Data Augmentation Methods for Indoor Localization using RSSI Fingerprinting

Christian Thomas Hurley
The Technical Faculty of IT and Design
Aalborg University, Denmark
churle17@student.aau.dk

Christoffer Hansen
The Technical Faculty of IT and Design
Aalborg University, Denmark
ch17@student.aau.dk

Fredrik De Frène
The Technical Faculty of IT and Design
Aalborg University, Denmark
fdefre16@student.aau.dk

Mads-Bo Bomholt Lassen
The Technical Faculty of IT and Design
Aalborg University, Denmark
mlasss17@student.aau.dk

Mathias Andresen
The Technical Faculty of IT and Design
Aalborg University, Denmark
mandr17@student.aau.dk

Simon Park Kærgaard
The Technical Faculty of IT and Design
Aalborg University, Denmark
skarga17@student.aau.dk

***Abstract*—In this paper, we explore indoor localization data. We analyze the data in-depth, and we ascertain an imbalance between ground truths and labeled sensory data. We propose three data augmentation methods specific to WiFi-based RSSI fingerprinting that attempt to remedy the imbalance. Through our experiments, using established machine learning algorithms, we show that in most cases, the augmentation methods decrease the mean position error. Our experiments also document to what degree the machine learning algorithms used in this study are affected by the data's shortcomings. In the case of augmentation, our methods decrease the mean position error by -6.92 - 59%, with a mean of 25.97%. We also propose a postprocessing technique that utilizes metainformation about the data to improve predictions. We apply this technique to our best-performing models' predictions, as well as our worst-performing model's predictions for reference. The postprocessing technique decreases the mean position error by 0.24 - 18.23%, with a mean of 6.81%.**

***Index Terms*—mobility, feature engineering, machine learning, location based services, snap, postprocessing**

I. Introduction

Outdoor localization is aided dramatically by GPS, which is included in virtually all modern smartphones. However, since GPS accuracy is dependent on line-of-sight, it is greatly reduced when users are located indoors. Hence, a positioning system for indoor localization has become a research area with significant interest. The ability to determine the precise location of a device indoors can aid users in navigating large structures such as malls, airports, or hotels. Thus it can also succor in areas such as disaster management, security and tracking, and logistics.

Many techniques for indoor positioning utilize a fingerprint of the location by collecting sensor data at specific positions, which acts as a unique signature for that position. The increased availability of WiFi access points in indoor locations, such as shopping malls, has made WiFi a reliable source of data to build a fingerprint. Studies also indicate that additional data from sensors in modern smartphones can further increase the accuracy of predictions [1].

In most cases, such fingerprints are constructed by manually surveying the indoor area and establishing ground truths, denoting physical locations within the building. WiFi based fingerprinting works by recording the state of signals, such as the RSSI of observed WiFi signals, at the given locations. This means that in order to increase the number of data points, one has to spend more time manually surveying the area, ensuring high coverage, which can be cumbersome. The data collection is not a trivial task either [2], as RSSI signals are affected by challenges in the online phase, where all online RSSI measurements are compared to the RSSI's in the fingerprint. Depending on the fingerprint's size, this quickly becomes computationally expensive.

This paper proposes data augmentation methods that utilize the established *ground truths* within a given trace to generate additional labels for recorded data. Furthermore, a sophisticated step calculation that uses the sensory data recorded in-between *ground truths* is used to calculate the surveyor's trace to create a more accurate fingerprint of the buildings. This lets us create additional labels and more accurately connect WiFi observations to a location. The augmented data is tested extensively using several established ML algorithms.

Furthermore, we propose a postprocessing technique that adjusts predictions by using *hit detection* of polygons, which indicates shops and hallways lying within the building floor plan.

We refer to Table I for the reader to look up abbreviations used throughout this paper.

The following is a summary of our findings:

- section II: We research the area of indoor localization to get acquainted with the literature and recent advancements in the field.
- section III: An introduction to the dataset, which is fundamental for our fingerprinting and postprocessing techniques.
- section IV: Our dataset analysis shows that a significant portion of the observed RSSIs is recorded far in-between ground truths, making them less reliable for fingerprint-

TABLE I: List of abbreviations and terms.

Abbreviation	Expansion
BSSID	Basic Service Set Identifier
EDA	Exploratory Data Analysis
GPS	Global Positioning System
GSM	Global System for Mobile Communications
IMU	Inertial Measurement Unit
kNN	k-Nearest Neighbors
LGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory
ML	Machine Learning
MPE	Mean Position Error
NN	Neural Network
RFID	Radio Frequency Identification
RSSI	Received Signal Strength Indicator
SSID	Service Set Identifier
SVM	Support Vector Machine

ing. The higher offset in time and distance a WiFi reading is from a ground truth, the more unreliable it becomes for fingerprinting.

- section V: We introduce our course of action when approximating the surveyor's location in-between the original ground truths. Moreover, we explain the process of comparing predictions to the wall boundaries within the floor maps of buildings in order to *snap* them, i.e., move their location away from non-walkable areas.
- section VI: We explain the structure of our experiments and how we use these methods in practice, e.g., how we augment the data, tune, train, and predict with our models, and apply our postprocessing technique to our predicted traces.
- section VII: We present the results of our pre and postprocessing techniques and analyze how they affect the predictions.
- section VIII: We discuss our results, methodology, and experiments, while explaining strengths and weaknesses to our approaches.
- section IX: We conclude on our findings.

II. Related Work

Indoor localization has been a prevalent research area in the past two decades, both within academia and industry. Early approaches to indoor localization utilize technology such as ultrasound [3], infrared [4], and RFID [5]. Many of these approaches work well within their respective area but require additional sensors or changes to infrastructure in order to work, which can be costly. Therefore, in recent years, the focus has shifted to find the most cost-efficient indoor localization method, using technologies such as WiFi [6] [7] [8], and Bluetooth [9], which are already primarily integrated into modern infrastructure and standard mobile devices. The papers presented in this section will be separated into two categories: geometric distance measurement and fingerprinting.

A. Geometric Distance Measurement

For non-fingerprint-based localization, we mostly have to look elsewhere than WiFi, as most WiFi-based localization systems use fingerprinting. ToneTrack [10] achieves a median accuracy of 90cm under good conditions by analyzing the time-of-arrival of signals on varying frequencies. However, it does require additional hardware, albeit minimal.

EZ [11] uses RSSI measurements without fingerprinting, where a localization server exploits real-time ground-truth locations gathered through GPS and the physics of wireless propagation to locate a device. It achieves only slightly worse results than at-the-time common localization schemes that utilize fingerprinting.

B. Fingerprinting

Xia et al. [6] analyze fingerprint indoor positioning on a broader level, discussing and explaining the different technologies involved in the process of fingerprinting using WiFi. The paper addresses the most recent technology used in fingerprint indoor positioning and explains the theory behind some of the crucial components in the aforementioned process.

Martin et al. [8] presents a mobile application that integrates both the offline and online phases of RSSI fingerprinting, making it possible to perform indoor localization on a single device. It also presents a study of at-the-time available resources in a standard smartphone, namely WiFi, GSM, and accelerometer, to determine the possibility of a multimodal approach for localization.

Based on sensors in at-the-time standard smartphones, Mascharka et al. [12] perform a large-scale examination of machine learning algorithms for indoor localization, achieving better accuracy than algorithms considered in Martin et al. [8]. They also propose a hybrid approach, using multiple classifiers, where each one is responsible for a different area. This approach has similar accuracy and significantly reduced speed, making it more suitable for a real-world setting.

A generative approach to WiFi-based indoor fingerprint localization is proposed in Belmonte-Fernández et al [7]. The method presented takes advantage of the temporal autocorrelation in WiFi signals by modeling the RSSIs using a hidden Markov model. The proposed method performed slightly better than common machine learning methods, namely kNN, Naive Bayes, and Multi-Layered Perceptron, which seems to correlate to the sample size used in the experiments. However, regardless of sample size, it was outcompeted by Random Forest in the majority of the experiments.

