussed in Section 2.1.

In this chapter we describe how we collected the data for this task and what the format of the data is (Section 3.1). The localisation model takes the signal strengths received from the different access points as features, and predicts the coordinates of the location where the readings were taken. We define the cost function we are using for adjusting the models' hyperparameters and comparing the results (Section 3.2) and describe a simple baseline model that all subsequent iterations of the model are compared against (Section 3.3). Then, we discuss the preprocessing steps we experimented with in Section 3.4. We also compare two methods for increasing the density of our training data in Section 3.5. In Section 3.6, we outline the machine learning algorithms we considered and give a summary of our results and identify the model we chose for use with our app.

3.1 Data collection

An Android smartphone app was built for the purpose of constructing the fingerprint database. In the app, the user can import floor plans, take Wi-Fi readings using Android's WifiManager API and pick the location on a map of the building where the reading is taken. Details on how readings are taken in Android are discussed in Section 5.2.2. The User Interface of the app is shown in Figure 3.1. The readings are then stored in a JSON file, either locally on the device, or on a cloud storage solution such as Google Drive.

We refer to a signal strength reading as RSSI (Received signal strength indicator). The definition of RSSI varies depending on the manufacturer of the device receiving the signal. In Android's WifiManager API, RSSI is measured in dBm, a unit of power. The lower bound of RSSI is -100 dBm, below which the signal is undetectable by a typical Android device. The upper bound is 0 dBm - the closer to 0 dBm the reading,

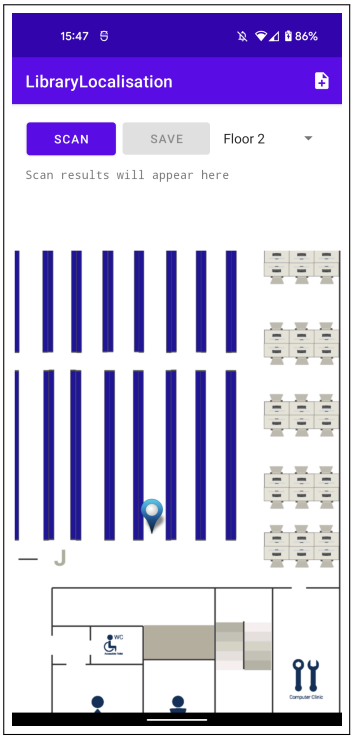


Figure 3.1: Screenshot showing the user interface of the Android application used for RSSI data collection

the stronger the signal. Stronger signal indicates closer proximity to the access point. In practice, the readings we observed ranged from -90 dBm to -20 dBm.

For developing and evaluating the localisation model we collected the following sets of readings:

- Readings from a section of the second floor of the University Library collected in December 2021 - used for training and adjusting the hyperparameters of the model
- Readings from the Informatics computing lab at the ninth level of Appleton Tower - used for some earlier experimentation work and for a brief evaluation of the possibility of applying the same localisation approach at a different environment (Section 6.2).
- A set of readings from the same section of the second floor of the Main Library collected in March 2022 - used as a test set for evaluation (Section 6.2).

Each RSSI value received is associated with an interface of an access point, denoted by its BSSID (Basic Service Set ID). BSSID is the identifier for an interface of an access point. Figure 3.2 shows an example of readings received by one interface of an access point. In some readings, no signal was received from the interface, even though in some cases signal was received in adjacent locations, suggesting that signal can be momentarily lost.

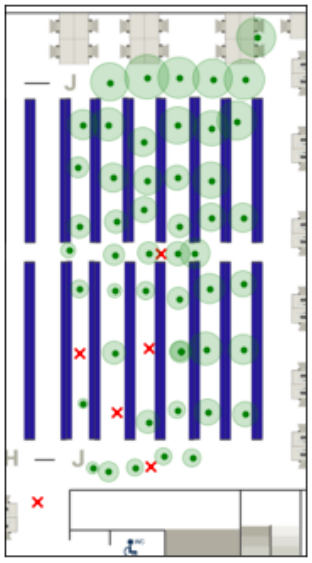


Figure 3.2: Example of RSSI readings at different locations received from a specific access point interface. Green circles at reading locations indicate the RSSI value - larger area means larger RSSI. Reading locations where no signal was received are marked with a red cross.

3.2 Cost function

For adjusting the hyperparameters of our models and reporting the error using cross validation we use the Root Mean Square Error (RMSE). We define the error for each individual prediction as the Euclidean distance between the estimated location's coordinates and the actual coordinates. This is a widely used metric in the literature for comparing the performance of localisation models [30, 33]. Therefore, in order to calculate the performance of a model M with hyperparameters θ with a validation (or test) set $\mathcal{S}$ with N examples we have:

$$E(M;\theta) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[(x_i - x_i')^2 + (y_i - y_i'^2)\right]}$$

where (x_i, y_i) are the actual coordinates where reading i was recorded and (x_i', y_i') are the coordinates of the reading as estimated by the model M.

When cross-validation is used, the error reported is the mean error across all iterations of cross-validation.

3.3 Baseline model

Before proceeding with any experimentation, we first implemented a simple baseline model in order to be able to assess the benefit of any preprocessing steps and any other machine learning algorithms.

We used a Nearest Neighbours regressor as a baseline model. Before applying it we

only did minimal preprocessing. This was shifting the reading values up by 100 so that they fall in the range of 0 to 100, and replacing all missing values with 0. This is equivalent to the assumption that whenever signal is not received, the relevant location is out of range of the access point, something that in reality is not the case, as signal can sometimes be momentarily lost.

Using 10-fold cross-validation, we optimised the number of neighbours K and decided whether predictions should be calculated using the ordinary mean or a weighted mean with the inverse of the distance between the neighbours and the new data point as weights. The best performance was given by $K = 7$ and the weighted mean. The RMSE is 2.62 metres, calculated by averaging the RMSE from the 10 iterations.

3.4 Preprocessing

In order to improve the performance of our model we experimented with different preprocessing techniques, described in this section.

Firstly, we shift all RSSI readings upwards by 100. This ensures all readings are between 0 and 100, which means that 0 represents the case where no signal has been received from an interface of an access point.

In Section 3.4.1, we describe an approach to aggregating highly correlated features. This also has the welcome side-effect of reducing the number of missing values from the dataset.

Then, in Section 3.4.2 we describe our slightly modified version of a Nearest Neighbour imputer, to estimate some more of the missing readings. However, we find that the imputer improves error only when used without the aggregation step, and increases the error when used after aggregation. Therefore, we omit this preprocessing step in our subsequent analysis.

The remaining missing signal strengths are marked as zero, indicating that no signal was received.

3.4.1 Aggregating RSSIs from interfaces of the same access point

An access point can be part of multiple networks at the same time, participating in each of those networks with a different BSSID. For example, in the University Library, the networks with SSIDs ‘central’ and ‘eduroam’ share the same networking hardware. This means that in the results from a Wi-Fi scan, RSSI readings corresponding to different BSSIDs, might have originated from the same access point. By observing the correlations between the readings corresponding to the different interfaces (shown in Figure 3.3), we can see that there are many groups of access point interfaces whose readings are highly correlated - they have correlation very close to 1. These groups can be seen in the correlation matrix between the RSSI values of different interfaces in Figure 3.3 as yellow squares.

Table 3.1 shows the BSSIDs and SSIDs of one of those groups of interfaces with highly correlated RSSI values. The RSSI values are almost identical in all readings

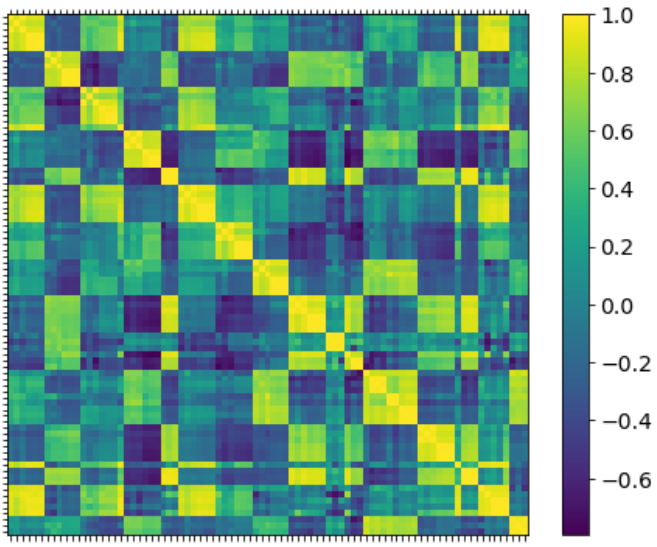


Figure 3.3: Correlation matrix of RSSI values corresponding to the different access point interfaces. Interfaces which were observed in at least 10 readings are included. Interfaces are ordered alphanumerically along the axes based on their BSSID

we collected for these interfaces. Their BSSIDs have a common prefix, a pattern we observed for other groups of interfaces with highly correlated RSSIs. This suggests that interfaces of the same access point have been configured to have a common prefix. We can exploit this to detect more reliably the groups of interfaces with highly correlated readings. In the absence of this, we could use the correlation coefficient to detect the groups.

BSSID	SSID
bc:9f:e4:e1:96:c0	eduroam
bc:9f:e4:e1:96:d0	eduroam
bc:9f:e4:e1:96:c1	central
bc:9f:e4:e1:96:d1	central
bc:9f:e4:e1:96:d2	HANABI

Table 3.1: BSSIDs and SSIDs of one group of interfaces with highly correlated RSSI values

We can aggregate the readings for those interfaces by simply replacing them with a single feature that is equal to the mean of the non-null values of the readings for the interfaces in the group. This results in having a single feature for each physical access point, drastically reducing the number of highly-correlated features.

As mentioned in Section 3.1, it is not uncommon that access points might be missed and record no signal even though from other measurements nearby we know that they are in close proximity. Aggregating the readings in this way reduces the amount of null readings we have in our data, since the probability of erroneously getting no signal corresponding to any of the interfaces of an access point is small - unless there is a fault

with the access point itself. Figure 3.4 shows the aggregated readings for the access point with the interface from Figure 3.2. It shows that as a result of the aggregation, all missing values have been filled.

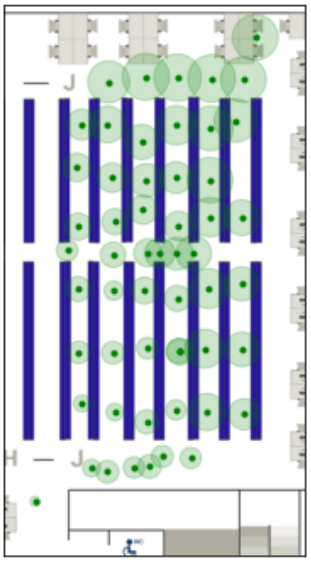


Figure 3.4: Example of the aggregated RSSI readings at different locations for the access point the interface from Figure 3.2 is part of. There are no missing values.

Aggregating access points in the way described in this section before using a Nearest Neighbours regressor improves the localization results. We achieved RMSE of 2.18 metres, with $K = 4$ and the predictions made using the weighted mean. This is lower than the 2.62 metre RMSE of the baseline.

3.4.2 Nearest Neighbours imputer

Beretta and Santaniello [4] state that Nearest Neighbour imputation algorithms can fill in missing values by estimating them using related entries elsewhere in the training dataset. The most common approach to this, is calculating the similarity (using Euclidean distance) of the entry with the missing value to other entries in the dataset. Similarity is calculated using the non-null features of the entries. The K most similar entries are found and the missing value is replaced with an average of the respective values from the K most similar entries.

