IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 27 August 2024, accepted 5 October 2024, date of publication 14 October 2024, date of current version 28 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3480828

RESEARCH ARTICLE

Addressing Temporal RSSI Fluctuation in Passive Wi-Fi-Based Outdoor Localization

FANCHEN BAO, STEPAN MAZOKHA, AND JASON O. HALLSTROM
Institute for Sensing and Embedded Network Systems Engineering (I-SENSE), Florida Atlantic University, Boca Raton, FL 33431, USA
Corresponding author: Jason O. Hallstrom (jhallstrom@fau.edu)

This work was supported by the NSF Engineering Research Center for Smart Streetscapes under Award EEC-2133516.

ABSTRACT Passive outdoor localization is valuable in understanding pedestrian mobility patterns for city planning and other purposes. While there are various approaches to Wi-Fi localization, few demonstrate robustness to temporal fluctuations in RSSI measurements. In this paper, we continue the work on the Mobility Intelligence System (MobIntel) and address RSSI temporal fluctuation via two approaches. First, given a rectangular testbed bounded by four sensors, we reformulate the problem from localizing the emitters from anywhere within the testbed to classifying whether they originate from the North or South strip and estimating their East-West position on the identified strip. This change simplifies the problem, reduces the impact of RSSI fluctuations, and better resembles how pedestrians move in city streets. Second, we present a multi-stage model (*InXModel*) to perform localization based on the reformulated problem. Within this model, we compare the performance of four data transformation methods that further dampen the effect of RSSI fluctuation—basic standardization (STD), Kernel Principal Component Analysis (KPCA), Transfer Component Analysis (TCA), and Semi-Supervised TCA (SSTCA). Data resistant to improvements via transformation are discarded to preserve model performance. We observe that the *InXModel* with STD is the fastest and most accurate model, achieving a localization error $\leq$ 4 m in 94.3% of the cases with a 30.2% data discard rate. Finally, we compare the *InXModel* with a recently published method (EMDT-WKNN) specialized for handling RSSI temporal fluctuations and find that the former outperforms the latter. We discuss the causes of the difference in performance.

INDEX TERMS Passive outdoor localization, RSSI fluctuation, Wi-Fi probe requests, mobility intelligence.

I. INTRODUCTION

Received Signal Strength Indicator (RSSI) measurements have been used for WiFi-based localization in both indoor [2] and outdoor [3] environments. RSSI is defined within the 802.11 protocol and is supported in all off-the-shelf WiFi hardware. While popular for passive device localization, the nonlinear and non-stationary nature of RSSI measurements due to device power fluctuation, multi-path effects, interference, and environmental factors can yield different data distributions over time, even if the data collection conditions are unchanged [4]. Such fluctuations can pose a severe challenge to the robustness of a localization model [5], particularly as the temporal distance between the training dataset and the test dataset increases.

In this paper, we address the challenges of temporal fluctuations in RSSI with respect to the performance of passive outdoor localization via two steps: We first reformulate the problem from localizing emitters within a rectangular testbed bounded by four sensors to first classifying whether the emitters originate from the North or South strip, and then estimating their East-West position on the identified strip. Under this reformulation, we introduce a robust multi-stage localization model that performs well even when the test data are collected months after the training data – the *Inlier X-coordinate Model (InXModel)*. We develop the model under the constraint that it be scalable and readily applied in our production environment on Clematis Street, West Palm Beach, FL, where more than 50 sensors have been

The associate editor coordinating the review of this manuscript and approving it for publication was Nafees Mansoor.

deployed along the stretch of five city blocks. We focus exclusively on the 2.4 GHz band due to its ubiquitous use across municipalities. The choice is also motivated by hardware limitations within our existing streetscape testbed. The specific contributions of the paper include the following:

1) We establish a new data collection layout (i.e., how the data collection points are distributed in the testbed)—the split data layout—to accommodate the reformulated problem. The split data layout designates the reference points (RPs), where the training data are collected, only along the Northern and Southern strips of the testbed, similar to how pedestrians typically move on a city street along the two sidewalks. We argue that for pedestrian mobility patterns, it is sufficient to predict which sidewalk the pedestrian is on is sufficient and the relative distance between the pedestrian and a designated sensor on the same sidewalk is sufficient. Both predictions are easier to make than the traditional approach of predicting two-dimensional coordinates. We provide qualitative and quantitative evidence supporting the problem reformulation and explain why the split data layout reduces the impact of RSSI fluctuations.
2) We introduce the concept of inliers and outliers based on the relative position of an RP or a test point (TP) to the sensors. A TP is where the test data are collected. We show that data collected on the inliers, defined as those located within the sensor boundary, are the least affected by RSSI fluctuations and, thus, the best candidates for training a localization model.
3) We propose the new, robust, multi-stage Inlier X-coordinate Model (*InXModel*) for localization. The model includes a three-step inlier classification (pre-screening + inlier classifier + post-screening) to isolate inliers, followed by a one-dimensional inlier regression that predicts the distance between the inlier and a designated sensor.
4) We incorporate four data transformation methods—basic standardization (STD), Kernel Principal Component Analysis (KPCA), Transfer Component Analysis (TCA), and Semi-Supervised TCA (SSTCA)—into the *InXModel* to further reduce the impact of RSSI temporal fluctuation and compare their performance and hyperparameter tuning efficiency. We find that STD is the most efficient and performs the best.
5) We compare the performance of the *InXModel* with that of a recently reported method: Empirical Mode Decomposition Threshold smoothing (EMDT) and Weighted K-Nearest Neighbor (WKNN), or EMDT-WKNN. The EMDT-WKNN method is explicitly designed to handle RSSI fluctuations in the context of indoor localization [6]. We show that the EMDT-WKNN method does not perform as well as our *InXModel* using our dataset and discuss the difficulties of adapting a model for indoor localization to outdoor localization.

Paper Organization. Section II provides additional background on the impacts of RSSI fluctuation and provides a brief review of prior work addressing these impacts. Section III describes the main methods used in the paper, including the four data transformation methods, algorithms used for classification and regression, our adaptation of the EMDT-WKNN method, and hyperparameter tuning. Section IV describes the data collection procedure, explains the difference between the two data collection layouts—broad and split—, and explains how the split data layout overcomes the shortcomings of the broad data layout. Section V presents two unsuccessful attempts at rescuing the performance of our prior models, with a summary of key lessons learned. Section VI steps through the conceptualization of the new model. Section VII details the efforts of achieving inlier classification: separating inliers from outliers. Section VIII reports the results of inlier regression: predicting the distance from an inlier to a specified sensor. Section IX combines inlier classification and regression to show the overall performance of the *InXModel* and determines the best data transformation method. Section X compares the performance of the *InXModel* with the EMDT-WKNN method and discusses its implications. Finally, Section XI concludes by presenting limitations and directions for future work.

II. BACKGROUND AND RELATED WORK

The challenge of RSSI fluctuation in the context of localization has been observed with various signal sources (e.g., Wi-Fi, Bluetooth Low Energy) in indoor and outdoor environments. This section provides additional background on the associated challenges and surveys how prior literature has dealt with these challenges. We also highlight why prior methods may or may not be appropriate to meet our needs for passive outdoor localization.