Lastly, the following papers explore a more specialized area pertaining to the scope of this paper. Sinha et al. [13] propose two data augmentation schemes for WiFi-based fingerprint indoor positioning. The first augmentation scheme introduces slight diversity to the data set by subtracting a single RSSI value in a reference point, which is a collection of WiFi access points, by a constant, repeated for the total number of RSSI values in all reference points. The second augmentation scheme replaces RSSI values using the mean value of RSSIs and uniform random numbers in the range of the minimum and the mean of the original RSSI values, adding a user-defined number of samples to the data set.

Another paper by Sinha et al. [14] explore an improved method of performing RSSI-based data augmentation for use in fingerprint indoor localization. The augmentation technique proposed in the paper revolves around increasing the size of the data samples by creating N new samples consisting of values similar to those that exist in the original data. The proposed technique introduces variation in RSSI signals, reflecting how RSSI signals act in a real-world scenario and therefore helps create more robust models. It also increases test accuracy levels in their experiments compared to other data augmentation technique for indoor positioning [13] [15].

III. Dataset

The dataset [16] used for this study is created and publicized by XYZ10 in collaboration with Microsoft Research for educational use and academic studies. We present the explicit features of the dataset to get acquainted with its content and structure.

A. Data Collection

The data is collected by surveyors walking through 204 different buildings and malls in Shanghai, China. The surveyors collect data through an Android phone, which is attached to their chest to stabilize it and reduce data pollution throughout the collecting. The collection is managed by an application, which records sensory data through the Android API periodically throughout the surveyor's path. Traces are constituted by a collection of ground truth waypoints that the surveyors manually indicate in the application during their traversal of the buildings.

In addition to sensory data, the application also records a snapshot of reachable WiFi access points and their RSSIs during various time intervals. This also includes the WiFi access points' SSID and BSSID, which are used to identify them, and their associated hotspot group, respectively. The sensory data, which includes the calibrated and uncalibrated versions of the IMU, is collected every 20 *ms*. A full overview of the collected data's structure is found in Appendix X-A.

Data quality analysis is conducted during the data collection phase by XYZ10 in order to ensure extensive path coverage of the different building floors.

B. Training Data

The entirety of the training data consists of 204 buildings that are all organized by floors. There are a total of 981 floors, ranging from the 2nd floor below ground to the 9th floor above ground. Moreover, each floor has several traces collected by the surveyors. A trace is limited to a single floor, which means that surveyors have not collected data while moving between floors. In total, there are 26 925 trace files. Additionally, the dataset includes a collection of metadata files corresponding to the building and floors of the training set. The metadata for each floor contains geographic information such as an image representing the floor map and geometrical shapes and coordinates of each shop located on the floor.

We choose to focus on the 24 of the 204 buildings that are used for the Kaggle competition. These 24 buildings are determined from the `submission.csv` template submission file, included with the dataset.

C. Data Correction

The sensory data collected through the Android API utilizes internal timestamps. These are not always coherent with the waypoints' timestamps, annotated by the surveyors. In order to make the timestamps consistent, we perform preprocessing on the dataset. We adjust the existing waypoint timestamps by changing them to the timestamp of the preceding Android API timestamp.

Due to dirty data concerning certain iBeacon data entries, where missing newlines causes reading errors, we create a script that fixes these errors by inserting missing newlines and necessary padding data. In one of the metadata files, the height information specified in the `floor_info.json` is incorrect. To correct this, we manually adjust the height in order for it to match.

IV. Exploratory Data Analysis

In this section, we present the results of our EDA, in which we inspect the underlying features of the dataset. This includes two areas of interest; frequency analysis of the sensory and waypoint readings and a timestamp analysis of the readings. These results help us understand the input data and shed light on potential shortcomings or anomalies that need addressing.

We conduct these analyses on all 204 buildings within the dataset. Even so, we only display the results of a subset of buildings, which corresponds to the 24 focus buildings. Furthermore, we refrain from looking at the iBeacon readings due to Apple's developer guide [17] stating: *"Due to the issues around signal strength and the variabilities in deployment environments, iBeacon technology is not intended to be used for specific location identification."*

A. Frequency Analysis

Figure 1 shows the results of the initial analysis in which we count the frequency of sensor readings in the paths of the training data.

As the WiFi per waypoint and sensor per waypoint ratios imply, the number of recorded waypoints is substantially lower than the frequency of WiFi and sensor readings. By way of example, on average, we observe 3.4 WiFi readings per recorded waypoint in the focused 24 buildings.

During our analysis, we encountered 278 traces from various buildings with no recorded WiFi readings. We initially suspected this to be caused by a lack of WiFi coverage at the locations of the path; however, we refute this suspicion by closer analysis. By way of example, Figure 2 shows that several WiFi readings have been recorded in other traces in the same area as the trace with none. The same applies to the other 278 traces with no WiFi readings.

To summarize, the skewed ratio between waypoints and sensor/WiFi readings may be a potential bottleneck within

building	sensors	wifi	waypoint	wifi/ waypoint ratio	sensor/ waypoint ratio
5a0546857ecc773753327266	1019140	9296	2491	3.732	409.129
5c3c44b80379370013e0fd2b	1046675	9737	1698	5.734	616.416
5d27075f03f801723c2e360f	2947072	23666	7290	3.246	404.262
5d27096c03f801723c31e5e0	904216	9100	1925	4.727	469.723
5d27097f03f801723c320d97	1041924	10507	3795	2.769	274.552
5d27099f03f801723c32511d	446357	4251	897	4.739	497.611
5d2709a003f801723c3251bf	393936	3940	1008	3.909	390.810
5d2709b303f801723c327472	1572793	15358	3119	4.924	504.262
5d2709bb03f801723c32852c	1711777	17203	3613	4.761	473.783
5d2709c303f801723c3299ee	1053043	10083	4110	2.453	256.215
5d2709d403f801723c32bd39	1044923	10027	2978	3.367	350.881
5d2709e003f801723c32d896	1106148	11042	2721	4.058	406.523
5da138274db8ce0c98bbd3d2	261201	2662	809	3.290	322.869
5da1382d4db8ce0c98bbe92e	1050594	8999	2636	3.414	398.556
5da138314db8ce0c98bbf3a0	917758	9012	1547	5.825	593.250
5da138364db8ce0c98bc00f1	271089	2767	518	5.342	523.338
5da1383b4db8ce0c98bc11ab	1305198	13196	3067	4.303	425.562
5da138754db8ce0c98bca82f	704262	7188	1430	5.027	492.491
5da138764db8ce0c98bcaa46	927769	9420	3489	2.700	265.913
5da1389e4db8ce0c98bd0547	647518	6312	2775	2.275	233.340
5da138b74db8ce0c98bd4774	1790646	17382	6111	2.844	293.020
5da958dd46f8266d0737457b	1601428	15148	3831	3.954	418.018
5dbc1d84c1eb61796cf7c010	1850916	16174	7169	2.256	258.183
5dc8cea7659e181adb076a3f	1749139	15655	6258	2.502	279.504
Total	27365522	258125	75285	3.429	363.492

Fig. 1: Sensor frequencies of the 24 buildings included in the test set. For each building, the *sensors* column describes the number of the recorded IMU (accelerometer, gyroscope) and geomagnetic field (magnetometer) readings. We present these in one column as they are all read with the same frequency per building. The *wifi* column constitutes the number of segments of consecutive WiFi observations that share the same timestamp.

Fig. 2: Side-by-side comparison of a trace with no recorded WiFi readings (left) and a heat-map of the WiFi counts at given positions in other traces(right). The number indicator does not apply to the left image.

the data that needs addressing. As a result of fingerprinting research in section II, we have recognized WiFi readings to be a crucial aspect in modern fingerprinting techniques. For this reason, we shift our attention to the WiFi/waypoint ratio, since this ratio indicates that we can generate additional waypoints from the excess WiFi readings. We apply methods to determine how big of an impact this possible shortcoming has on future prediction models in later sections.