As in our case, the RSSI entries are associated with (x, y) coordinates we can use those to find the K most similar entries, by calculating the physical distance between the entries, instead of using the similarity between the non-null RSSI vectors. Then, we fill in the missing value with a weighted average of the signal strength with weights the inverse of the distance.

Using only the imputer for preprocessing without aggregating interfaces from the same access points, yields a small reduction in the error compared to the baseline. It results in 2.53 metre error, compared to the 2.62 metres of the baseline. However, applying the imputer after the aggregation described in Section 3.4.1, leads to a worse result

than applying only the aggregation. It increases the error to 2.36 metres, compared to the 2.18 metre error from Section 3.4.1. This is possibly because, as we discussed in Section 3.4.1, aggregation is effective for reducing the number of missing values by itself and missing value estimates through aggregation only use readings collected at the same location, rather than nearby locations, as in the case of imputation.

3.5 Data augmentation with interpolation

Given the collected set of readings, we can use interpolation techniques to calculate a radio map for each access point, in order to artificially generate signal strength readings and increase the density of our training data. We explored two ways of generating the radio maps. The first is by fitting a Path-loss function for each AP (Section 3.5.1), and the second is using Gaussian processes instead (Section 3.5.2). We compare the results of using these interpolation approaches with the baseline model described in Section 3.3.

3.5.1 Path-loss function

The path-loss function is a parametric model describing how the strength of a radio signal, such as Wi-Fi, changes as it propagates through air. Its general form is given by Yiu et al. [33] as:

$$\psi^*(\mathbf{z}) = C + \alpha \log_{10}(||\mathbf{z} - \mathbf{z}_{AP}||) + \beta(||\mathbf{z} - \mathbf{z}_{AP}||) \tag{3.1}$$

where $\psi^*(\mathbf{z})$ is the estimated signal strength at location $\mathbf{z}$, $\mathbf{z}_{AP}$ is the location of the access point and C, α, β are parameters. The locations $\mathbf{z}$ are represented by three-dimensional coordinates. For our readings with two-dimensional coordinates (x, y) we set $\mathbf{z} = (x, y, 0)$, assuming that all of our readings are taken at the same height.

We fit a path-loss function for each of the access points in our dataset, estimating $\mathbf{z}_{AP}$, C, α and β using the algorithm proposed by Yiu et al. [33]. In each iteration of the algorithm, C, α and β are updated by least squares. The location of the access point $\mathbf{z}_{AP}$ is updated using a random walk, rejecting the new $\mathbf{z}_{AP}$ if it results in higher error, and accepting it otherwise. Figure 3.5 shows the radio map produced by the path-loss function for the access point whose readings for one interface are shown in Figure 3.2.

We observed a large increase in the error compared to the baseline when using the fingerprint data generated from the path-loss functions for Nearest Neighbours regression (6.41 metres with $K = 15$ with one data point for every 0.25 square metre). This can be explained by the fact that the path-loss function makes the strong assumption that signal propagates evenly in all directions. This can be seen from the equation where the estimated signal depends only on the distance from the location of the access point and from Figure 3.5. Using the path-loss function to artificially generate training data, removes important information on how signal is affected by the many obstacles in the space, which is captured by the original training data.

Nevertheless, this could be a useful approach in parts of the building that are more open where signals propagate more evenly, such as the concourse of the library or in places

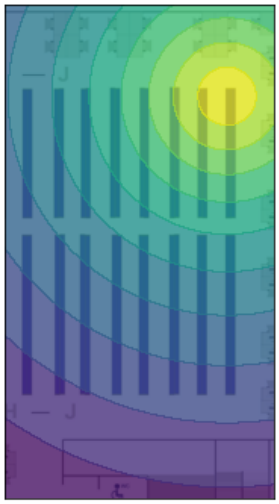


Figure 3.5: Contour map of the radio map generated for the access point from Figure 3.4 using a Path loss function. Yellow shading indicates stronger signal.

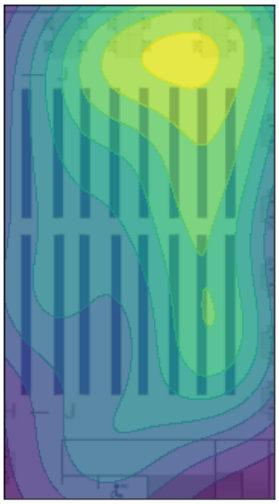


Figure 3.6: Contour map of the radio map generated for the access point from Figure 3.4 using a Gaussian process. Yellow shading indicates stronger signal.

like stadiums. In those areas, a much smaller number of readings could be taken, and then readings could be generated using the approach described in this section. This would allow resources for constructing the fingerprint database to be redirected in more complex parts of the building.

3.5.2 Gaussian processes

Another way to artificially generate fingerprint data, is by modelling the signal strength of each access point with a Gaussian process (See Section 2.2.3). Gaussian Processes are flexible, non-parametric functions that can capture the impact of obstacles on the RSSI values. We are using a Radial Basis Function (RBF) kernel and a zero-mean prior. The length scale of the RBF kernel and the noise of the observed RSSI values are fitted using maximum likelihood estimation.

Figure 3.6 shows the contours of the mean of the Gaussian process fitted for the same access point from Figures 3.4 and 3.5. Both the path-loss model (shown in Figure 3.5) and the Gaussian process (shown in Figure 3.6), place the access point at similar locations. However, in contrast to the path-loss model, it is able to capture the important features of the variation of the RSSI values in the space, such as the fact that signal is stronger in areas where there is line of sight to the access point, and weaker when the line of sight is interrupted by an obstacle, such as a bookcase.

We can use the Gaussian process models to generate data points along a grid, in a similar way we did with the Path-loss models. At each location in the grid $\mathbf{x} = (x, y)$ we sample the signal strength from each access point i from the normal distribution $N(\mu_i(\mathbf{x}), \sigma_i^2(\mathbf{x}))$ where $\mu_i(\mathbf{x})$ and $\sigma_i^2(\mathbf{x})$ are the mean and the variance respectively, of the Gaussian process at the coordinates $\mathbf{x}$. Using the generated data to fit a Nearest Neighbours regressor achieves RMSE of 1.99 metres, lower than the error with the

original aggregated data (2.18 metres).

3.6 Model selection

In this section we explore the use of other machine learning algorithms, other than the Nearest Neighbours regressor used for the baseline and for evaluating the preprocessing and augmentation techniques in the previous sections. For the models in this section the preprocessing used involves shifting the RSSI values up by 100, aggregating RSSI values from interfaces of the same access point and filling in the rest of the missing values with 0, as we described in the preceding sections. We first describe the use of a Random Forest regressor 3.6.1. Then we describe an approach of using the Gaussian processes modelling the signal of each access point to predict the location of a new reading using maximum likelihood estimation (Section 3.6.2). Then, we explore modelling the mapping of the signal strength vectors to the coordinates with a single Gaussian process (Section 3.6.3). Our results are summarised in Section 3.6.4.

3.6.1 Random Forest

A Random Forest regressor was considered as a candidate model. We optimized the maximum depth of the decision trees and the number of decision trees in the model using 10-fold cross validation. We compared the results using both the original signal strength data, as well as the artificial data generated during the GP-based data augmentation approach (Section 3.5.2). In both cases, we optimized the hyperparameters controlling the number of decision trees and the maximum depth of the trees with 10-fold cross validation. Using the original signal strength data the error is 2.50 metres (with 500 trees with maximum depth 10). Using the artificial data, the RMSE increases to 3.04 metres, in contrast to the Nearest Neighbour model where it resulted in a decrease.

3.6.2 Multiple Gaussian processes with MLE

We experimented with the approach proposed by Bekkali et al. [3]. It is a probabilistic approach where the objective is to find the location coordinates that maximise the probability that the given signal strength vector is observed. The coordinates that maximise the likelihood are the ones emitted by the model as the predicted location. More specifically, we seek to maximise the log-likelihood, which is given by:

$$l(\mathbf{x}) = \sum_{k=1}^{K} \log p_k(s_k \mid \mathbf{x}) = \sum_{k=1}^{K} \log N(s_k;\, \mu_k(\mathbf{x}), \sigma_k^2(\mathbf{x})) \tag{3.2}$$

where $N(s_k;\, \mu_k(\mathbf{x}), \sigma_k^2(\mathbf{x}))$ denotes the probability density function of the Normal distribution with mean $\mu_k(\mathbf{x})$ and variance $\sigma_k^2(\mathbf{x})$, $\mu_k(\mathbf{x})$ and $\sigma_k^2(\mathbf{x})$ are the mean and variance of the Gaussian process for the k-th access point at the input coordinates $\mathbf{x} = (x, y)$, s_k is the RSSI value recorded for the k-th access point and K is the number of access points detected by the device.

This model assumes that RSSI values from different access points are independent, something which is likely to hold after the aggregation preprocessing step.

With this model, we achieved a 2.56 metre RMSE, which is similar to the performance achieved by the baseline.

3.6.3 Single Gaussian process

This approach involves modelling the mapping from the signal strength vectors to the location coordinates with a single Gaussian process (Section 2.2.3), instead of modelling the signal of each access point with a separate Gaussian process, as we did in Section 3.6.2. We set the mean of the prior distribution of the Gaussian process to be equal to the mean of all the coordinates in our training set. We used a Radial Basis Function kernel, and fitted the noise level of the observations and the length scale of the RBF kernel with 10-fold cross validation.

With this model, the lowest cross validation error achieved was 1.70 metres, lower than any of the models considered so far. This was given by the Gaussian process with length scale 55 for the RBF kernel.

3.6.4 Summary of results

Table 3.2 summarises the results for the different models we described in this section. Overall, Nearest Neighbour models can perform well, with appropriate preprocessing. The Nearest Neighbour model with the aggregation preprocessing step and the GP-based data augmentation is the second best performing model with 1.99 metres RMSE. Random Forest regression models, give worse performance compared to the corresponding Nearest Neighbour models with the same preprocessing, indicating that the Random Forest might not be a good fit for this data.

The best performing model is the single Gaussian process, described in Section 3.6.3, that maps the entire signal strength vector to the location coordinates. We selected this model to be deployed for use with our app.

Model	Preprocessing	Augm.	RMSE (m)
Nearest neighbours	Minimal (baseline)	-	2.62
	Aggregation	-	2.18
	Imputation	-	2.53
	Aggregation & Imputation	-	2.36
	Aggregation	Path-loss	6.41
	Aggregation	GP	1.99
Random Forest	Aggregation	-	2.50
	Aggregation	GP	3.04
Multiple GP with MLE	Aggregation	-	2.56
Single GP	Aggregation	-	1.70

Table 3.2: Summary of localisation results for the different models

Chapter 4

Pathfinding

After the user has been given an estimate of their location, they may wish to go to a point of interest of their choice. This might be, for example, a study space or a specific book. The pathfinding module is responsible for calculating a path from their location to the selected destination.

In this chapter we discuss the different parts of the pathfinding system. First, we describe our approach to determining which areas of the library are passable (Section 4.1). Our approach for pathfinding follows the general approach proposed by Zhou et al. [36], which involves first extracting a topological map from the floor plans for each floor. The portion of the space forming the topological map is discovered using the Zhang-Suen thinning algorithm (Section 4.2), and based on this we construct a graph representation of the topological map (Section 4.3). This topological map is then used to find paths from any given point to a destination of the user's choice (Section 4.4).