A. BACKGROUND

Fig. 1 illustrates a case of RSSI temporal fluctuation in the real world from January through June 2023, observed in our Mobility Intelligence System (MobIntel) deployed in downtown West Palm Beach, FL [7]. During the six-month period, the emitter **s25** sent ten mock probe requests at 08:30 EST every day, while the four sensors, **s27**, **s53**, **s55**, and **s56**, captured the probe requests and recorded the associated RSSI measurements. Fig. 1(A) shows the fluctuation of the mean RSSI obtained by the sensors every day. The x- and y-axis represent the time of probe request emission and the RSSI measurement, respectively. The blue, orange, green, and red curves represent the RSSI measurements captured by s27, s53, s55, and s56, respectively. The gaps in the curve are due to sensors being offline at the time. Fig. 1(B) shows the relative positions of the emitter and the sensors. Although all devices are stationary, mounted on light poles at least three meters above the ground

(little interference from ground traffic and always line-of-sight), and all measurements were taken at the same time of day (minimum environmental variation), there is still approximately 10 to 20 dBm of fluctuation in mean RSSI for all sensors during the measurement period.

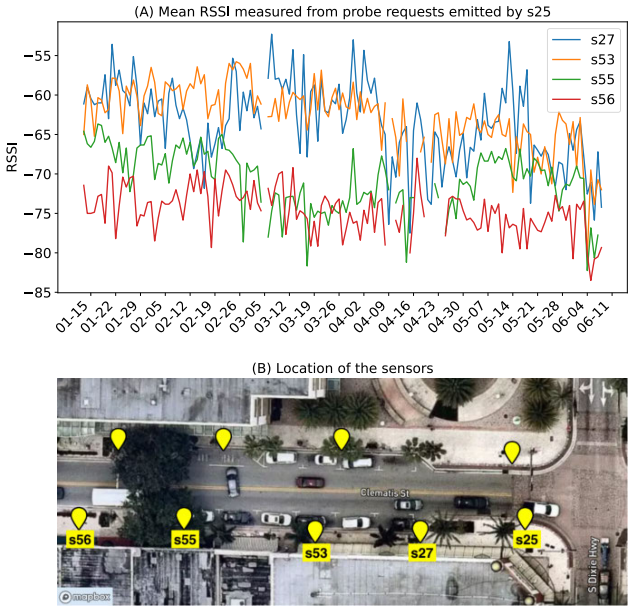


FIGURE 1. RSSI fluctuation over six months among MobIntel sensors.

The impact of temporal fluctuations in RSSI on a localization model's performance can be significant. Take the best models in our previous study as examples [3]. Fig. 2 shows the cumulative density function (CDF) curves of localization error across the three models—fingerprinting (*fp*), machine learning classification (*ml-clf*), and machine learning regression (*ml-regr*)—evaluated in a broad data layout under different testing conditions (see Section IV-A for the definition of the broad data layout). The x-axis represents the localization error, computed as the Euclidean distance between the predicted and actual emitter locations, respectively. The y-axis represents the cumulative probability of a model achieving a localization error less than or equal to a specific x-axis value. The model with the CDF curve that has the highest area-under-the-curve has the best performance. The *fp* method matches the RSSI pattern of an unknown location to previously collected training signatures at various reference points (RPs). The average positions of the best matching training signatures is the predicted location. The *ml-clf* method trains a classifier with the RSSI values as features and their corresponding RPs as labels. The location of an unknown RSSI pattern is predicted directly by the classifier. The *ml-regr* method trains two regressors for the x- and y-coordinates of the RPs, respectively, with their RSSI values as features. After passing an unknown RSSI pattern to the two regressors, we obtain the prediction of its x- and y-coordinates. Curves *fp-same*, *ml-clf-same*, and *ml-regr-same* represent the localization performance of the three models trained and tested on the same data cohort collected on 2021-04-01 (70% training, 30% testing), whereas curves *fp-diff*, *ml-clf-diff*, and *ml-regr-diff* represent the localization performance of the models trained on the 2021-04-01 data, but tested on the 2021-07-22 data. Since the two data collection sessions use the same devices, take place at the same location, and follow the same procedures, the primary cause of the drastic performance deterioration is temporal fluctuation in RSSI associated with the three-month gap.