In regards to the traces with no recorded WiFi readings, they pose a problem in a fingerprinting approach relying strictly on WiFi readings; thus, we omit them from the data.

B. Time Difference Analysis

Figure 3 presents the mean time differences between the recorded waypoints and WiFi readings, and vice versa, of the 24 buildings.

building	wifi mean	waypoint/wifi mean	wifi/waypoint mean
5a0546857ecc773753327266	2142.851	983.817	3384.420
5c3c44b80379370013e0fd2b	2109.466	984.710	5595.828
5d27075f03f801723c2e360f	2192.081	1700.459	2882.908
5d27096c03f801723c31e5e0	1944.347	826.425	3768.023
5d27097f03f801723c320d97	1940.376	678.072	2052.717
5d27099f03f801723c32511d	2051.513	965.377	4385.832
5d2709a003f801723c3251bf	1943.306	990.050	3403.190
5d2709b303f801723c327472	2016.763	1067.710	4656.684
5d2709bb03f801723c32852c	1975.050	635.604	3236.827
5d2709c303f801723c3299ee	2038.051	833.863	1890.022
5d2709d403f801723c32bd39	2015.098	939.653	2810.535
5d2709e003f801723c32d896	1971.528	822.072	2994.727
5da138274db8ce0c98bbd3d2	1887.359	929.095	2711.502
5da1382d4db8ce0c98bbe92e	2293.656	935.375	2933.731
5da138314db8ce0c98bbf3a0	2015.737	1034.691	5858.124
5da138364db8ce0c98bc00f1	1917.656	775.253	3686.351
5da1383b4db8ce0c98bc11ab	1959.554	650.954	2843.639
5da138754db8ce0c98bca82f	1963.298	824.340	3496.907
5da138764db8ce0c98bcaa46	1931.032	760.831	1987.329
5da1389e4db8ce0c98bd0547	1937.519	1015.152	2029.177
5da138b74db8ce0c98bd4774	2028.033	747.632	2117.792
5da958dd46f8266d0737457b	2096.473	1040.009	2995.559
5dbc1d84c1eb61796cf7c010	2206.793	983.824	1798.356
5dc8cea7659e181adb076a3f	2127.578	1232.554	2077.242
Total mean	2029.380	931.563	3149.892

Fig. 3: Mean time differences per building in milliseconds. The *wifi mean* column encapsulates the mean time between each recorded WiFi block's first entry. *Waypoint/wifi mean* contains the mean time between waypoints and the closest entry of a WiFi reading, and *wifi/waypoint mean* contains the mean time between each WiFi reading and its closest waypoint.

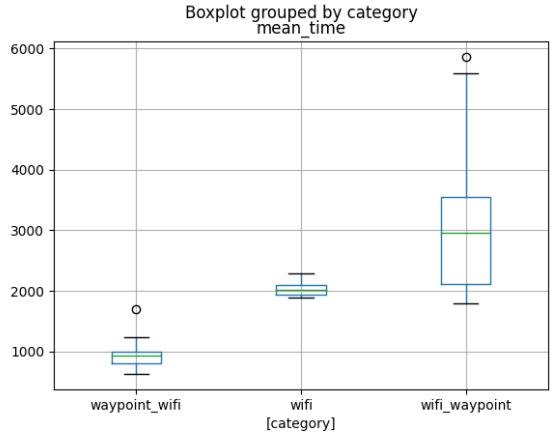


Fig. 4: The distribution of the mean time differences between each recorded WiFi, waypoint to WiFi, and WiFi to waypoint. In the two latter groups, we observe two cases of outliers.

Figure 4 presents the distribution of the mean times within each of the aforementioned categories. We observe outliers within the two waypoint categories but we refrain from altering the outlying data points, as they still serve as valid readings for our fingerprinting methodology. Furthermore, we observe a skewed distribution within the WiFi to waypoint category. On closer inspection, we discover that this is caused by paths that have a long time interval between the recorded waypoints, which results in longer times between each recorded WiFi reading and its closest waypoint. Figure 5 shows an example illustrating this case. We consider this a problem that needs addressing in order to use the data to its full potential.

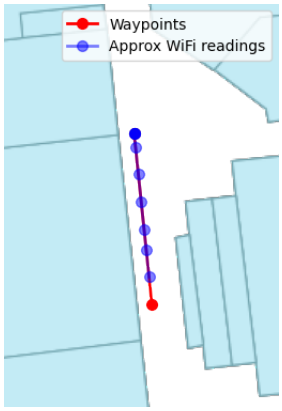


Fig. 5: Approximated WiFi readings (blue) within the path of the two observed waypoints (red). We observe several WiFi readings in between the two observed waypoints of the visualized path. The approximation of the WiFi readings is done by the method explained in subsection V-C.

With the mean times, we can estimate how far the surveyor has been able to walk in between the recorded waypoints and WiFi readings. From the study done on *"Normal walking speed"* by Richard W. Bohannon and A. Williams Andrews in [18], we use their results to determine how far an average man or woman can walk in the mean time between each WiFi reading. They claim: *"The grand mean gait speed was relatively consistent for the decades 20 to 29 years to 60 to 69 years for men (133.9 to 143.3 cm/second) and women (124.1 to 139.0 cm/second)."* Thus, if the timestamp of a WiFi reading and the timestamp of its closest waypoint are too far apart, we must consider it inadequate to be used as a feature for model training. The distance will likely affect the accuracy of our predictions, where a greater distance between readings means a lower accuracy.

V. Methodology

This section serves as a reference point to the following sections of the paper, in which we present the methodology behind the techniques we use to improve our localization accuracy. This includes preprocessing techniques to generate labels for our fingerprints and postprocessing techniques to improve our predictions.

A. Data Representation

Our fingerprint's main feature is extracted from the WiFi readings. As mentioned in subsection III-A, reachable WiFi access points are periodically pinged by the application used to survey, and their RSSIs are measured and recorded. We aggregate all WiFi observations with the same timestamp into a single WiFi reading. This representation is shown in Table II, where the RSSI is annotated for each BSSID within the building.

TABLE II: Representation of 4 WiFi readings.

Reading	$bssid_1$	$bssid_2$	$bssid_3$	...	$bssid_{n-1}$	$bssid_n$
1	-50	-45	-999	...	-999	-34
2	-60	-50	-999	...	-45	-32
3	-88	-30	-999	...	-999	-999
4	-999	-999	-45	...	-89	-90

The BSSID of all WiFi units within a given building is first gathered and then arranged along with the column headers. WiFi readings do not contain RSSI for all BSSIDs, as some WiFi units are unreachable in certain areas when pinged. When generating our datasets, we express this by using the RSSI value -999 for BSSIDs where this is the case, indicating that the unit is out of reach. The normal RSSI values range from 0 to -100. Alternative values were experimented with, but using -999 produced the best results.

B. Mean Position Error

The main metric used for evaluating our predictions is MPE, presented by the Microsoft Research competition team in [19], is defined as:

$$\frac{1}{N}\sum_{i=1}^{N}\left(\sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2} + p \cdot |\hat{f}_i - f_i|\right) \quad (1)$$

where:

- N is the number of rows in the test set
- $\hat{x}_i, \hat{y}_i$ are the predicted locations for a given test row
- x_i, y_i are the ground truth locations for a given test row

- p is the floor penalty, set at 15
- $\hat{f}_i, f_i$ are the predicted and ground truth integer floor level for a given test row

This metric is also used in the competition to evaluate submissions. Since we are working with an error measurement, the lower the score, the more accurate the predictions are.

C. WiFi Reading Interpolation

Intuitively, we want to utilize all the available WiFi readings, as they contribute to the unique RSSI image, i.e., fingerprint, of the different areas in the building. If possible, it would create additional data points for the models to learn and create robustness to a variety of inputs. However, as discussed in subsection IV-B, if a WiFi reading is too far away from a ground truth waypoint, it opposes its viability.