4.1 Detecting passable space

Building Information Modelling (BIM) systems are becoming increasingly adopted in the construction industry. They provide detailed digital representations of indoor environments, providing geometric as well as semantic information about the environment. Lin et al. [20] propose a way of extracting information about which parts of a building are passable using BIM systems, as well as assigning different 'risk levels' to the obstacles depending on the nature of the obstacle. This allows, for example, to assign doors as obstacles with low risk as we can be confident that a person would be able to open it and walk through.

We do not have access to BIM information for the University Library building. Therefore, we follow a more simplistic approach only using 2-dimensional floor plans which are provided to us as image files. For each floor, we split the space in a grid with square cells of fixed size. We assume that space that is passable is denoted by white pixels on the image. If all cells in the pixel are white, the cell is marked as passable. Otherwise, it is marked as occupied and paths cannot pass through it.

This approach means that some adjustments might have to be made to the floor plan,

depending on the way objects are represented in it or any additional information that might be present on it. In the case of the library, doors are denoted by gaps on the walls, so no adjustment needs to be made in that regard. If a door symbol was used, as is the case in most architectural drawings that would need to be removed. Some of the floor plans had additional text labels to give information about the different areas and rooms of the floor. These need to be removed before establishing which cells are occupied. For this to be reliable, the floor plan needs to be consistent with the actual layout of the space, which means that it needs to be updated if there are any changes.

We denote occupancy by a binary variable O_{ij} with value 0 if the cell with cell coordinates (i, j) is passable and value 1 if it is occupied.

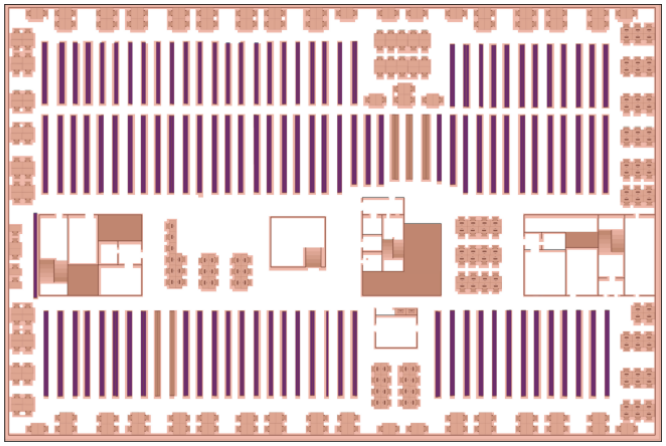


Figure 4.1: Occupied cells detected using cell size of 30 centimetres

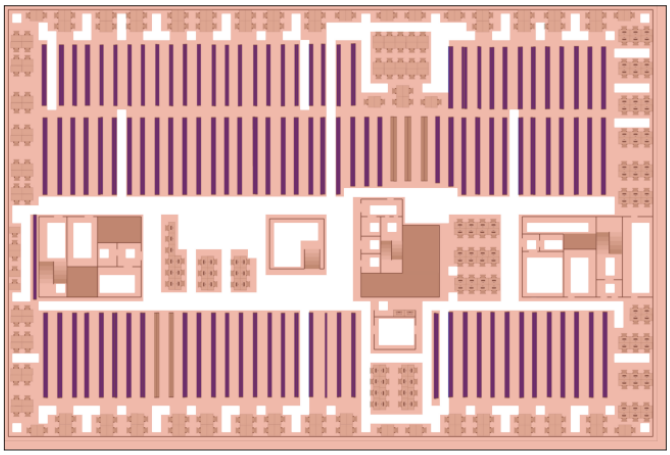


Figure 4.2: Occupied cells detected using cell size of 1 metre

Figures 4.1 and 4.2 show the results using two different cell sizes: 30 centimetres and 1 metre respectively. Using a larger cell size can cause larger areas to be marked as occupied in the presence of small obstacles, while smaller cell sizes can increase the runtime of the algorithms we will describe in the following sections. As Figure 4.2 with a larger cell size many areas of the floor can be marked as occupied, such as the narrow

corridors between bookshelves, significantly limiting the ability of the pathfinding system to produce sensible paths. Therefore, cell size should be chosen carefully.

4.2 Zhang-Suen thinning algorithm

Thinning algorithms transform a binary image to a more compact representation, while retaining details and meaningful features, according to Zhang et al. [34]. For example, in a binary image where the subject of the image is represented by white pixels, it is reduced so that at any point, shapes represented by white pixels have single-pixel width, as shown in the example in Figure 4.3.

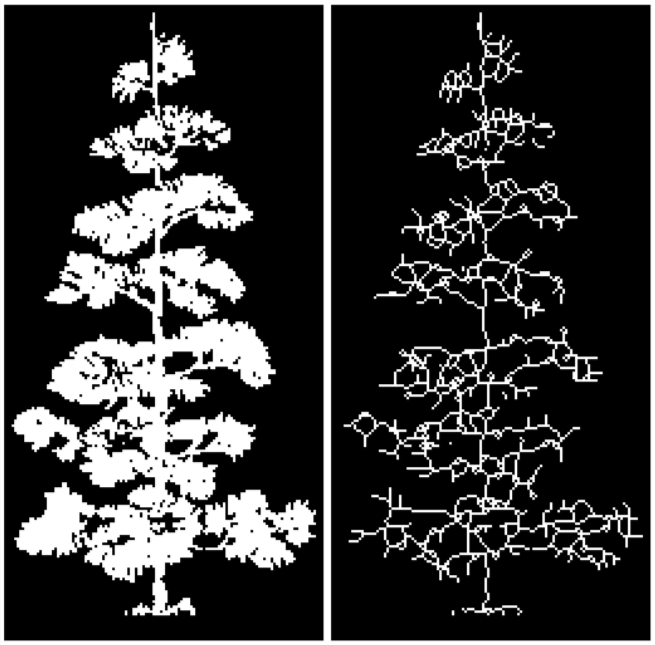


Figure 4.3: An example of the result after applying a thinning algorithm on a binary image (From MathCAD documentation [8])

The Zhang-Suen thinning algorithm, proposed by Zhang and Suen [35], is a popular thinning algorithm used extensively in the field of computer vision, for applications such as Optical Character Recognition and biometric authentication according to Bansal and Kaur [2].

For our purpose, the objective is to reduce the passable space to a skeleton, in order to then extract a topological map from it. Therefore, white pixels from the example above would correspond to passable cells and black pixels to occupied cells.

The algorithm operates on Passable cells, i.e. Cells (i, j) such that $O_{ij} = 0$ with eight neighbouring cells. The neighbours are numbered in the way shown in Figure 4.4. Using these neighbours of a cell (i, j) we define two quantities: A_{ij} and B_{ij}.

A_{ij} is the number of transitions from an occupied cell to a passable cell in the path

N_8 (i-1, j-1)	N_1 (i-1, j)	N_2 (i-1, j+1)
N_7 (i, j-1)	Current cell (i, j)	N_3 (i, j+1)
N_6 (i+1, j-1)	N_5 (i+1, j)	N_4 (i+1, j+1)

Figure 4.4: Numbering of neighbouring cells for the Zhang-Suen algorithm

$N_1N_2\cdots N_8$:

$$A_{ij} = \sum_{k=1}^{7} O_k(1 - O_{k+1}) \tag{4.1}$$

where O_k denotes the occupancy of the neighbour N_k.

B_{ij} is the number of neighbours of the cell (i, j) that are not occupied:

$$B_{ij} = \sum_{k=1}^{8} (1 - O_k) \tag{4.2}$$

Initially, we mark all passable cells as 'topological'. Zhang-Suen is an iterative algorithm with two sub-iterations. In the first sub-iteration, for every cell (i, j) still in our topological set of cells, we check if it satisfies all the conditions in the **Condition set 1**:

(a) $2 \leq B_{ij} \leq 6$
(b) $A_{ij} = 1$
(c) $O_1 \cdot O_3 \cdot O_5 = 0$
(d) $O_3 \cdot O_5 \cdot O_7 = 0$

We mark cells that satisfy the Condition set 1 to be removed from the set of topological cells, but do not remove them at this point. Then, in the second sub-iteration we do the same for the cells that satisfy all conditions in **Condition set 2**. Only the last two conditions differ from Condition set 1:

(a) $2 \leq B_{ij} \leq 6$
(b) $A_{ij} = 1$
(c) $O_1 \cdot O_3 \cdot O_7 = 0$
(d) $O_1 \cdot O_5 \cdot O_7 = 0$

Now, we remove all cells that have been marked for deletion from the set of topological cells. If no cells are removed and the set of topological cells remains unchanged, the algorithm terminates and outputs the resulting set of topological cells. Figure 4.5 shows the cells that the algorithm marks as topological in the grid from Figure 4.1.

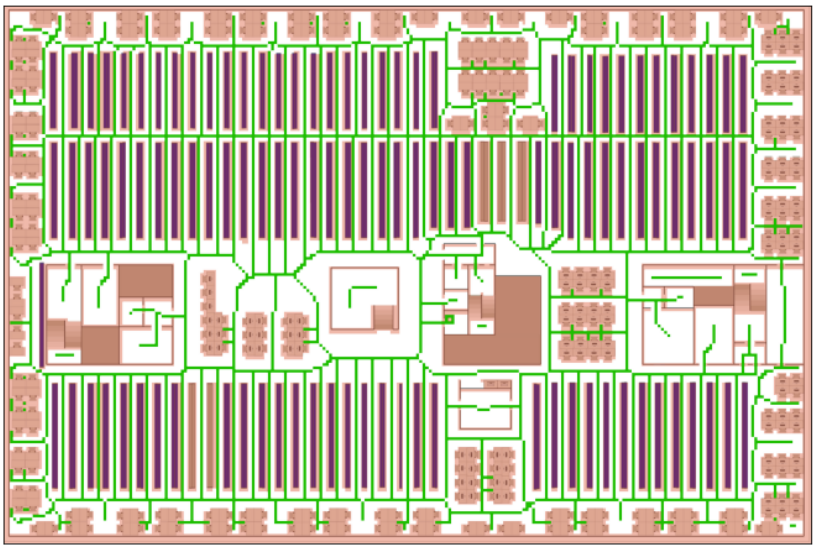


Figure 4.5: Topological cells (marked in green) extracted using the Zhang-Suen algorithm with cell size of 30 centimetres

4.3 Topological map graph representation

After discovering the cells that are part of the topological map, using the Zhang-Suen algorithm as described in Section 4.2, we need to link them to create a graph representation of the topological map. Zhou et al. [36] proposed an algorithm (Algorithm 1) that does this efficiently, which produces a combined grid and topological map that can be then easily used for path-finding.

The algorithm adds nodes in the graph when:

- A topological cell has only one topological cell neighbour (The cell is at the end of an edge)
- A topological cell has two topological cell neighbours but the cell is not along the same straight line with the two neighbours (The edge bends at the current cell)
- A topological cell has more than two topological cell neighbours (The cell is at the intersection of edges)

An edge is added between the nodes when there is a straight path of topological cells between them, as described in Algorithm 1. Figure 4.6 shows the result after applying the algorithm to the topological cells detected in the grid with 30 cm cell size (Figure 4.5).