Although RSSI fluctuation can severely compromise the robustness of a localization model, it is not an issue commonly addressed in recent literature [8], [9], [10]. One study explicitly tries to account for the problem by extending the data collection window over several days [11]. While multi-day RSSI collection takes into account fluctuations during the data collection window, the common practice of extracting both the training and test data from the *same* dataset might still overestimate the model performance because the variability in the current dataset is likely not representative of the RSSI measured at a later time. The underlying principle of splitting data into training and tests requires that the random variables have stable distributions, such that data collected today is representative of data collected tomorrow. Unfortunately, this principle does not apply to RSSI measurements due to their nonlinear and non-stationary nature. Therefore, to more robustly evaluate the performance of an RSSI-based localization model and avoid overfitting, we advise that the training and test data be collected in separate sessions, preferably several days or weeks apart.

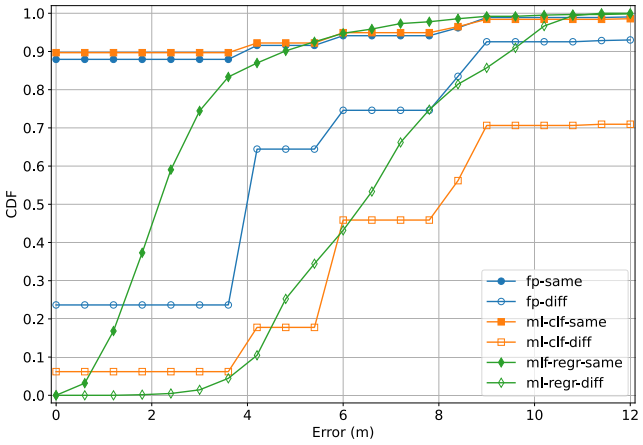


FIGURE 2. Localization performance comparison: Training and test data from the same session vs. training and test data three months apart.

Another exacerbating factor is that we focus on passive outdoor localization in our study, which precludes the application of various tools that have been shown effective at tackling RSSI fluctuation. Generally speaking, RSSI-based device localization can be indoors or outdoors, active or passive. Indoor localization takes place inside buildings. Given its easier access to infrastructure, indoor localization typically involves many more sensors than its outdoor

counterparts. The sheer abundance of data opens doors to more sophisticated methods to counter fluctuation (e.g., deep learning [12], [13]).

Active localization refers to scenarios where the device to be localized is actively involved in the localization process. The device can take RSSI measurements on its own from signals sent by the nearby access points and ameliorate RSSI fluctuation by incorporating additional information (e.g., device orientation from a magnetometer [14]). Alternatively, it can communicate directly with the access points and obtain more stable signals, such as time of flight, angle-of-arrival, etc., to compensate for RSSI variability [15], [16]. Even if the device does not measure RSSI or communicate with the access points, it can still maintain a constant data transmission rate such that the collected RSSI measurements constitute rich time-series data, which unlocks methods such as Recurrent Neural Networks and Kalman Filters to smooth RSSI signals [17], [18].

Passive localization, on the other hand, refers to scenarios where the target device has no knowledge of being localized, does not exchange any data with the sensors, and emits signals in an unpredictable manner. It does not benefit from the luxury of leveraging any of the above-mentioned methods. Consequently, passive outdoor localization is doubly challenged.

B. FINGERPRINTING OVER CROWDSOURCED MAPS

Fingerprinting is a standard method in RSSI-based localization. It includes an offline and an online phase. During the offline phase, a vector of RSSI measurements collected at an RP serves as the RSSI pattern for the RP. For active localization, RSSI is measured by the device to be localized from the signals sent by the access points. For passive localization, RSSI is measured by the sensors from the signals sent by the device to be localized. RSSI measurements are collected on many RPs to generate a radio map, which maps one or more RSSI patterns to each RP. During the online phase, a device at an unknown location produces an RSSI pattern, which matches some signatures in the radio map by similarity (usually Euclidean distance). The average location of the RPs associated with the matched signatures is the predicted location of the device.