Naively, we can interpolate the surveyor's location by first calculating the linear location change between two waypoints and the time difference. By taking two neighboring waypoints, wp and wp_{next}, in chronological order, we first find the difference in coordinates and time.

$$c_x = (wp_x - wp_{next-x}) \cdot -1 \tag{2}$$
$$c_y = (wp_y - wp_{next-y}) \cdot -1 \tag{3}$$
$$c_t = |wp_t - wp_{next-t}| \tag{4}$$

c_x of Equation 2 and c_y of Equation 3 each refer to the change of x and y coordinates for the current waypoint to reach the next. Equation 4 is the time c_t difference between the two waypoints. This gives us the distance to travel in a 2d-plane to reach the next waypoint.

To interpolate the location of a WiFi reading, we first observe the time of the WiFi reading and find the time difference d between the previous waypoint and the WiFi reading. From this, we get the time differential ratio r, which is used to interpolate the location of the surveyor when the next reading is recorded.

$$d = |wp_t - wifi_t| \tag{5}$$
$$r = \frac{d}{c_t} \tag{6}$$
$$x = wp_x + c_x \cdot r \tag{7}$$
$$y = wp_y + c_y \cdot r \tag{8}$$

We multiply the change of c_x and c_y with the ratio r, in order to get the interpolated x and y coordinates of the WiFi reading.

Consider this alternative notation instead of Equation 2-8 (Shows only for x):

$$x = wp_x + (wp_x - wp_{next-x}) \cdot \frac{|wp_t - wifi_t|}{|wp_t - wp_{next-t}|} \cdot -1 \tag{9}$$

Intuitively, this approach can cause issues due to the behavior of the surveyor, as they could be standing still for a period before moving to the next waypoint. This can result in a waypoint being placed ahead of the surveyor's actual location. Additionally, if the surveyor is not marking a waypoint on the corner of a shop or other out-of-bounds areas, the interpolation technique can place an invalid WiFi reading.

D. Approximation by Step Calculation

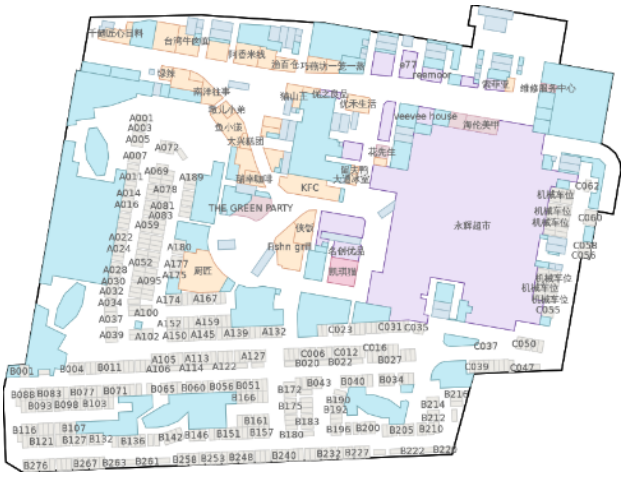
We also apply a more sophisticated approach, which includes further sensory data like the IMU and magnetometer readings in the approximation of positioning. Attempting to calculate the trace of a surveyor accurately is complex but can improve the approximation of where each WiFi reading is observed. For this approach, we generate a number of new waypoints for a trace by supplying the step calculation algorithm developed by XYZ10 [20] with the accelerometer data, rotation vector, and the original trace. Within the 24 buildings, we generate 666 934 new waypoints from the original 75 285. That results in an average time between each waypoint of approximately 712 ms compared to approximately 8488 ms originally. The new trace can then give us a more accurate representation of the surveyor's path by accounting for when the surveyor is, e.g., not moving or not taking straight paths. We incorporate the interpolation of subsection V-C between the points of the new trace to further place the WiFi readings in the correct positioning. With the inclusion of IMU, it is possible to deduce the heading and gait speed of the surveyor in between the approximated locations, hence obtain a more realistic trace.

E. Floor Representation

We propose a method of correcting predicted waypoints. Predictions are corrected by moving out-of-bounds waypoints, meaning they are outside the building or inside a wall or a shop, to a valid area. These methods incorporate that waypoints never occur inside shops, which may not be true for other datasets. However, the methods are usable for any indoor localization problem where out-of-bounds prediction placement is defined, such as walls or staff-only areas.

A coordinate representation of the designated building's floors and wall boundaries is a prerequisite for our methods. We make use of the included metadata in the dataset, as presented in subsection III-B. Each floor's layout is represented by a GeoJSON-formatted file, from which we create the polygons that make up the internal structure of each building's floor. The GeoJSON file specifies the overall floor area and each shop as separate polygons. We transform the GeoJSON coordinates to correspond with the waypoints, as the GeoJSON format specifies coordinates in the real world [21].

We do the transformation by calculating an affine transformation between the bounding boxes of each floor and the floor image, which we then apply to the floor polygons. The `floor_info.json` file in each floor's metadata folder specifies the measurements of the floor's bounding box in meters, which we use to calculate another affine transformation to bring the building polygons to the same scale as the waypoint coordinates. By using affine transformations, we ensure that all points of the polygons keep the same relations to each other before and after transforming them. This provides



(a) This shows an image of a floor that is included in the metadata.

(b) The black outline marks the overall floor, while each colored polygon is a separate shop.

Fig. 6: The polygons produced on the floor shown on Figure 6b using the GeoJSON file are shown in Figure 6a. Instead of representing the floor as pixels, we now have precise coordinates for every shop located on it.

us with some amount of polygons for each floor, where one polygon represents the floor's outline, and thus the total area of the floor, and the rest each represents a shop area. An example can be seen in Figure 6b.

F. Snap to Corridors

Our initial snapping approach firstly determines whether a predicted waypoint is placed within the floor's outside bounds, and secondly, whether it is inside the boundaries of a shop. If this is the case, we *snap* the corresponding waypoint to the closest location within bounds.

However, to make the method more realistic, we ensure a fixed distance between the wall and the corrected prediction, as the surveyors collecting the data are unlikely to have walked directly up against or inside the wall. By use of the polygons of each floor, we create new polygons that define the walkable areas of each floor. We designate them as the corridors of the floor and create them by removing the area of each shop polygon from the entire area of the floor plan. To make sure we eliminate any small spaces between shops, e.g., invalid areas we could snap waypoints to, we expand the polygon size of each shop by one centimeter in all directions before calculating the corridors. Lastly, we reduce the corridors' area by shrinking the corridor polygons by the fixed distance in all directions. Examples of this are shown in Figure 7.

G. Snapping with a Heuristic

This method follows the same logic as the previous, including the calculation and use of corridors, but also applies a heuristic that influences the placement of the poorly predicted waypoints. The location of other predictions guides this heuristic. We aim to snap the predictions within a realistic distance from the other predictions to prevent the situation depicted in Figure 8, where a prediction is on the other side of a wall than the other predictions.

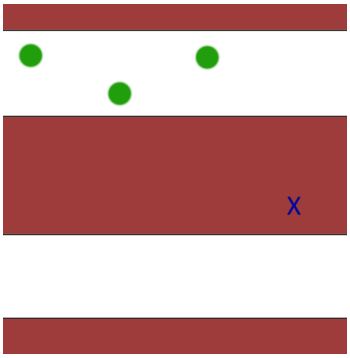
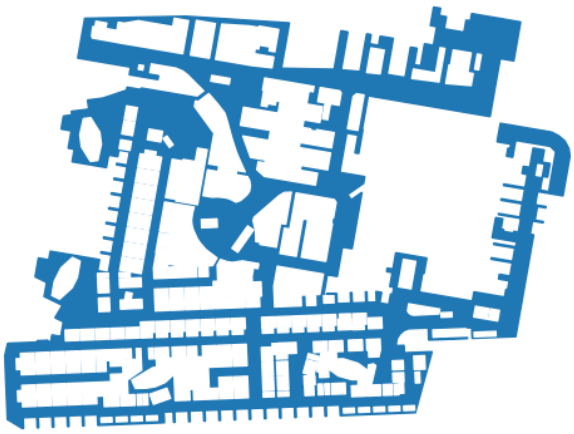


Fig. 8: The white areas indicate corridors, i.e., where we can validly place a waypoint. The red areas indicate out-of-bounds areas. The green points indicate already predicted waypoints. The blue *X* indicates a prediction that we want to correct since it is placed inside a shop. If we snap it to the nearest corridor, we predict that a surveyor has suddenly walked through a shop, which is not possible.