4.4 Using the topological map for pathfinding

The pathfinding task can be split into three parts. First, a path from the start s to its nearest point on the topological map s' needs to be found. Then, we find a path from the nearest point to the destination on the topological map d' to the destination d. We

Algorithm 1 Generating a graph representation of the topological map (Adapted from Zhou et al. [36])

Input: Thinned grid map g
Output: Topological map in graph representation t

```
1:  t = ∅
2:  s = topological cell in g with 1 topological cell neighbour
3:  PROCESSCELL(g, t, s, s)
4:  return t
5:  function PROCESSCELL(g, t, s, c)
6:      N = untraversed topological neighbours of the current cell c
7:      if |N| = 0 then
8:          Add (s, c) to t
9:          return
10:     end if
11:     n = some element of N
12:     T_c = number of neighbours of c that are topological cells
13:     if T_c = 1 then
14:         Add (s, c) to t
15:         return
16:     else if T_c = 2 then
17:         if c, s, n are along the same straight line then
18:             PROCESSCELL(g, t, s, n)
19:         else
20:             Add (s, c) to t
21:             PROCESSCELL(g, t, c, n)
22:         end if
23:     else if T_c > 2 then
24:         Add (s, c) to t
25:         for n' in N do
26:             PROCESSCELL(g, t, c, n')
27:         end for
28:     end if
29: end function
```

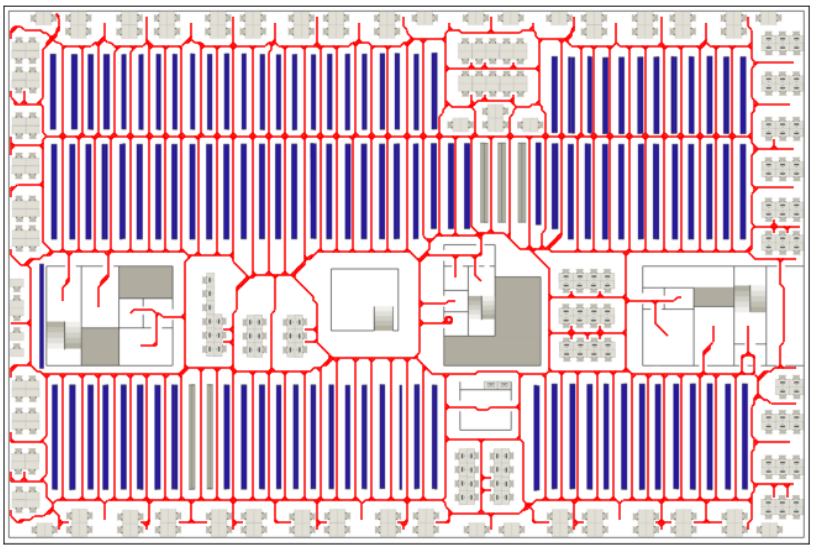


Figure 4.6: The resulting topological map with cell size of 30 centimetres

join these two parts of the path by using the topological map to find a path from s' to d'. The path that consists of these three parts gives us the full path from the start s to the destination d.

Given an arbitrary cell, to find its nearest cell on the topological map, we use Dijkstra's algorithm (Section 2.3.1) to find the shortest path from the cell to any topological cell. Transitions from cell to cell have different costs. Transitions in the four vertical or horizontal directions have cost equal to the cell size. Transitions to cells diagonally carry cost equal to the cell size scaled by $\sqrt{2}$. This step gives us the paths (s, s') and (d', d).

Sometimes, the current location can be at an occupied cell. This can happen because of large error from the localisation model, or sometimes the user can be at a study space, for example, which are usually marked as occupied spaces. In that case, we follow a similar approach to when finding the nearest topological cell, to find a simple path joining the current location to the nearest passable cell. This path is prepended to the path from the start to the topological map.

To find the shortest path from s' to d' we use A* search (Section 2.3.2) with the graph representation of the topological map, which completes the full path from s to d. An example of a path calculated using this approach is shown in Figure 4.7, distinguishing between the parts of the path found using the topological map and the parts found using Dijkstra's algorithm.

The A* search algorithm searches for a path using the nodes of the topological map. However, in most cases s' and d' are not nodes of the topological map. Instead, they can lie on an edge of the topological map. The naive approach to address this would be to create a temporary copy of the topological map's graph representation each time before applying the A* algorithm, and breaking up the edges s' and d' lie on to insert s' and d' as nodes of the graph. However, creating a copy of the data structure holding

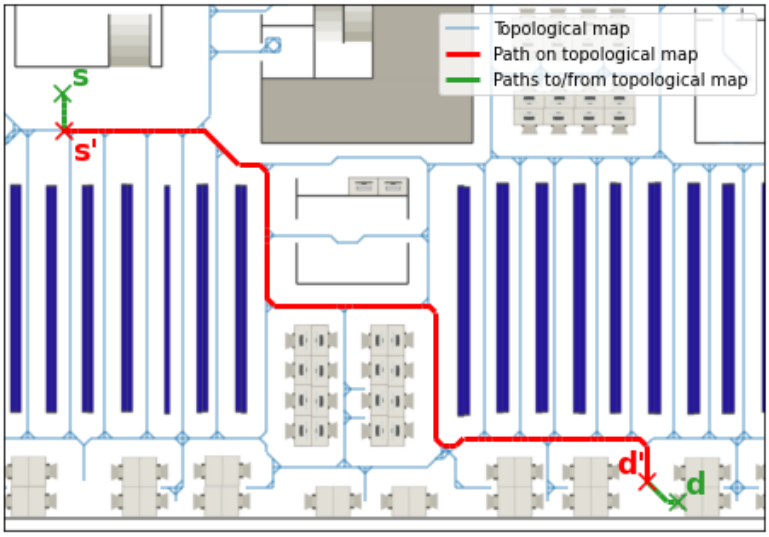


Figure 4.7: An example of a path calculated using the topological map approach. In red is the portion of the path that lies on the topological map (Path from s' to d') and in green the portions of the path connecting the topological map to the start s and the destination d

the topological map for each pathfinding request is computationally expensive, as the topological map can be big for a large indoor environment. A more efficient approach is to slightly modify the approach of finding the neighbours of a node in the graph, while keeping the graph representing the topological map intact. Before applying A* we find the edges that s' and d' lie on, which we denote by $\{u_1, u_2\}$ and $\{v_1, v_2\}$ respectively. We initialise the A* search with s', despite s' not necessarily being a node of the graph. Let c denote the current node that is being explored. We find the neighbours of c based on the following conditions:

- If $c = s'$, the neighbours of c are u_1 and u_2
- If $c \in \{v_1, v_2\}$, the only neighbour of c is d', which means that A* will terminate in the following iteration
- Otherwise, the neighbours of c are the ones as found from the graph representing the topological map.

Chapter 5

Implementation

A proof-of-concept Android app (Section 5.2) has been developed, in order to demonstrate the capabilities of the system and evaluate its effectiveness. The app receives input from the user, performs Wi-Fi scans and displays the localisation and pathfinding results. To fetch the localisation and pathfinding results, it interacts with a web server via a Web API (Application Programming Interface). The server is responsible for receiving the user's requests via the app, alongside the relevant readings and returning the output to the app, which then displays the results. Section 5.1 outlines how the server is structured and how it handles requests in order to produce the desired results.

The Android app has also been used for evaluating the system with the participation of users, details of which are given in Chapter 6.

5.1 Web server

In this section we outline key information and decisions about the server: our choice of web framework (Section 5.1.1), the kinds of requests it can handle (Section 5.1.2), the internal structure of the server (Section 5.1.3) and how the server was deployed so it can be accessed by users via the app (Section 5.1.4).

5.1.1 Framework

As we used Python for developing our models and we made extensive use of Python libraries, we decided to use a Python-based web framework for the server to allow reuse of existing code as much as possible. Flask and Django are the two most widely used Python-based web frameworks. According to Ghimire [15], Django is a 'batteries included' framework, with extensive functionality built-in, something useful for large applications. However, our web application does not require most of the additional functionality built into Django. As Flask's more minimal basic functionality was sufficient for our needs, we chose it for our server. Its simplicity made it easier and quicker to get our server up and running. Although it provides minimal built-in functionality, it works well with a variety of third-party libraries if in the future there is a need to develop additional features.

5.1.2 API Design

The API exposes an interface to perform the localisation and pathfinding tasks, and to return a list of destinations the user can select to find a path to. The routes the API provides along with the arguments they can take are the following:

/localise It receives the list of BSSIDs observed during a Wi-Fi scan alongside the corresponding RSSI values for each BSSID. It returns the estimated location which is defined by the appropriate floor identifier and a set of (x, y) coordinates.

/path It receives the user's current floor and coordinates and the destination. The destination can be defined in two ways. The first, is by its floor and coordinates. The second, is by its unique destination identifier which can be used to look up its coordinates and floor internally by the server. It returns the path as a sequence of coordinates.

/destinations It does not receive any arguments. It returns the list of all destinations in the library. For each destination information such as its unique identifier, name and other attributes (e.g. author if it's a book) are returned.

The API routes have been designed to be as simple and as flexible as possible - such as allowing a destination to be defined in two separate ways as described above. They only allow access to the necessary resources, such as the destination list which needs to be displayed by the application, and restrict access to information that the user does not need - such as individual reading entries from the fingerprint database.

By hiding implementation details behind the API, the server-side implementation and the Android application are as loosely coupled as possible. This means that they can be developed independently and it allows, for example, a complete rebuild of the application in the future, or integrating the localisation and path-finding functionality into existing systems, if that is desired.

5.1.3 Structure

The localisation and pathfinding functionality was developed before building the server. All that functionality was enclosed within a Python package, called 'libloc'. The package contains all methods and classes necessary for training the localisation model, making localisation predictions, building the grid-topological maps and finding paths.

For the interface of the 'libloc' package we followed the Facade design pattern, one of the Gang-of-four design patterns. Gamma et al. [13] state that the Facade pattern provides a single interface to all interfaces in a subsystem. In our package, the Facade class is the single point of entry for all interactions with the package. This reduces coupling between the Web API that receives requests from users, and the package, allowing both to be developed as independently as possible and making changes to the internals of the 'libloc' package easier.

Figure 5.1 summarises the structure of our web server. The Facade of the package provides an interface both for online actions that are called by the Web API upon request from the users, as well as offline actions. Online actions are localisation or

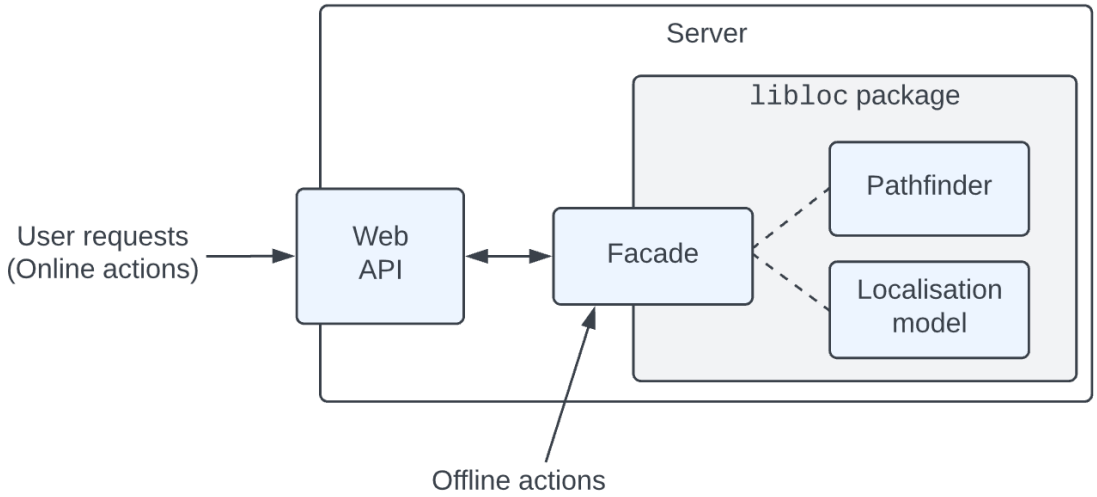


Figure 5.1: The structure of our web server. The user interacts with the web API that produces the results by interacting with the Facade class of the libloc package. The Facade also provides an interface for offline actions such as updating the model or generating the topological maps.

pathfinding requests. Offline actions include generating the grid-topological maps, importing fingerprinting data and training the localisation model. The Facade creates all necessary objects and calls the functions needed within the package to perform the requested action.