Reliance on a static radio map could result in deteriorating performance over time. The more time that elapses between the online and offline phases, the more divergent the RSSI collected in the online phase will be from the radio map. Constant updates of the radio map can prevent the performance slip, but usually at a prohibitive cost of time and effort.

This is where crowdsourcing provides value. Crowdsourcing refers to a method where normal mobile device users help construct the radio map by reporting RSSI measurements and their physical locations while conducting their own business. As long as the area of interest has a constant flow of visitors, the radio map can stay up-to-date, and the issue of RSSI fluctuations can be mitigated.

The use of crowdsourcing to construct radio maps has been reported before. Reference [19] shows that by using crowdsourced RSSI data, a Local Gaussian Process model, and a map of the area of interest, it is possible to produce virtual radio maps for both surveyed and nearby unsurveyed areas, yielding a localization error of 1.84 m for the former and 5.75 m for the latter. However, this method has a major drawback where a user must actively report their current physical location upon submitting RSSI measurements. In the study, this was achieved by the user double-tapping the map on a custom-built crowdsourcing app.

To reduce user involvement, [20] exploits motion sensor data within the mobile device to automatically obtain information about physical location. A crowdsourcing user only needs to report their initial location and can then roam the area of interest without further engagement with the data reporting mechanisms. The study fuses motion sensor data with RSSI measurements via Extended Kalman Filtering and uses the fused data to generate the radio map. Their system yields less than 2.5 m error in 90% of the cases. While this method largely limits user collaboration, it demands motion sensor data from the device, which makes it incompatible with passive localization.

Reference [21] reports an unsupervised learning method that does away with the need for a crowdsourcing device to report location-related information. It uses RSSI measured at a high frequency (1 Hz) and the physical constraints of the indoor structure (e.g., walls and partitions) to optimally predict the crowdsourcing device's location as it roams the area of interest. The predicted location and its accompanying RSSI measurements are used to generate the crowdsourced radio map. The study reports an average localization error of approximately 3 m. Unfortunately, this method is still unsuitable for passive localization due to the high frequency of data reports. In Wi-Fi-based passive localization, RSSI is measured from spontaneously emitted probe requests. It is impossible to achieve a consistently high emission rate from a standard, non-cooperative device because Wi-Fi probe requests are sent in a manufacturer-dependent, unpredictable, and sporadic manner [22].

In summary, although crowdsourcing is a viable method for active localization to keep the radio map fresh, it is not applicable to passive localization.

C. RSSI SMOOTHING

RSSI fluctuation manifests as peaks and troughs when viewed as a time series. One way to reduce noise in RSSI data is to smooth the data. If RSSI data are collected at high frequency, a smoothing algorithm such as Kalman Filtering can leverage the correlation between adjacent data points to smooth the fluctuations. To obtain such high-frequency RSSI data, one can approach it either statically or dynamically.

Static, high-frequency RSSI measurements are collected at the same RP over a short time. For an indoor environment, RSSI collected in this way exhibits fluctuations due to

complex RF dynamics, and they can be smoothed by Kalman Filtering [18], [23]. However, RSSI fluctuation is not common at the same RP over a short period of time (e.g., within the same data collection session) in an outdoor environment. Typically, outdoor RSSI fluctuation happens over a large time scale (see Fig. 1), in which Kalman Filtering cannot help due to the associated errors not having a Gaussian distribution.

Dynamic, high-frequency RSSI measurements are typically collected over short intervals along a trajectory. In this case, RSSI fluctuation has both a temporal and spatial component. Reference [24] addresses this case in a passive outdoor localization setting similar to ours. The work relies on Constant Velocity Kalman Filtering to smooth real-time RSSI measurements as a pedestrian moves across a sensor-covered area. Unfortunately, the requirement of RSSI reporting at 1 Hz precludes the use of this method on non-collaborating devices due to the unpredictability of the emission rate of Wi-Fi probe requests in a real-world setting.