We handle each predicted waypoint individually, and each time we snap a waypoint to a corridor, we add it to a list of corrected predictions. Whenever we handle a prediction, we draw a line between all previously corrected predictions and expand it into a polygon. The polygon describes all points within some distance of the line, providing us with the heuristic of an area where the prediction should be. Since the now expanded line also describes parts of the floor that are not walkable, we calculate the intersection between the corridors and the extended line to get a new polygon that only has a walkable area within some given distance of the already correct waypoints. We then move the prediction to the nearest point described by this polygon. If we are handling the first prediction in a trace, we move it to the nearest point of the shrunk corridors, as in the previous method. Figure 9 shows an example of the different polygons calculated and used during this method.



(a) With no shrinking applied, the corridors corresponds to the areas highlighted in blue.

(b) With a shrinking of 1 meter, the corridors are smaller than in Figure 7a, hence parts of the previously walkable area are lost.

Fig. 7: The difference between Figure 7a and Figure 7b poses both pros and cons of our shrinking method. In Figure 7a the original corridors are highlighted. They also include thin lines between some shop polygons, which are unlikely to be walkable areas in the physical world. Shrinking addresses this issue and allows for more sensible predictions. However, if the chosen shrinking size is too large, it will remove the corridor entirely. Examples of this can be seen in Figure 7b, where some corridors have disappeared. Therefore, it is important to experiment to find the most optimal amount of shrinking, as a suboptimal amount can make snapping much worse.

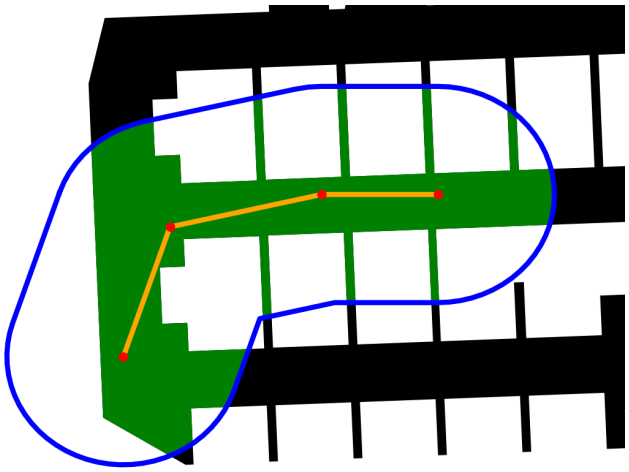


Fig. 9: The black area is the corridor area, to which we snap the predictions. The red points indicate predictions that are known to be correct. The blue outline is the polygon we create from the orange line between the red predictions, describing the initial boundaries that a prediction is allowed to be placed within. Since we do not want to place the predictions outside the floor, we calculate the green area as the intersection between the corridors and the blue polygon. Thus, it is the green area we snap the predictions to in the method presented in subsection V-G.

VI. Experiments

We now apply the methods from section V and conduct a series of experiments to measure their improvements of the MPE. Throughout this section, we will focus on using the data from the 24 test buildings once more. First, we will present our experiments when we generate more labeled waypoints, which seek to counteract the data flaws discovered in section IV. Then we present the process of tuning and training the models we use to make our predictions. Lastly, we explain the process of applying our postprocessing techniques on the predicted traces in order to improve their results.

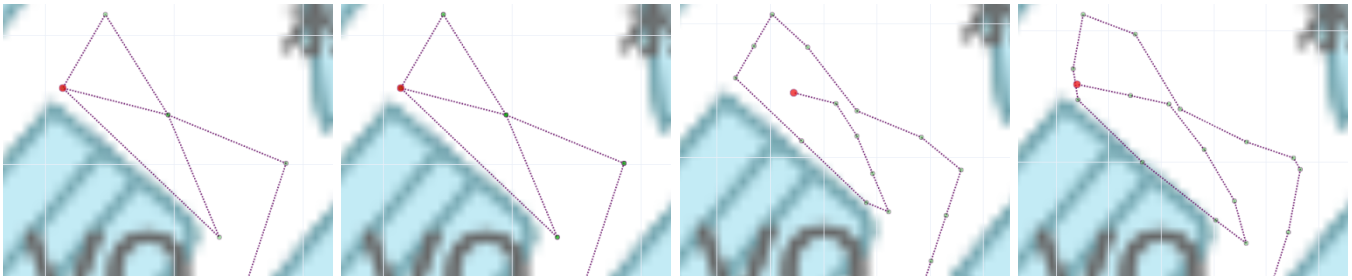
A. Feature Engineering

During section IV, we discovered a sparsity of waypoints among the 24 buildings, compared to a large amount of sensory data available. Due to the time difference, we consider it unreasonable to associate this sensory data with existing waypoints. In the following paragraphs, we present four feature engineering strategies to augment our dataset based on the methods proposed in section V. The augmented datasets are tested on our models and compared by their MPE. We describe this process further in subsection VI-B.

TABLE III: List of abbreviations for datasets.

Abbreviation	Expansion
BGT	Baseline Ground Truth
NWWR	Nearest Waypoint to WiFi Reading
IWR	Interpolation of WiFi Reading
ATI	Approximated Trace with Interpolation

The WiFi readings, which we presented in subsection V-A, are utilized by all of our datasets. There is a slight difference in the number of features within the datasets depending on the type of model. The dataset for classification models (kNN, SVM, LGBM) makes use of the entire RSSI vector, while the dataset for our deep learning models takes the 100 nearest BSSI units based on their RSSI value. This is further explained



(a) BGT dataset. **(b)** NWWR dataset **(c)** IWR dataset **(d)** ATI dataset

Fig. 10: Plotted traces from our four generated datasets. Every point denotes a waypoint. The dotted lines between traces are simply a visual aid for the reader to see the temporal connection between nodes.

in subsection VI-C. Below, we will go through each of them in greater detail. We refer to Table III for the reader to look up the abbreviations used to name our datasets.

- The BGT dataset serves as a baseline dataset, and it is made up of only the WiFi reading that is closest in time to the original waypoints, making it as close to ground truth data as possible.
- The NWWR dataset is created by utilizing all WiFi readings throughout each trace, in order to create additional waypoints. For each WiFi reading, the recorded timestamp is compared to the waypoint timestamps and passed the location of the closest waypoint. This dataset poses as a naive dataset with an increased amount of ground truth waypoints but with a less refined method of placement compared to the following two datasets. It has the same amount of data points as all following datasets to give a closer comparison between techniques.
- To create the IWR dataset, we use our WiFi reading interpolation as described in subsection V-C to estimate the location where each WiFi reading was recorded and label waypoints based on that.
- The ATI dataset includes the approximated waypoints we get from the step calculation method of subsection V-D. Similar to our interpolation method, we estimate each recorded WiFi reading's location, which gives us the same amount of data points as in the two previous datasets.

Finally, these datasets constitute four fingerprints of each of the 24 buildings. Figure 10 gives a side-by-side view of the placement and number of waypoints within each of the four datasets from the perspective of a single trace.

B. Model Training

We measure our data on well-known classification, regression, and deep learning algorithms. These include a set of kNN, SVM, LGBM, LSTM, and NN models. We cover such a variety of algorithms to illustrate that our pre and postprocessing strategies are universal. We go through several steps for each of these before we arrive at the final predictions.