5.1.4 Deployment

Heroku is a Cloud Platform as a Service (PaaS) that allows hosting of web applications, while needing minimal configuration [7]. The web application is run in a container called ‘dyno’. It is compatible with Flask, the framework used for developing our web application.

We used the free tier of Heroku, which gives access to 1 dyno (equivalent to 512 MB of RAM) for 550 hours each month. It provides easy integration with GitHub, which we used to set up a pipeline to automatically deploy the server whenever changes are made to the main branch of the GitHub repository where the source code for the web application is hosted.

Since the server cannot be continuously active, it sleeps automatically after 30 minutes if it receives no traffic. When there is a request to the server while it is not active, there is a delay of about 30 seconds before the request is handled. However, at this stage of development we considered this to be sufficient. In our testing we did not notice any performance issues when the server is active and all requests are handled instantaneously. However, the server has not had to handle multiple concurrent requests thus far as the app has only been used by users as part of our evaluation.

5.2 Android application

In this section we give an overview of the intended use of this proof-of-concept application (Section 5.2.1), outline how Wi-Fi scanning is performed and discuss its limitations (Section 5.2.2). Then, we outline the key features of the User Interface (Section 5.2.3) and describe how the app interacts with the server-side API (Section 5.2.4).

5.2.1 App Overview

The main workflow of a user interacting with the app to find a book is as follows:

- The user requests an estimate of their location.
- The app conducts a Wi-Fi scan and interacts with the server to get the estimated location which is then displayed on a map of the building.
- The user selects from a provided menu the destination they wish to find - this can be a book, a study space or any other library resource.
- The app interacts with the server to calculate a path to the selected destination and shows the result on the floor plan.

The user is able to send additional localisation requests whenever they wish. If a path is displayed when the localisation request is sent, the path is updated accordingly to start from the new current location. In addition, the user can simply use the app to view the layout of the different floors of the building.

5.2.2 Wi-Fi scanning

In order to localise the user, their device needs to perform a Wi-Fi scan, to obtain a list of nearby access points alongside their respective RSSI values. This can be done using Android's WifiManager API [6]. This API can be used to trigger a new Wi-Fi scan, which then makes the results known through an asynchronous event. The event notifies the app if the scan was successful or not. If it was successful, the RSSI readings are collected and communicated to the server which is responsible for processing the readings and producing an estimate for the location. If the scan failed, an appropriate error message is displayed to the user.

To limit apps' consumption of battery life, each app is limited by the Operating System to initiating 4 scans per 2 minutes. Therefore, by default, our app only allows one scan every half-minute. We added the option to remove the 30 second delay in the app's settings which can be used when Wi-Fi scan throttling is disabled though the developer settings. While we cannot expect users to modify their device's developer settings, we made use of this option for constructing our fingerprint database and for our own testing and evaluation.

Android acknowledges the fact that a user's location can be inferred from Wi-Fi scan data and requires that the app has permission to access the user's precise location. Since Android classifies this permission as 'dangerous', the user has to explicity grant permission to the app. We followed all of Android's standards and guidelines to get

permission. This includes only asking for permission immediately before it is needed. Permission is seeked when a user first clicks on the button to request localisation. This is done through a dialog, shown in Figure 5.2, where the user has the option to grant permission to access their precise location, to their approximate location or to refuse permission altogether. If they refuse or only give permission for the approximate location, the Wi-Fi scan cannot proceed and the app cannot request and receive localisation results. In that case, we display a message explaining the rationale behind requesting permission for precise location with the option to go to the device's settings to grant permission, shown in Figure 5.3.

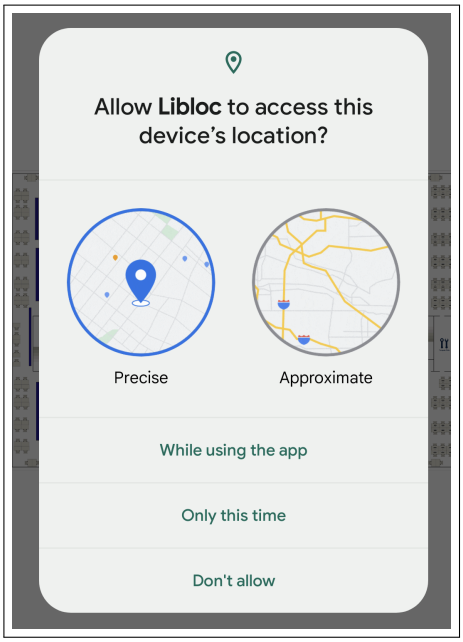


Figure 5.2: Prompt for receiving permission to access the device's precise location

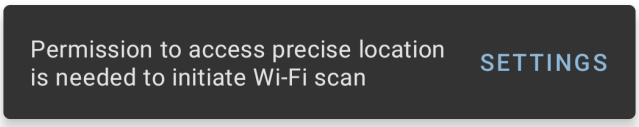


Figure 5.3: Message explaining the rationale behind requesting permission for access to precise location with a link redirecting to the relevant section in the device's Settings

Aside from the location permission, the app needs permission to access and change the Wi-Fi state to start a Wi-Fi scan, which is declared in the app's configuration files. However, Android does not classify these permissions as 'dangerous' and there is no need to request explicit permission from the user.

5.2.3 User Interface

The centerpiece of the app's user interface is a map of the building. It displays the layout of the floors, as well as the localisation and pathfinding results. Floor plans of the library building are available to us as image files. To display the pathfinding and

localisation results we need to be able to draw a pin and a path over the image. To improve usability, the map should support the pinch-to-zoom gesture.

We identified three libraries that could provide us with the base to build this functionality: SubsamplingScaleImageView [24], PhotoView [10] and AndroidImageMap [21]. We picked SubsamplingScaleImageView since it is more up-to-date compared to AndroidImageMap, which is no longer actively maintained, and its functionality better aligns with our requirements compared to PhotoView. Both SubsamplingScaleImageView and PhotoView provide the same basic functionality which is displaying an image which is zoomable through a pinch gesture. However, SubsamplingScaleImageView also adjusts the resolution of the image depending on the zoom level. This provides better performance, especially when using high-resolution image files, such as the library's floor plans. SubsamplingScaleImageView is also designed to be easily extended.

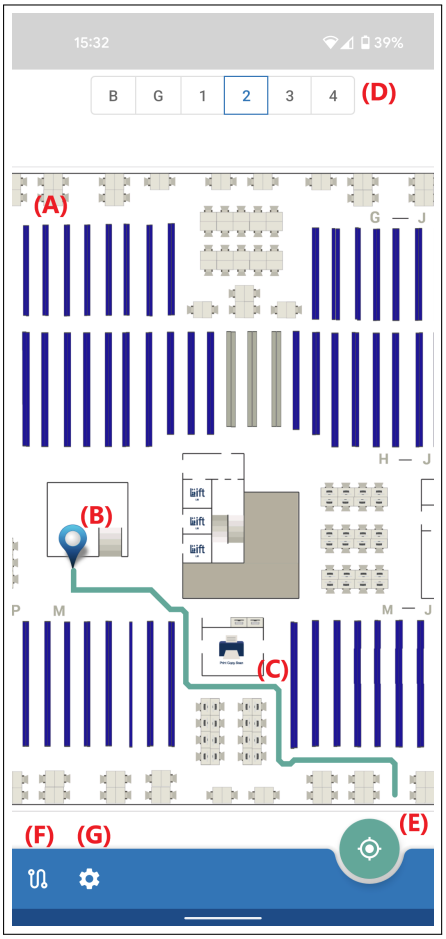


Figure 5.4: Screenshot of the app displaying the map (A) with the current location (B), a path to a user-selected destination (C), the toggle group for switching between floors (D), the localisation (E), the pathfinding (F) and the settings buttons (G)

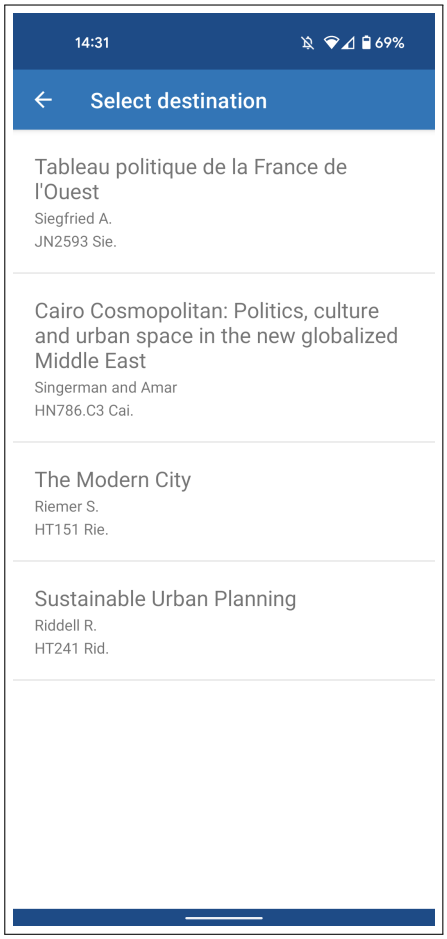


Figure 5.5: Screenshot of the list of destinations, displayed after the user taps on the pathfinding button

We extended the base class of SubsamplingScaleImageView to enable displaying elements on top of the map, using Android's Canvas library. This allows us to draw a path and a pin on the map, which are anchored on specific points on the floor plan and scale appropriately when the user zooms in or out. Figure 5.4 shows the map displaying a floor plan with a pin showing the current location and a path guiding the user to the destination of their choice.

At the top of the screen, a toggle button group allows the user to switch between different floors of the building, as shown in Figure 5.4. When clicked, the image displayed in the map changes to show the correct floor and the pin and the path are removed or added so that they are shown only on the correct floor.

At the bottom, an app bar contains the buttons that trigger a localisation and a pathfinding request, as well as the Settings of the app, which contain some options mainly used for debugging.

When a user taps on the pathfinding button, they are redirected to a list of destinations, populated with data fetched from the server. The data is displayed in a RecyclerView, a component that can efficiently display a list of items formatted in the same style, as shown in Figure 5.5. When an item is scrolled away, the corresponding component is reused for a another item that appears. This allows displaying a large number of destinations without an impact on performance. For each item, we display the name of the destination alongside some details for the destination below that. In the case of books, the name is their title and the details consist of the name of the author and their classification code which is displayed on the spine of the book, allowing users to easily locate their chosen book when they reach the correct location, as guided by the app. When a destination is selected, a pathfinding request is sent to the server, the user is redirected back to the main screen and the path is displayed on the map.

5.2.4 Interacting with the Web API

Volley [11] and Retrofit [12] are the two most commonly used Android libraries for sending HTTP requests. Our app interacts with the server via HTTP GET requests and receives the responses in the JSON (JavaScript Object Notation) format. Lachgar et al. [18] conducted a comparative study for Volley and Retrofit and concluded that while Retrofit is easier to use, Volley has an advantage in execution time. As our use case is straightforward, we did not encounter any issues with ease of use with Volley. As it is faster and it satisfies our requirements we decided to use it over Retrofit. Another advantage of Volley is its support of caching. For example, the list of destinations will not be to be fetched from the server each time the user wants to find a path, removing some load from the server.