While smoothing short-term time-series RSSI data is incompatible with passive localization, it is possible to smooth long-term RSSI data collected at the same RP across multiple data collection sessions. Despite not having a uniform interval between all RSSI measurements, the data can be decomposed via Empirical Mode Decomposition (EMD) and smoothed by removing high-frequency components [25]. Moreover, if the time-series data span a sufficiently long period, the data may include a wide range of RSSI patterns to have high matching probabilities against any future fluctuations. Thus, empirically smoothed, long-term, time-series RSSI data can form a solid basis for a robust fingerprinting radio map. This is precisely the idea behind [6], which uses Empirical Mode Decomposition with soft Thresholding (EMDT) to decompose and smooth RSSI data collected over three months. The smoothed data are used to build a radio map, and WKNN is applied for localization. The method yields an average localization error of 1.52 m, with less than 2.5 m error in 87.44% of the cases.

Although the EMDT-WKNN method is designed for active indoor localization, its independence of user collaboration, device communication, and data collection frequency makes it readily adaptable to passive outdoor localization. Hence, we apply this method on our dataset and compare its performance with the *InXModel* in Section X.

D. TRANSFER LEARNING

Transfer learning refers to a machine learning technique to resolve the problem where the source domain (training data) does not share the same data distribution as the target domain (test data), and consequently, a model trained on the former is inapplicable to the latter. This is precisely the issue when the training and test RSSI data are collected at different (distant) points in time. Transfer learning can be a tool to help bridge the divergence in RSSI data distributions [5].

Typically, transfer learning algorithms extract features common to both the source and target domain and train models based on the shared features. In [26], the shared features are called transfer components, which are found via an unsupervised learning algorithm, TCA, in a reproducing kernel Hilbert space (RKHS). These transfer components allow the differences between the training and test RSSI data distributions, computed as Maximum Mean Discrepancy (MMD), to be minimized when they are projected to the subspace spanned by the transfer components. The same study also proposes SSTCA, which maximally preserves the dependency between the training input (RSSI) and the label (location of RP) while searching for the optimal transfer components. TCA and SSTCA perform better than other commonly used data transformation methods, with SSTCA slightly outperforming TCA in most cases.

Reference [27] uses Transfer Kernel Learning (TKL) to learn a domain-invariant kernel across the source and target domains. The goal of TKL is to minimize the distribution difference between the source and target domains in RKHS, but in contrast to TCA, TKL does not rely on MMD for computing the distribution difference. Instead, it defines the distribution difference as the Nyström approximation error between the extrapolated kernel (the kernel matrix computed from the combination of the source and target domain) and the ground truth kernel (the kernel matrix computed just from the source domain). The identifying domain-invariant kernel can be directly used as input for a regression algorithm to predict location. Interestingly, the study finds that localization performance improves significantly if APs at known locations are enabled to calibrate RSSI measurements among themselves in real time and incorporate the calibrated data in the source domain.

One can also consider RSSI-based localization as an image recognition problem, i.e., representing an RP by a 2D matrix of RSSI values instead of a 1D vector and tackling it with deep learning. With deep learning, one can obtain a feature extraction layer by training against the RSSI images. Later, one can apply the same feature extraction layer to the training and test data to acquire the shared hidden features. These features can then be connected to a traditional classification algorithm to finalize localization. The process of obtaining and reusing a feature extraction layer to find common ground between the training and test data is a key aspect of deep transfer learning, and a scheme has been adopted by [11] to attenuate fluctuation in the concatenated signal of RSSI measurements and local magnetic field strength data, and by [28] to alleviate variation in Wi-Fi channel state information.

Among the transfer learning techniques mentioned above, deep transfer learning is unlikely to be applicable to passive localization due to the demand for a high AP count and data quantity. TKL is tempting, but repurposing existing sensors for RSSI calibration requires non-trivial modifications to our current streetscape infrastructure. Therefore, we will only

consider TCA and SSTCA as part of the effort to identify the optimal data transformation method.