Prior to training, we use common techniques to perform hyperparameter tuning, such as grid and random search, concerning each algorithm. From our generated datasets, we focus on predicting the floor and the x and y-coordinates. For each classification algorithm, i.e., kNN, SVM, and LGBM, we train three individual models per building, one for each of the aforementioned labels.

Regarding LSTM and NN, we train a single model to predict all three target labels for every building. During training, we use grouped 5-fold cross-validation to measure the performance of the models. We group the validation samples based on the trace ID, as this ensures that the model will not be able to *peek* into traces it has already trained on, and thereby be familiar with samples of the same trace. In each fold, we calculate the score using MPE. Our results are presented in section VII.

C. Generalized Models

We want our deep learning models to be a generalization tool for predicting indoor positioning. This implies that instead of having the dataset split into each building, we combine them. Having every BSSID from all buildings presented simultaneously produces a long and sparse vector, which is an undesirable input for our models. For each dataset introduced in subsection VI-A, we instead take the first 100 entries of each WiFi reading with RSSI values above -100. If there are less than 100 entries, we fill the rest of the vector with -999 values. We argue that this is an acceptable assumption to make, as a person will rarely reach and ping over 100 nearby WiFi units from the same location. In our experiments, 100 features per label provided sufficient coverage for fingerprinting a location. Along with the 100 RSSI readings, we also want to include each BSSID of the 100 values in the input vector for downstream label encoding and embedding.

Furthermore, we accommodate an alternative approach by having our deep learning models take arbitrary input from the buildings in the training set. To accomplish this, the NN and the LSTM have embedding layers for encoding the WiFi BSSID and the building IDs. We provide the model with a vector of the BSSID and a vector with the RSSI values in their correlated indices with this approach. Lastly, the three input layers are concatenated and fed into the prediction models, producing three individual outputs for x, y, and floor.

D. Postprocessing

We utilize the postprocessing technique presented in subsection V-G in order to improve our predictions. The snapping is

TABLE IV: The different models and datasets compared by MPE values. For the rows NWWR, IWR, and ATI, the number on the left-hand side is the MPE value, and the right-hand side number is the percentage decrease from the base value in the BGT row.

Dataset	Prediction Model										Mean	
	kNN		SVM		LGBM		LSTM		NN			
BGT	10.67		21.80		24.24		9.34		9.11		15.03	
NWWR	11.41	-6.92%	11.56	46.98%	10.05	58.53%	8.76	6.25%	8.48	6.93%	10.05	22.35%
IWR	**10.34**	**3.12%**	**10.91**	**49.94%**	**9.89**	**59.19%**	**8.10**	**13.29%**	**8.07**	**11.46%**	9.46	27.40%
ATI	10.73	-0.48%	11.46	47.43%	10.23	57.79%	8.22	12.04%	8.28	9.14%	9.78	25.18%

applied to the predictions from the trained models. A visualization of the predictions, the snapping, and the validation trace is presented on Figure 11. We conduct the experiments for this postprocessing technique on the predictions of our best- and worst-performing models in terms of MPE. For our snapping technique, we set the *shrinking* parameter, i.e., distance to each wall, to $0.3m$ and the max distance to other predictions to $5m$.

Additionally, we measure the results of applying our snapping technique to the ATI dataset *prior* to training the models. The reasoning is that through inspection of traces generated from the approximation using step calculation, we noticed waypoints placed out-of-bounds.

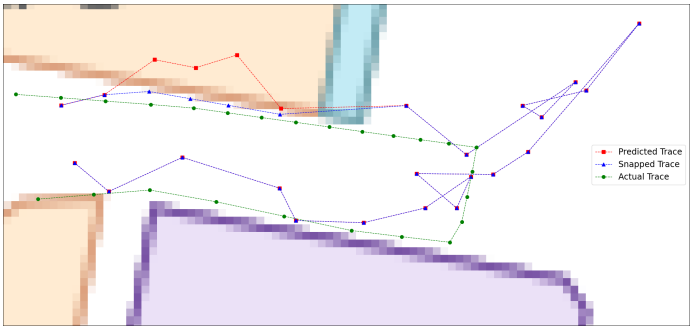


Fig. 11: The figure shows three traces: Raw predictions (red squares), snapped predictions (blue triangles), and the validation trace (green circles). We can see that snapping notably improves the trace by altering four predictions that it detected inside a shop. The rest of the predictions are unchanged. The dotted lines between traces are simply a visual aid for the viewer to see the temporal connection between nodes.

VII. Results

We present the results of our experiments to evaluate the performance of our proposed pre and postprocessing techniques. Firstly, we will look at the results from the predictions made on our four generated datasets to see how the additional waypoints affect the different algorithms. Lastly, predictions are compared to the results of applying our postprocessing technique to the predictions.

A. Prediction Scores

Table IV shows the MPE and its decrease in percentages of each model trained on each dataset described in subsection VI-A without using postprocessing techniques. From these results, we observe several patterns. In terms of model performance, the LSTM and NN models have a lower MPE than the others.

Our baseline dataset BGT performs worst. This is consistent with our expectations and thus proves that our subsequent data generation methods improve predictions. Especially our SVM and LGBM models seem to be affected by the lower amount of input features.

B. Postprocessing Effect

We compare the raw predictions to the predictions with our postprocessing applied to measure the effect of our snapping method.

We refer to Tables V & VI for the results of the experiments described in subsection VI-D. The *Before* column of both tables are values taken from Table IV.

TABLE V: The results of snapping applied to the predictions of our best performing models, LSTM and NN, trained on the IWR dataset. We also apply the postprocessing technique on our worst-performing model's predictions, LGBM, trained on the BGT dataset, to test its impact.

Model	Dataset	MPE			Wrong Floors
		Before	After	% Decrease	
LSTM	IWR	8.10	7.94	1.98%	4.0%
NN	IWR	8.07	8.05	0.24%	1.8%
LGBM	BGT	24.24	19.82	18.23%	6.1%

TABLE VI: MPE values when we apply snapping on the ATI dataset before we train the models.

Model	Dataset	MPE		
		Before	After	% Decrease
LGBM	**ATI-SNAP**	10.23	10.28	-0.49%
LSTM	**ATI-SNAP**	8.22	9.06	-10.21%

By way of example, we analyze five folds of predictions from the IWR dataset done by our LSTM model, as this combination yields the lowest MPE between our recorded results. These folds consist of 247 273 predicted waypoints in total, whereas 38.25%, corresponding to 94 574, are addressed by our snapping method. This is a noticeable amount of predicted waypoints, which have been predicted to be out-of-bounds.

Focusing on the snapped predictions, we illustrate the distribution of how far these waypoints are snapped from their predicted location in Figure 12 and Figure 13. We observe a median snapping distance of $1.98m$ in conjunction with a minimum distance of $0.30m$ and a maximum distance of $149.02m$. Figure 13b gives a full view of the snapped predictions, including outliers, hence the large y-axis values. Out of the 94 574 snapped predictions, 5 883 are considered outliers, with a snap distance over $7.61m$.