Chapter 6

Evaluation

To evaluate the system, we conducted a user study, as described in Section 6.1. The study involved asking participants to use the app to find three books in a section of the University Library. The aim of the study was to establish how accurate users found the localisation results and whether they could successfully navigate around the space following the path provided through our app, by the pathfinding module of our system. We also evaluated how well our localisation model generalises, using a test set collected a few months after the data used to train the model was collected (Section 6.2).

6.1 User study

In this section we describe how the user study for evaluating the localisation and pathfinding systems was planned, as well as discuss the findings from the results. During the study users are asked to complete a sequence of tasks using the app. We evaluate the results through a questionnaire that they complete and using the signal strength data that we collect during their participation. First, in Section 6.1.1 we outline the changes we made to the app in order to conduct the study. In Section 6.1.2 we describe how the app was published and access was given to the participants and in Section 6.1.3 we discuss the tasks and the questions participants were asked to respond to. In Section 6.1.4 we summarise the findings from the study.

6.1.1 Preparing the app for the study

Before conducting the study, some small additions needed to be done to the app. As we are logging the requests the users send to the server during their participation in the study, we need a way to link their questionnaire responses with these requests. To do this, each participant is assigned a unique participant number (UPN).

To facilitate this, we have added a **/get_participant_number** route to the API of the server which randomly generates a four-digit UPN and returns it. The users can interact with this through the Settings activity of the app. There, we have added the option to request a UPN which when tapped it requests a UPN from the server and displays it, as

shown in Figure 6.1. This UPN is then added in the arguments of all localisation and pathfinding requests sent from the app to the server.

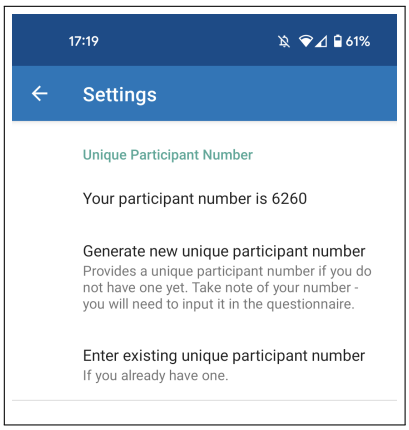


Figure 6.1: Part of the app's settings with the option to fetch a Unique Participant Number

6.1.2 Distributing the app

Android apps are typically distributed through Google's Play Store. However, it involves an approval process which might not have been completed in time to conduct the study.

Therefore, we distributed the app via an APK (Android Package) file, hosted on the server that the users can access through a URL shared with them through the questionnaire. Participants needed to allow installation of apps from unknown sources to complete the installation successfully. They were reminded to re-enable this setting once they finished the questionnaire.

6.1.3 Questionnaire design

The purpose of the survey is to evaluate the accuracy of the localisation model and the pathfinding system. More specifically, it will help us establish how good users perceive the results to be and to gauge what level of error is considered acceptable by the users. At this stage in development we are not evaluating the app in terms of usability of the User Interface. It suffices that users are able to interact with the UI to perform the necessary actions.

The platform we used for the questionnaire is Microsoft Forms. It allows branching, meaning that different questions can follow depending on the participants' responses to previous questions. Apart from the questionnaire, we also store the users' requests linked by the UPN. This allows us to compare the responses to the actual results that were available to the participants, and establish whether the performance of the localisation model degrades with the readings from the participants' devices.

The study involves the users completing a 'treasure hunt' style task. They are asked to start from a specific point, where they will find a particular book (Book 1). They use the app to detect their location. The questionnaire, which consists of six parts, guides the participants to find two other books (Books 2 & 3), where they will also use the

app for localisation. The first part is a set of preliminary questions followed by three localisation tasks with pathfinding tasks between them, as follows:

Setup At the start of the survey, the user is given instructions to download and install the app. They are asked if they were able to successfully install and launch the app. If they were not able to, they have the option to outline the issues they encountered, in order to enable us to resolve them, if possible. If they can access the app, they are asked to generate a Unique Participant Number (UPN) which they have to input in a text box provided in the questionnaire. Then, they are shown a description of the plan for the survey, giving information on the different parts of the survey and given instructions to go to the location in the second floor of the library where the survey starts.

First localisation (At Book 1) At the start of this part they are first asked what they would consider a reasonable error level for localisation in indoor environments, such as the University library. This is to get an insight on what they think is acceptable before, they actually use the app and see results. They select one of four options (Less than 1 metre/3 metres/5 metres/10 metres). This will allow us to see if this is contradictory to responses they give later on when they assess results from the app. Then, they are asked to interact with the app to request from it to detect their location. They are asked to respond with a 'Yes' or 'No' if the location looks right to them. Afterwards, they are asked to give an estimate of the size of the error, by picking one of 4 options (Less than 1 metre, 1-3 metres, 5-10 metres, more than 10 metres).

First pathfinding (From Book 1 to Book 2) For the first pathfinding task, they are asked to request a path to a specific book - the location where the second localisation task will be carried out. They are asked to follow the path to reach the book, and indicate in the questionnaire whether they could successfully reach it. Then, irrespective of whether they could successfully follow the path they are asked to comment on any issues with the provided path by selecting from a list of possible issues or writing a custom response. The list includes issues such as the path having an incorrect starting point (caused by a large error in the localisation in the previous part), the path being too circuitous or being hard to follow.

Second localisation (At Book 2) Before starting the second localisation task, the participants are asked if they were able to find the location of the book relevant to this task, irrespective of whether they could follow the path in the previous part. If they cannot find the location of the book, they cannot proceed to the next parts of the survey. Here, they also have the option to finish their participation if they wish to do so. The rest of this part is the same as the first localisation task, but at a different location.

Second pathfinding (From Book 2 to Book 3) The second pathfinding task is similar and involves the same questions as the first pathfinding task, but with the destination being a different book.

Third localisation (At Book 3) Then follows a third localisation task, similar to the second localisation task.

After the tasks have been completed, the participants have an opportunity to write any feedback they wish, either on the localisation, on pathfinding or on the app more generally. The full questionnaire is provided in Appendix C.

6.1.4 Results

Seven participants took part in the study, all of whom completed all tasks. It was challenging to recruit participants for this study, as it required them to be physically present at the University Library to complete the tasks. However, the participation has been sufficient to be able to extract some insights on how well the system performs in the real world, and get an idea of how accurate the users perceived it to be. No participant reported any issues with installing or setting up the app.

Acceptable localisation error The users were asked before interacting with the app, what is an error they would generally consider acceptable from a localisation service. Four of the participants responded with 'Less than 3 metres', and the remaining three with 'Less than 5 metres'.

Localisation results Figure 6.2 shows the locations of the three books that participants were asked to find during the study. In addition, it displays the estimated locations from our localisation model when the participants sent a localisation request while at each of the three books. There is more variability in the predictions along the aisles, than between aisles. This can be explained by the fact that bookshelves obstruct the signal from access points, resulting in different locations within the same aisle having similar values for the signals received and bigger differences between different aisles.

The overall RMSE we observed from the data we collected was 2.83 metres. The errors vary between the books. The highest error was noted with Book 3 (3.64 metre RMSE), whereas for Book 2 the error was lowest (1.71 metre RMSE), with Book 1 in between (2.81 metre RMSE). This is also reflected in the participants' responses. Two participants considered the prediction for Book 1 to be wrong, whereas only one said the same for Book 2. The prediction for Book 3 was considered incorrect by the majority of participants. Looking at the locations where the readings that are part the training dataset (see Figure 3.4), one can see that they are more sparse around the location of Book 3. At the time of the collection of the training data, little experimentation had been done at that point, which would inform a data collection strategy. It might improve results if a more structured approach is followed for data collection. For example, if a reading is collected at fixed intervals or if multiple readings are collected at each location. It is also worth examining whether the collection of data for a wider area, might improve results for Book 3, as it currently lies at the edge of the area where data was collected.

We observed that for some participants the error was higher than average for all three books. This could reflect the use of different devices by participants. As noted in Section 2.4, the received signal strength values can vary between devices. Participants with lower errors might have used devices receiving signal strength values, more consistent with the device we used for collecting the training data. In addition, the time of the day

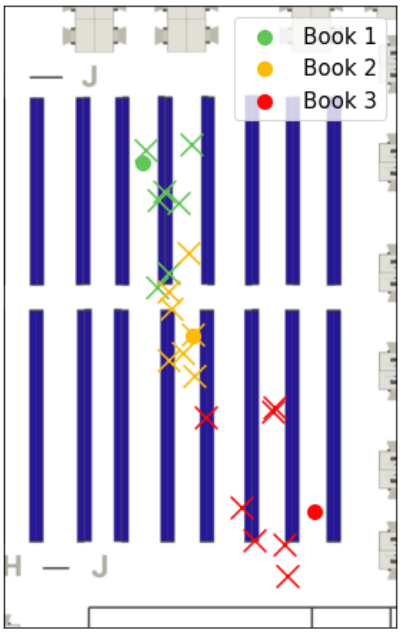


Figure 6.2: The actual locations of the three books (marked with circles) where participants were asked to use the app for localisation, and the estimated locations of the participants (marked with crosses)

could have an impact. Training data collection was conducted in the early morning, when the library was not busy. The two participants who completed the survey at a similar time, noted lower errors than average, indicating that the occupancy of the space could have an impact on the RSSI values. In light of this, compiling a training dataset with readings from different devices and collected at multiple times of the day, could improve localisation results.

Participants' judgement of localisation results An interesting observation is that users who responded with 'Less than 5 metres' for the error they considered acceptable, in some cases responded that the localisation result was not right, even though they judged the error to be less than 5 metres. This indicates, that depending on the context, users may have different tolerance for error. For example, if a user wants to see in which room of the building they are in, error of 5 metres could be acceptable, as the prediction will still be within the same room. However, if they are in parts of the building with bookshelves, where the aisles can be 1 metre wide, an error of 2 metres can be on a different aisle, a result which is not as useful. In fact, one participant mentioned this directly, that the error that is acceptable depends on the context.

Pathfinding results Users were able to successfully follow the path in all pathfinding tasks, with the exception of five, out of a total of 21 tasks. In all the cases where they were unsuccessful, the issue mentioned was the path having an incorrect starting point. No other issue was mentioned by the participants. This highlights the reliance of the pathfinding functionality on good localisation results.

High-level feedback Among the more high-level feedback provided, was a suggestion for the UI, that the map would be easier to use if it could be rotated. The app at this stage is intended as a proof-of-concept. More detailed work could be carried out in the future on the usability of the app, which should be evaluated in more detail. Another user mentioned that they enjoyed using the app and that they could see this being useful, particularly for new students at the University.

Overall, feedback was mostly positive. Participants were able to successfully complete most tasks using the app. There is room for improvement, as highlighted in the comments above.

6.2 Localisation evaluation with test set

We compiled a set of 14 readings from the same part of the library, where training data was collected. This test dataset was collected four months later, at a different time of the day, compared to the training dataset. We use it to establish whether the model generalises well, when the data was collected at a time when the space is busier with people and examine whether the performance of the model deteriorates, due to changes in the environment or networking configuration during those four months.

The RMSE with this test set was 2.88 metres. It is higher than the 1.70 metre RMSE we observed with cross-validation on the original dataset. Figure 6.3 shows the distribution of the errors for the test examples. While the error for most examples was between 1 to 3 metres, there was one example with error close to 6 metres. This training example was near the edge of the area where our training data was collected, where we also noted higher errors in Section 6.1.4.

