**Background and Motivation**

* Indoor localisation helps us locate a system within an enclosed space, it’s applications ranging from helping firefighters conduct search and rescue operations to allow indoor robot assistants map their surroundings
* Indoor vs outdoor localisation
  + Outdoor localisation technologies are highly developed, consider Google maps
  + However, with indoor localisation there is no such industry wide standard and there are non-line of sight issues with outdoor based localisation which makes it non-scaleable to indoor localisation
* So currently, industry and academia are working on solutions to implemented indoor based localisation
* Industry and academic have proposed solutions ranging from custom hardware setups (relays, bluetooth beacons, magnetic resonators) to so-called infrastructure free setups using WiFi technologies
* WiFi based technologies are considered an ideal candidate for performing indoor localisation with, because there are very little modifications that need to be made to architecture, and because this technology is ubiquitous.
* A common technique used in WiFi localisation is the use of fingerprinting, where the Received Signal Strength Indicator (RSSI) from an AP is measured at a specific point in the space, and the model is trained on WiFi Signal strengths in dB versus the location it was measured it.
* Physics based simulations can provide WiFi strengths that are very close to reality, however these simulations are computationally expensive (take a long time and a lot of compute resources). Therefore less accurate data can help us strike a balance between accuracy as well compute resources by training a machine learning model on this data.
* This calls for the use of Machine learning to find patterns in the data and associate them with an access point - two teams before our own have tried to create ML models to create data models with, however, their models have either inaccurate (a large error in the prediction).
* So our motivation was coming up with a machine learning model that could predict the x-y location of a point in the infrastructure given it’s RSSI Wifi Signal strength.

**Project Goal**

Use machine learning and low fidelity RSSI data to predict the location of a person/object within an indoor environment.

**Project Requirements**

* Functional Requirement
  + Location in x-y coordinates of a static target given a WiFi (RSSI) signal: this is the fundamentals of WiFi fingerprinting.
  + Create RSSI WiFi Fingerprint map using raytracer simulations
  + Map of 8th floor Bahen
* Constraints
  + Raspberry pi access points configured separately from existing APs in Bahen
    - This was to ensure that if we were going to try and collect data we would be able to isolate the WiFi signals we were measuring from the access points in bahen and also control the power output from them
  + Another constraint we imposed on ourself was that we would only be looking at the 8th floor of Bahen
* Objectives
  + Augment training data
  + Prediction < 4m
    - With the Wifi technology that we were using, we wanted to be able to measure 1/frequency…

**Methodology**

We divided the project into three parts, data collection, data cleanup and localization.

For data collection we simulated data using the RayTracer. The ray tracer takes three inputs, a map of the 8th floor of Bahen, the receiver x-y coordinates over the entire map and configurations for the RayTracer which includes the transmitter coordinates. Of the three inputs, we were provided with a .stl file containing the Bahen 8th floor map and basic configurations for the RayTracer. We selected the receiver points by creating a python script to select uniform points throughout the map. We varied the distance between points to get one dataset for training points and one for testing. After getting the receiver points we selected 5 transmitter points also known as access points. We selected the access points based on geometry, 3 at each corner and 2 at the middle of the corridors. This was done to ensure all the areas of the map are equally covered and all are equidistant.

Lastly, we had to find the number of reflections to consider for this map and to do so we implemented a script to perform convergence analysis. We had to run the ray tracer 5 times with a set of data containing the same receiver and transmitting points but different number of reflections. The convergence analysis took the RSSI values to find the points of convergence using which gave its corresponding number of reflections which came out to be 2 in our case.

— 3D model —

After we had all the inputs, we ran the ray tracer 10 times to get the RSSI values for each access point on both training and testing receiver points. Each time we run the raytracer we get an excel file containing data for that configuration.

We performed data cleanup on the excel files by combining them and cleaning up to make sure we get 5 RSSI values for each receiver coordinates. We ended up getting a ratio of 82 : 18 for training vs testing data.

Once data cleanup was done, we moved on to the localization phase where we implemented three models to find the best model. These models were trained on the RSSI values and the coordinates of the training data set and they predicted the coordinates using the RSSI values of the testing data set.

**Design Overview**

Supervised vs Unsupervised Models

A supervised model refers to a model that was trained/validated against ground truth labels, while an unsupervised model has no labels and learns based on its own clusterings.

Explain the three models

* KNN:
  + The K-Nearest Neighbors (KNN) is an unsupervised learning model that takes in the row vector of RSSI values from each access point from our dataset, and predicts k of the nearest x-y points from it. The k value is one of the hyperparameters we tuned, and we found it was optimal at 10 neighbors. Once we have the 10 possible neighbors, we calculated the weighted average, using a weight of 0.2 as the smoothing factor (this was also a tuned hyperparameter). Overall, this model achieved an error of 4.08m, which is slightly above the 4m threshold we were given.
* ANN:
  + The Artificial Neural Network (ANN) is another supervised learning model. Our version was a scikit learn model with 2 hidden layers, a learning rate of 10^-5, and the data randomly split to reduce overfitting. This model was able to achieve an error of 5.17m.
* Random Forest:
  + Explain that a decision tree is itself a machine learning algorithm that takes a dataset and splits the dataset based on different features. We enter input data into the model which splits the data [explain how it does this] the regression model splits the data on different features that are generated by the model. This makes leaf nodes and the model does this recursively until it hits upon a max depth
  + When an input sample is processed through the model it goes along this tree until a decision node is reached upon which the model delivers its output
  + The problem with a decision tree is that it is not good at generalizing and often overfits the data. In order to fix this we use a random forest
  + Use a group of decision trees and gather a consensus by averaging the output of each decision tree
  + Explain the hyperparameters and justify what we did

**Design Challenges and Final Results**

* Two of our major challenges came from the dataset itself. Throughout the project we made several steps toward engineering the data that would enhance the accuracy of the models. For example we used convergence analysis to find the number of reflections in the raytracer, furthermore we found that the data could be enhanced if we used 5 rather than three transmission areas. While we engineered the raytracer data we did some preliminary collection of live data. When collected we found that the data points collected mismatched the data projected by the raytracer.
* Our third major challenge was narrowing down which model to use, since this was a regression problem our focus was on using models best suited toward regression we decided to use models from both the unsupervised and supervised paradigms in order to compare and contrast their efficacy
* In the end despite these challenges we found that our unsupervised model the cluster KNN produced results close to last years results. Of the two supervised models the Random Forest Regressor performed the best with a MAE of 2.77 while the ANN performed close to the KNN
* In the future, models can use real world data instead of simulated data to get a better sense of how to actually perform given a real object in Bahen. This may reduce the error as the RSSI values would be more precise.
* Self organizing maps??