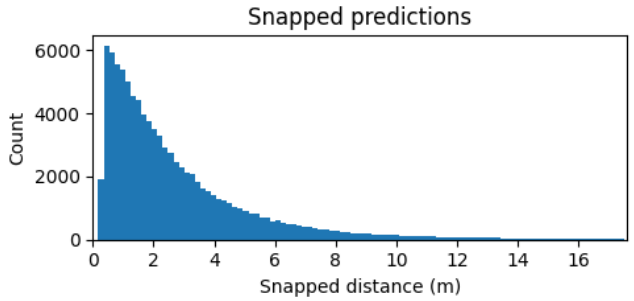
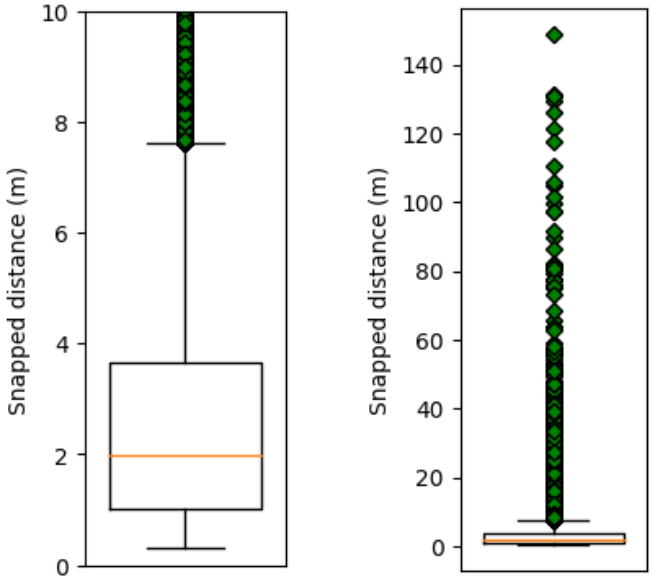


Fig. 12: Histogram of the snapped predictions for the IWR dataset using the LSTM model.



(a) Magnified view of the boxplot's minimum, median, and quartiles.

(b) Full view of the boxplot of the snapped predictions.

Fig. 13: Distance distribution of snapped predictions from the LSTM model on the IWR dataset from five folds.

VIII. Discussion

In this section, we will reason on our prediction results both prior to and after postprocessing. This involves a discussion on our methodology presented in section V and our experiments conducted in section VI. We will discuss both strengths and weaknesses of our approaches as well as plausible sources of error. Finally, we discuss ways to extend our current work in subsection VIII-A.

Contrary to our expected outcome, the waypoint interpolated dataset, IWR, yields better results than our approximated dataset, ATI, on all our models. We presumed that the step calculation technique would construct consistently more accurate traces than our linear interpolation technique due to the inclusion of IMU data. There can be several reasons for its underperformance, but an important aspect is the sensor calibration of the survey devices used to collect the original data. If the sensor readings used in our step calculation are indeed poorly calibrated, it can explain why the results contradict our expectations. The step calculation technique also includes a high number of hyperparameters, of which we have tested some combinations. There might be a set of values more fit for the data.

As presented in subsection VII-B, we witness a substantial number of outliers when examining the distances of how far our LSTM predictions are snapped from their predicted locations. Undoubtedly, these are caused by poor predictions from our model, in this case, the LSTM. In Table V, we observe that 4% of our predicted waypoints are labeled with the wrong floor. When the snapping method alters predictions relative to the corridors created from the floor metadata, it can cause predicted waypoints with an incorrect floor label to be snapped to a different floor map. Since most longer traces' predictions rarely mislabel the floor, we could include an initial step to our snapping method prior to the actual snapping. This step could involve aggregating a trace's floor labels based on the majority of predicted labels. This would ensure that the entire trace would stay on the same floor and possibly remedy a part of the issue.

Regarding the performance of our snapping technique, we implemented *distance to previous waypoints* as part of the snapping heuristic as a security measure to minimize drastic snapping of predictions. Nonetheless, Figure 14 shows how a max distance too large can cause the snapping method to allow snapping predictions far from previous predictions.

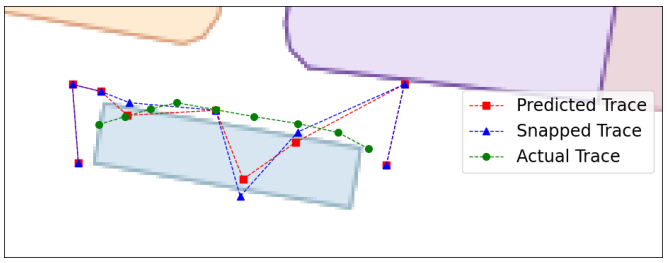


Fig. 14: An example of the snapping method, snapping a point to the wrong side of an out-of-bounds area.

On the contrary, a low line distance also has vulnerabilities. It makes the snapping reliant on previous waypoints in the predicted trace, thus create a chain reaction of incorrect snapping if the first waypoint of the trace has a bad label. Instead of the snapping method relying on all previous predictions, when calculating where the next prediction can realistically be placed, we instead tried implementing a buffer only consisting of the last four points. Another approach we took was implementing a random restart to the buffer to avoid reliance on the previous prediction. Neither of these approaches yielded meaningful results and only increased the MPE.

Another interesting approach would be to experiment with a topology-preserving snapping method that would consider the entire trace when snapping. With our current approach, we might disturb the curvature and direction of the traces. We illustrate the desired result of this approach in Figure 15.

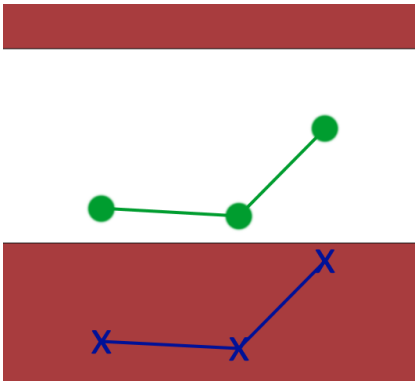


Fig. 15: An example of the alternative snapping method, where the topology between the corrected predictions are preserved. The blue X's represent the predictions before snapping, while the green circles represent the predictions after snapping.

This shows how the method would still make use of our corridors to denote walkable areas while also preserving the topology of the predicted trace.

A. Future Works

The approximation and snapping methods seem to improve our score to a minor extent, but we are confident that the technique allows for greater results. We have seen examples of snapped traces that would achieve lower MPE if the proper hyperparameter values, i.e., line distance and shrinking size, were applied. Due to the nature and size of the different floor plans, no static, one-size-fits-all value exists for these hyperparameters. The same applies to the step calculation. Alternatively, it can be interesting to further develop our snapping method's heuristic by finding a way to dynamically calculate the proper values for the parameters, to which we have already discussed several possibilities above.

To summarize, we have discussed several points of improvement for our various techniques and propose different strategies to remedy parts of their shortcomings. We have succeeded in developing a WiFi-reading interpolation method, applicable for general use when surveying indoor locations for RSSI fingerprinting. This mitigates the need for meticulous data collection of the indoor areas, as it allows the surveyor only to label critical parts that denote the topology of the trace, e.g., when shifting direction. Thus, it eases the surveying workload, which is an essential preliminary step regarding indoor location navigation.

IX. Conclusion

In this study, we have analyzed indoor localization data and proposed several data augmentation methods, as well as a postprocessing technique. Through experiments, we document the effect of said data augmentation and postprocessing. The experiments show a clear improvement in most models' accuracy using any augmentation method. The postprocessing technique also improves the accuracy of predictions made on data augmented by any method proposed in this study. Ultimately, we found that the best performing data augmentation method was interpolating WiFi readings based on the Spatio-temporal difference between ground truths, and our best performing model after postprocessing was an LSTM. Additionally, our postprocessing technique suffers from diminishing returns as the models' mean position error decreases.

References

[1] Y. Li, Z. He, J. Nielsen, and G. Lachapelle, "Using wi-fi/magnetometers for indoor location and personal navigation," 10 2015.

[2] A. Khalajmehrabadi, N. Gatsis, and D. Akopian, "Modern wlan fingerprinting indoor positioning methods and deployment challenges," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1974–2002, 2017.

[3] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 32–43. [Online]. Available: https://doi.org/10.1145/345910.345917

[4] B. Sohn, J. Lee, H. Chae, and W. Yu, "Localization system for mobile robot using wireless communication with ir landmark," in *Proceedings of the 1st International Conference on Robot Communication and Coordination*, ser. RoboComm '07. IEEE Press, 2007.

[5] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with rfid technology," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 1, 2004, pp. 1015–1020 Vol.1.