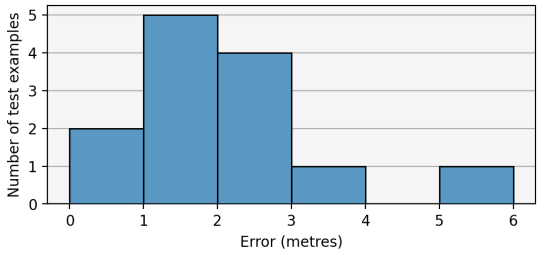


Figure 6.3: The distribution of the error for the test set examples

Evaluation in a different environment We have also collected data from a computing lab at the University. 70% of this dataset was used to train a model configured in the same way as the one created for the library, and the remaining 30% was used to evaluate this model. The main differences of this environment, compared to the library, is the presence of only two access points on the entire floor, lower than the number of access points in the part of the library we collected readings, and the presence of fewer obstacles that can obstruct Wi-Fi signals. This model achieves a 2.44 metre RMSE, indicating that the proposed model could be applied in other environments outside the library. This error cannot be compared directly with the test set error from the library, as that test set was collected separately from the training data.

Chapter 7

Conclusion

We summarise the main achievements of the work described in this report in Section 7.1. In Section 7.2, we outline some key challenges faced and lessons learned from our work. In light of these, we propose some next steps in Section 7.3.

7.1 Summary of work

The main achievements of this project can be summarised as follows:

- We built an Android smartphone application that can be used to construct signal strength datasets for localisation, which was used to compile three separate Wi-Fi signal strength datasets for localisation using the fingerprinting approach.
- We reported the impact different preprocessing and data augmentation approaches have on the performance of a localisation model for the University Library.
- We compared four machine learning algorithms for localisation and found a Gaussian Process model to provide the best cross-validation score, closely followed by weighted Nearest Neighbours regression when trained with artificial data generated through a Gaussian Process based data augmentation approach.
- We implemented a pathfinding algorithm for use in an indoor environment. To improve efficiency, a topological map is first generated using the Zhang-Suen thinning algorithm, before using it to find a path using a combination of A* search and Dijkstra's algorithm.
- We built an Android application that conducts Wi-Fi scans, receives input from users and interacts with a server containing the localisation and pathfinding modules to provide results.
- The localisation and pathfinding modules were evaluated through a user study, where participants navigated through the library to locate three books using the app, with mostly good results.
- We evaluated the localisation model with a test set collected separately from the training data, which gave a 2.88 metre RMSE.

7.2 Remarks

Our findings from this project, can help inform how the system can be developed further in order to be deployed at full-scale. Firstly, while our final model did not involve the data augmentation approaches that we explored in Section 3.5, we have observed good results with the second best performing model using it. Data augmentation can be a key tool to develop the model at a larger scale. It can be used to reduce the number of readings that need to be taken in parts of the building where there are fewer obstacles, such as the concourse, while still yielding good results, as Wi-Fi signals in such spaces propagate more evenly.

As we observed in the results of the user study in Section 6.1.4, while error can be low on paper, in reality the error needed to make the result useful varies in different parts of the building. It could be useful going forward, to define additional error metrics for specific parts of the building, such as whether the prediction is within the correct aisle and focus the work on improving the model in parts of the building where it has the biggest impact.

The user study conducted during this project, was focused on evaluating the localisation model and the pathfinding algorithm. For the app, as it was intended as a proof-of-concept, we considered it sufficient at this stage for users to be able to use it successfully to complete the study. A more extensive user study on the usability of the app, should be considered once it has been developed further.

7.3 Future work

With the comments made in Section 7.2 in mind, we propose the following topics to be included in future work:

Integration with existing systems The possibility of integrating the app with existing systems in the University Library, such as data occupancy information or the book database, could be explored.

Scaling-up the system Further work is needed to scale up the system. This involves further data collection across all floors of the building, in conjunction with the use of data augmentation approaches as discussed in Section 7.2. The performance of the localisation and pathfinding systems should be evaluated at a larger scale, and depending on the results, modifications might be necessary.

Updating the localisation model As we observed, in Section 6.2 the performance of the localisation of the model can degrade over time if the fingerprint database is not updated. Approaches to continuously update the model, such as crowd-sourcing or incremental training on feedback from users should be considered.

Additional app functionality Additional functionality could be implemented in the user-facing app, such as the ability for users to share their location with friends. Further work is also needed to improve its UI, and to make sure it meets accessibility standards. At a later stage a user study about the usability of the app should be conducted.

Bibliography

[1] K. Al Nuaimi and H. Kamel. A survey of indoor positioning systems and algorithms. In *2011 International Conference on Innovations in Information Technology*, pages 185–190, 2011.

[2] P. Bansal and B. Kaur. To propose an improvement in zhang-suen algorithm for image thinning in dip. *International Journal of Computer Science and Information Security*, 14(5):205–211, 2016.

[3] A. Bekkali, T. Masuo, T. Tominaga, N. Nakamoto, and H. Ban. Gaussian processes for learning-based indoor localization. In *2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pages 1–6, 2011.

[4] L. Beretta and A. Santaniello. Nearest neighbor imputation algorithms: A critical evaluation. *BMC Medical Informatics and Decision Making*, 16(S3), 2016.

[5] G. Biczók, S. Díez Martínez, T. Jelle, and J. Krogstie. Navigating mazemap: Indoor human mobility, spatio-logical ties and future potential. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, pages 266–271, 2014.

[6] *Android* developers. Wi-Fi scanning overview. `https://developer.android.com/guide/topics/connectivity/wifi-scan`.

[7] *Heroku* developers. Heroku documentation. `https://devcenter.heroku.com/categories/reference`.

[8] *MathCAD* developers. Example: Thinning and skeletonization. `http://support.ptc.com/help/mathcad/en/index.html#page/PTC_Mathcad_Help/example_thinning_and_skeletonization.html`.

[9] A. El Ashry and B. Sheta. Wi-fi based indoor localization using trilateration and fingerprinting methods. *IOP Conference Series: Materials Science and Engineering*, 610(1):012072, 2019.

[10] *Baseflow* developers. PhotoView. `https://github.com/Baseflow/PhotoView`.

[11] *Google* developers. Volley. `https://google.github.io/volley/`.

[12] *Square* developers. Retrofit. `https://square.github.io/retrofit/`.

[13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley professional computing series. Addison-Wesley, 1994.

[14] A. Geron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2017.

[15] D. Ghimire. Comparative study on python web frameworks: Flask and django, 2020.

[16] J. Görtler, R. Kehlbeck, and O. Deussen. A visual exploration of gaussian processes. *Distill*, 2019.

[17] A. Javaid. Understanding dijkstra algorithm. *SSRN Electronic Journal*, 01 2013.

[18] M. Lachgar, H. Benouda, and S. Elfirdoussi. Android REST APIs: Volley vs Retrofit. In *2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, pages 1–6, 2018.

[19] N. Li and B. Becerik-Gerber. Performance-based evaluation of rfid-based indoor location sensing solutions for the built environment. *Advanced Engineering Informatics*, 25(3):535–546, 2011.

[20] Y. Lin, Y. Liu, G. Gao, X. Han, C. Lai, and M. Gu. The ifc-based path planning for 3d indoor spaces. *Advanced Engineering Informatics*, 27(2):189–205, 2013.

[21] S. Lund. AndroidImageMap. `https://github.com/catchthecows/AndroidImageMap`.

[22] L. Mainetti, L. Patrono, and I. Sergi. A survey on indoor positioning systems. In *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 111–120, 2014.

[23] R. Montoliu, E. Sansano, J. Torres-Sospedra, and O. Belmonte. Indoorloc platform: A public repository for comparing and evaluating indoor positioning systems. In *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8, 2017.

[24] D. Morrisey. Subsampling scale image view. `https://github.com/davemorrissey/subsampling-scale-image-view`.

[25] D. Rachmawati and L. Gustin. Analysis of dijkstra's algorithm and a* algorithm in shortest path problem. *Journal of Physics: Conference Series*, 1566(1):012061, 2020.

[26] C. Rasmussen and C. Williams. *Gaussian process for machine learning*. The MIT Press, 2006.

[27] G. Retscher and A. Leb. Development of a smartphone-based university library navigation and information service employing wi-fi location fingerprinting. *Sensors*, 21(2):432, 2021.

[28] S. Russell and P. Norvig. *Artificial Intelligence: a modern approach*. Pearson, third edition, 2009.

[29] E. Schulz, M. Speekenbrink, and A. Krause. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

[30] S. Subedi and J. Pyun. Practical fingerprinting localization for indoor positioning system by using beacons. *Journal of Sensors*, 2017.

[31] T. Van Haute, E. De Poorter, P. Crombez, F. Lemic, V. Handziski, N. Wirström, A. Wolisz, T. Voigt, and I. Moerman. Performance analysis of multiple indoor positioning systems in a healthcare environment. *International Journal of Health Geographics*, 15(1), Feb 2016.

[32] W. Wang, S. Klettner, G. Gartner, T. Fian, G. Hauger, A. Angelini, M. Söllner, A. Florack, M. Skok, and M. Past. Towards a user-oriented indoor navigation system in railway stations. In *LBS 2019; Adjunct Proceedings of the 15th International Conference on Location-Based Services/Gartner, Georg; Huang, Haosheng*. Wien, 2019.

[33] S. Yiu, M. Dashti, H. Claussen, and F. Perez-Cruz. Wireless rssi fingerprinting localization. *Signal Processing*, 131:235–244, 2017.

[34] F. Zhang, X. Chen, and X. Zhang. Parallel thinning and skeletonization algorithm based on cellular automaton. *Multimedia Tools and Applications*, pages 33215–33232, 2020.

[35] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, 27(3):236–239, Mar 1984.

[36] X. Zhou, Q. Xie, M. Guo, J. Zhao, and J. Wang. Accurate and efficient indoor pathfinding based on building information modeling data. *IEEE Transactions on Industrial Informatics*, 16(12):7459–7468, 2020.

Appendix A

Participants' information sheet

This study was certified according to the Informatics Research Ethics Process, RT number 2021/59600. Please take time to read the following information carefully. You should keep this page for your records.

Who are the researchers?

The study is conducted by Dimitris Christodoulou, a 5th year Informatics student, as part of his MInf thesis. The project is supervised by Dr Petros Papapanagiotou, Chancellor's Fellow in Digital Technologies

What is the purpose of the study?

The aim of the study is to evaluate an Android application that aims to help students find their way around the Main Library of the University of Edinburgh. The app has two main pieces of functionality:

- **Localisation:** Using the received signal strength from nearby access points, it estimates the user's location and displays it on a floor plan of the library.
- **Pathfinding:** Using the received signal strength from nearby access points, it estimates the user's location and displays it on a floor plan of the library.

The study will help determine whether users perceive the estimated locations as accurate. It will also establish whether users are able to use the location and paths provided by the app to successfully navigate around the library.

Why have I been asked to take part?

The target group of this study is students who use the Main Library building for any reason such as borrowing books, access to printers or using study spaces. We particularly welcome participation from students who are not regular users of the library. Participants need access to a device with the Android operating system (Version 11 or newer).

Do I have to take part?

No – participation in this study is entirely up to you. You can withdraw from the study

at any time, up until the end of March 2022 without giving a reason. After this point, personal data will be deleted, and anonymised data will be combined such that it is impossible to remove individual information from the analysis. Your rights will not be affected. If you wish to withdraw, please contact the PI. We will keep copies of your original consent, and of your withdrawal request.

What will happen if I decide to take part?

First, you will be given instructions to install and access to the Android application developed during this project. You will be given instructions to go to a particular area of the Main Library of the University of Edinburgh and use the app to find your location, request paths to certain destinations and follow the paths given by the application to reach those destinations. You will then be asked to fill in a short questionnaire about how accurate you believe the information provided by the app was and whether you were able to successfully complete the tasks using the information provided by the application.

This is expected to take approximately 15 minutes and no more than 30 minutes.

For the purposes of the study, the localisation and pathfinding requests sent from the app to the server will be stored. This includes Wi-Fi signal strength readings taken by your device. This data will only include a list of nearby access points and the corresponding signal strengths as received by your device. No identifying information about your device or you will be stored alongside it. It will only be associated with a unique number that will be provided to you by the server through the app. You will be asked to enter this number in the questionnaire so that we can associate your responses with the readings. You may also be asked to note your location at specific moments during the session on a library floor plan provided in the application, and these locations will also be stored in the server.

Data processing to estimate the location and calculate paths is conducted in the server. Aside from the data stated above no other data will be stored in the server after processing is complete.

Are there any risks associated with taking part?

There are no significant risks associated with participation.

Are there any benefits associated with taking part?

There are no direct benefits associated with participation. Indirect benefits include the opportunity of reviewing an early prototype of the app. If further development is pursued by the University, it could be a useful tool for students using the library.

What will happen to the results of this study?

The results of this study may be summarised in published articles, reports and presentations. Quotes or key findings will be anonymized: We will remove any information that could, in our assessment, allow anyone to identify you. With your consent, information can also be used for future research. Your data may be archived for a maximum of 2 years. All potentially identifiable data will be deleted within this timeframe if it has not already been deleted as part of anonymization.

Data protection and confidentiality.

Your data will be processed in accordance with Data Protection Law. All information collected about you will be kept strictly confidential. Your data will be referred to by a unique participant number rather than by name. Your data will only be viewed by the researcher/research team (Dr Petros Papapanagiotou, Dimitris Christodoulou).

All electronic data will be stored on a password-protected encrypted computer, on the School of Informatics' secure file servers, or on the University's secure encrypted cloud storage services (DataShare, ownCloud, or Sharepoint) and all paper records will be stored in a locked filing cabinet in the PI's office. Your consent information will be kept separately from your responses in order to minimise risk.

What are my data protection rights?

The University of Edinburgh is a Data Controller for the information you provide. You have the right to access information held about you. Your right of access can be exercised in accordance Data Protection Law. You also have other rights including rights of correction, erasure and objection. For more details, including the right to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk. Questions, comments and requests about your personal data can also be sent to the University Data Protection Officer at dpo@ed.ac.uk.

Who can I contact?

If you have any further questions about the study, please contact the lead researcher, Dimitris Christodoulou at [email]. If you wish to make a complaint about the study, please contact inf-ethics@inf.ed.ac.uk. When you contact us, please provide the study title and detail the nature of your complaint.

Updated information. If the research project changes in any way, an updated Participant Information Sheet will be made available on http://web.inf.ed.ac.uk/infweb/research/study-updates.

Appendix B

Participants' consent form

Consent was collected through an electronic from immediately before the questionnaire was provided to the participants. In order to proceed to the questionnaire, they were asked to click a button indicating their consent to the following statements:

- I have read and understood the above information.
- I understand that my participation is voluntary, and I can withdraw at any time.
- I consent to my anonymised data being used in academic publications and presentations.
- I allow my data to be used in future ethically approved research.

Appendix C

User study questionnaire

Installation

Before you install the app you should enable the option to install apps from unknown sources. You can do that by going to Apps -¿ Install apps -¿ Install unknown apps and then enable the option for the browser you will use to download the app. Remember to disable this again once you are finished with this study. The way to find this in the Settings may vary depending on your device's manufacturer.

You can install the app using the APK (Android package) that is available on [link to APK]. Once you download the file, open it on your Android device and you will be prompted to install it.

Have you been able to successfully install and launch the app?

- Yes
- No

What issue did you encounter that prevented you from installing and/or launching the app? (Optional)

Open-ended question

Go to the settings of the app by tapping on the gear icon at the bottom of the screen and generate a unique participant number. Enter your number here:

Four-digit number

Survey Plan

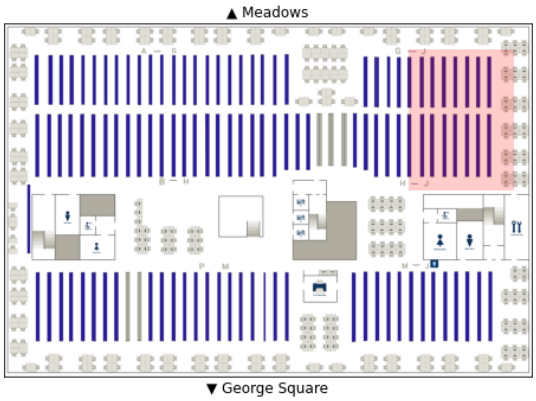
As part of this survey you will be asked to complete the following tasks:

1. Go to a pre-determined location on the 2nd floor of the library and request from the app to detect your location
2. Request a path to the location of a specific book and follow it to get there

3. Request from the app to detect your location again
4. Request a path to the location of a different book and follow it to get there
5. Request from the app to detect your location again

Before you start steps 3 and 5, if you were unable to get to the relevant books or you do not wish to continue to the next tasks, you will have the option to end the survey there.

Detecting your current location

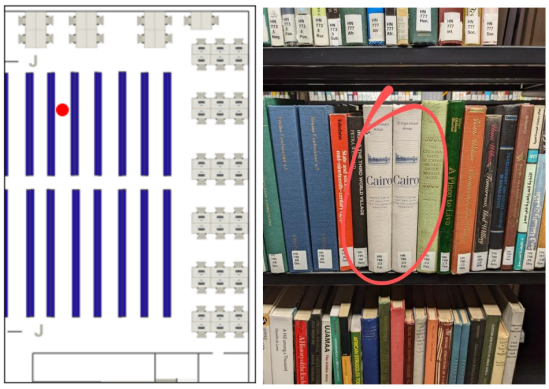


Please go to the part of the second floor of the Main Library highlighted in the figure above.

What is the error that you consider reasonable for location detection?

- Less than 1 m
- Less than 3 m
- Less than 5 m
- Less than 10 m

Please go to the location in the second floor of the Main Library indicated in the floor plan on the left. There, you should see the book 'Cairo Cosmopolitan' by Singerman and Amar. (HN786.C3 Cai.), which is highlighted in the image on the right. Tap on the 'Localisation' button (The circular button in the bottom-right corner) to initiate a Wi-Fi scan and request from the app to detect your current location.



Does the estimated location look right to you?

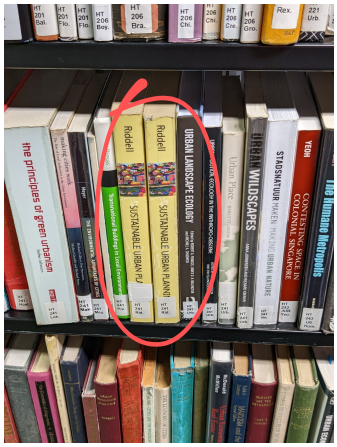
- Yes
- No

How big do you think is the difference between the estimated location and your actual location?

- Less than 1 m
- 1 m - 3 m
- 3 m - 5 m
- 5 m - 10 m
- More than 10 m

Pathfinding

Now tap the 'pathfinding' button (bottom left corner) and select the book 'Sustainable Urban Planning' by Riddell R. (HT241 Rid.) as your destination.



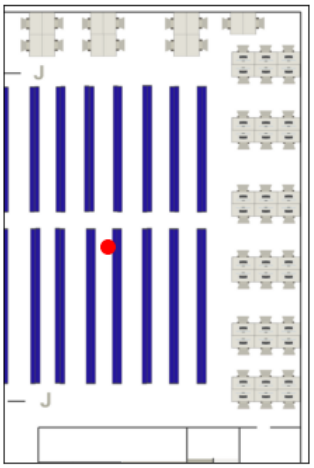
Were you able to successfully follow the provided path to reach the selected destination? There you should see the book 'Sustainable Urban Planning' by Riddell R. (HT241 Rid.) which is highlighted in the given image

- Yes
- No

Are there any issues with the path that was provided? (Optional)

- No Issues
- Too complicated
- Too circuitous (not direct)
- Has obstacles in the way
- Not easy to follow instructions
- Incorrect starting point

Detecting your current location (Part 2)



Can you find the book 'Sustainable Urban Planning' by Riddell R. (HT241 Rid.), irrespective of whether you were able to follow the provided path before? Its location is shown in the figure.

If you want to end the survey at this step, you can select the "I want to stop here" option.

- Yes
- No
- I want to stop here

While you are at the location of the book 'Sustainable Urban Planning', tap on the 'Localisation' button to initiate a Wi-Fi scan and request from the app to detect your current location.

Does the estimated location look right to you?

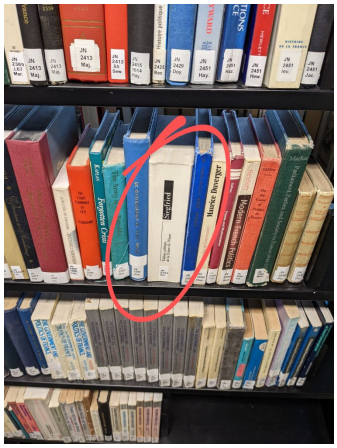
- Yes
- No

How big do you think is the difference between the estimated location and your actual location?

- Less than 1 m
- 1 m - 3 m
- 3 m - 5 m
- 5 m - 10 m
- More than 10 m

Pathfinding (Part 2)

Now tap the 'pathfinding' button (bottom left corner) and select the book 'Tableau politique de la France de l'Ouest' by Seifried A. (JN2593 Sie.) as your destination.



Were you able to successfully follow the provided path to reach the selected destination? There you should see the book 'Tableau politique de la France de l'Ouest' by Seifried A. (JN2593 Sie.) which is highlighted in the provided image.

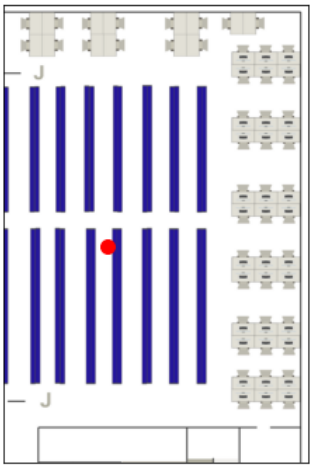
- Yes
- No

Are there any issues with the path that was provided? (Optional)

- No Issues

- Too complicated
- Too circuitous (not direct)
- Has obstacles in the way
- Not easy to follow instructions
- Incorrect starting point

Detecting your current location (Part 3)



Can you find the book 'Tableau politique de la France de l'Ouest' by Seifried A. (JN2593 Sie.), irrespective of whether you were able to follow the provided path before? Its location is shown in the figure

If you want to end the survey at this step, you can select the "I want to stop here" option.

- Yes
- No
- I want to stop here

While you are at the location of the book 'Tableau politique de la France de l'Ouest', tap on the 'Localisation' button to initiate a Wi-Fi scan and request from the app to detect your current location. Does the estimated location look right to you?

Does the estimated location look right to you?

- Yes
- No

How big do you think is the difference between the estimated location and your actual location?

- Less than 1 m
- 1 m - 3 m
- 3 m - 5 m
- 5 m - 10 m
- More than 10 m

Thank you for your participation

Please uninstall the app and remember to disallow installation of apps from unknown sources. You can do that by going to Apps -¿ Install apps -¿ Install unknown apps and then disable the option for the browser you used to download the app.

Please write any other feedback about the app here (Optional)

Open-ended question

[Submit button]