[6] S. Xia, Y. Liu, G. Yuan, M. Zhu, and Z. Wang, "Indoor fingerprint positioning based on wi-fi: An overview," *ISPRS International Journal of Geo-Information*, vol. 6, p. 135, 04 2017.

[7] O. Belmonte-Fernandez, E. Sansano-Sansano, A. Caballer-Miedes, R. Montoliu, R. Garcia-Vidal, and A. Gasco-Compte, "A generative method for indoor localization using wi-fi fingerprinting," *Sensors*, vol. 21, no. 7, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/7/2392

[8] E. Martin, O. Vinyals, G. Friedland, and R. Bajcsy, "Precise indoor localization using smart phones," in *Proceedings of the 18th ACM International Conference on Multimedia*, 10 2010, pp. 787–790.

[9] L. Aalto, N. Göthlin, J. Korhonen, and T. Ojala, "Bluetooth and wap push based location-aware mobile advertising system," in *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 49–58. [Online]. Available: https://doi.org/10.1145/990064.990073

[10] J. Xiong, K. Sundaresan, and K. Jamieson, "Tonetrack: Leveraging frequency-agile radios for time-based indoor wireless localization," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 537–549. [Online]. Available: https://doi.org/10.1145/2789168.2790125

[11] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 173–184. [Online]. Available: https://doi.org/10.1145/1859995.1860016

[12] D. Mascharka and E. Manley, "Lips: Learning based indoor positioning system using mobile phone-based sensors," *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Jan 2016. [Online]. Available: http://dx.doi.org/10.1109/CCNC.2016.7444919

[13] R. S. Sinha, S.-M. Lee, M. Rim, and S.-H. Hwang, "Data augmentation schemes for deep learning in an indoor positioning application," *Electronics*, vol. 8, no. 5, 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/5/554

[14] R. S. Sinha and S.-H. Hwang, "Improved rssi-based data augmentation technique for fingerprint indoor localisation," *Electronics*, vol. 9, no. 5, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/5/851

[15] R. S. Sinha, S.-M. Lee, M. Rim, and S.-H. Hwang, "Data augmentation schemes for deep learning in an indoor positioning application," *Electronics*, vol. 8, no. 5, 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/5/554

[16] M. Research. Data description. Accessed: 11-02-2021. [Online]. Available: https://www.kaggle.com/c/indoor-location-navigation/data

[17] Apple. Getting started with ibeacon. Accessed: 03-05-2021. [Online]. Available: https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf

[18] R. Bohannon and A. Andrews, "Normal walking speed: A descriptive meta-analysis," *Physiotherapy*, vol. 97, pp. 182–9, 09 2011.

[19] M. Research. Evaluation. Accessed: 18-03-2021. [Online]. Available: https://www.kaggle.com/c/indoor-location-navigation/overview/evaluation

[20] XYZ10. compute_f.py. Accessed: 23-05-2021. [Online]. Available: https://github.com/location-competition/indoor-location-competition-20/blob/master/compute_f.py

[21] B. H. et al. The geojson format. Accessed: 29-04-2021. [Online]. Available: https://tools.ietf.org/html/rfc7946

[22] S. Kaufman, S. Rosset, and C. Perlich, "Leakage in data mining: Formulation, detection, and avoidance," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 556–563. [Online]. Available: https://doi-org.zorac.aub.aau.dk/10.1145/2020408.2020496

[23] E. Martin, O. Vinyals, G. Friedland, and R. Bajcsy, "Precise indoor localization using smart phones," in *Proceedings of the 18th ACM International Conference on Multimedia*, ser. MM '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 787–790. [Online]. Available: https://doi.org/10.1145/1873951.1874078

[24] M. Research. Overview. Accessed: 11-02-2021. [Online]. Available: https://www.kaggle.com/c/indoor-location-navigation/overview

[25] N. A. R. Henry. Simple 99% accurate floor model. Accessed: 18-03-2021. [Online]. Available: https://www.kaggle.com/nigelhenry/simple-99-accurate-floor-model

[26] T. Inubushi. Postprocessing based on leakage. Accessed: 13-04-2021. [Online]. Available: https://www.kaggle.com/tomooinubushi/postprocessing-based-on-leakage

[27] ——. Retrieving user id from leaked wifi feature. Accessed: 13-04-2021. [Online]. Available: https://www.kaggle.com/tomooinubushi/retrieving-user-id-from-leaked-wifi-feature#Get-user-ID-based-on-leaked-common-records

[28] A. Palen. nudged 0.3.1. Accessed: 21-04-2021. [Online]. Available: https://pypi.org/project/nudged/

[29] S. Gillies. Shapely 1.7.1. Accessed: 21-04-2021. [Online]. Available: https://pypi.org/project/Shapely/

[30] ——. The shapely user manual - difference. Accessed: 29-04-2021. [Online]. Available: https://shapely.readthedocs.io/en/stable/manual.html#object.difference

[31] ——. The shapely user manual - buffer. Accessed: 29-04-2021. [Online]. Available: https://shapely.readthedocs.io/en/stable/manual.html#object.buffer

[32] XYZ10. Indoor location competition 2.0 (sample data and code). Accessed: 12-02-2021. [Online]. Available: https://github.com/location-competition/indoor-location-competition-20

X. APPENDICES

A. Data Structure

TABLE VII: Overview of data structure.

Time	Data Type	Value							
1574659531598	TYPE_WAYPOINT	196.41757	117.84907						
	Location surveyor labeled on the map	Coordinate x (meter)	Coordiante y (meter)						
1574659531695	TYPE_ACCELEROMETER	-1.7085724	-0.274765	16.657166	2				
	Android Sensor.TYPE_ACCELEROMETER	X axis	Y axis	Z axis	accuracy				
1574659531695	TYPE_GYROSCOPE	-0.3021698	0.2773285	0.107543945	3				
	Android Sensor.TYPE_GYROSCOPE	X axis	Y axis	Z axis	accuracy				
1574659531695	TYPE_MAGNETIC_FIELD	20.181274	16.209412	-32.22046	3				
	Android Sensor.TYPE_MAGNETIC_FIELD	X axis	Y axis	Z axis	accuracy				
1574659531695	TYPE_ROTATION_VECTOR	-0.00855688	0.051367603	0.362504	3				
	Android Sensor.TYPE_ROTATION_VECTOR	X axis	Y axis	Z axis	accuracy				
1574659531695	TYPE_ACCELEROMETER_UNCALIBRATED	-1.7085724	-0.274765	16.657166	0.0	0.0	0.0	3	
	Android Sensor.TYPE_ACCELEROMETER_UNCALIBRATED	X axis	Y axis	Z axis	X axis	Y axis	Z axis	accuracy	
1574659531695	TYPE_GYROSCOPE_UNCALIBRATED	-0.42333984	0.20202637	0.09623718	-7.9345703E-4	3.2043457E-4	4.119873E-4	3	
	Android Sensor.TYPE_GYROSCOPE_UNCALIBRATED	X axis	Y axis	Z axis	X axis	Y axis	Z axis	accuracy	
1574659531695	TYPE_MAGNETIC_FIELD_UNCALIBRATED	-29.830933	-26.36261	-300.3006	-50.012207	-42.57202	-268.08014	3	
	Android Sensor.TYPE_MAGNETIC_FIELD_UNCALIBRATED	X axis	Y axis	Z axis	X axis	Y axis	Z axis	accuracy	
1574659533190	TYPE_WIFI	intime_free	0e:74:9c:a7:b2:e4	-43	5805	1574659532305			
	Wi-Fi data	ssid	bssid	RSSI	frequency	last seen timestamp			
1574659532751	TYPE_BEACON	FDA50693-A4E2-4FB1-AFCF-C6EB07647825	10073	61418	-65	-82	5.50634293288929	6B:11:4C:D1:29:F2	1574659532751
	iBeacon data	UUID	MajorID	MinorID	Tx Power	RSSI	Distance	MAC Address	same with Unix time, padding data