III. METHODS

This section briefly surveys the core methods employed in this paper, including the four data transformation methods, the EMDT-WKNN method, the classification algorithms, the regression algorithm, and our hyperparameter tuning process. We also offer rationales for why the methods were chosen.

A. DATA TRANSFORMATION

Given the divergence in distributions between the RSSI training and test data, using raw measurements to train a model is not a viable strategy. As mentioned in Section II-D, our approach is to transform the raw measurements of both the training and test data such that there is better alignment between their respective distributions. We implement four data transformation methods; details of their implementations are summarized below.

1) BASIC STANDARDIZATION

Basic Standardization (STD) follows the most commonly used formula, Equation (1), to remove the mean and scale the data to unit variance, where x' is the standardized value, x is the raw value, and μ and s are the mean and standard deviation of x, respectively. It is implemented via the `StandardScaler` module of scikit-learn [29].

$$x' = \frac{x - \mu}{s} \tag{1}$$

STD is the first step of all data processing, which means all the other data transformation methods are built on top of the STD-transformed data.

2) KERNEL PCA

Kernel Principal Component Analysis (KPCA) is a standard denoising method that projects data non-linearly into a feature space defined by the kernel function. We choose not to use standard PCA because, for one, RSSI variability is non-linear, and two, given the small dimensionality of our RSSI data, there is little room for further dimensionality reduction. Note that the dimensionality of RSSI data is determined by the number of APs/sensors nearby, which is typically limited in an outdoor environment.

We implement the KPCA-based data transformation method based on the `KernelPCA` module of scikit-learn, using the Radial Basis Function (RBF) as the kernel. We use hyperparameter tuning to find the best values for the following parameters:

- `n_components`: number of dimensions to include in the transformed data
- `gamma`: kernel coefficient for RBF

3) TRANSFER COMPONENT ANALYSIS

As mentioned in Section II-D, Transfer Component Analysis (TCA) aims to find transfer components in the RKHS such that after projection, the difference in training and testing distributions, measured by MMD, is minimized.

Our implementation of TCA follows the algorithm described in [26]. We use hyperparameter tuning to locate the best values for the following parameters:

- `gamma`: coefficient γ in a Laplacian Kernel, defined as

$$k(x, y) = e^{-\gamma||x-y||_1} \tag{2}$$

  where $||x - y||_1$ is the Manhattan distance between the input vectors x and y. The Laplacian Kernel is used for MMD embedding.
- `mu`: a regularization term to control the complexity of the eventual transformation matrix
- `m`: number of dimensions to include in the transformed data

4) SEMI-SUPERVISED TCA

Also discussed in Section II-D is Semi-Supervised TCA (SSTCA), which restricts the search space for the transfer components by preserving the local data geometry (i.e., manifold information) between the training data and the RP locations, such that the new feature space not only minimizes the distribution difference between the training and test data but also optimizes for location prediction. The preservation is realized by maximizing the dependency, computed by the Hilbert-Schmidt Independence Criterion (HSIC), between the training data and their labels as they undergo TCA.

The implementation of SSTCA follows the algorithm introduced in [26]. It requires the K-Nearest Neighbors (KNN) algorithm via the `NearestNeighbors` module in scikit-learn. We use hyperparameter tuning to locate the best values for the following parameters:

- `gamma`: coefficient in a Laplacian Kernel, same as in TCA
- `mu`: a regularization term to control the complexity of the eventual transformation matrix, same as in TCA
- `m`: number of dimensions to include in the transformed data, same as in TCA
- `hsic_gamma`: a trade-off term to balance label dependence and data variance in HSIC
- `n_neigh`: a KNN parameter to control the level of preservation of the data locality property
- `sigma`: a term to control the affinity weight between neighbors or non-neighbors resulting from the K-Nearest Neighbors
- `lambda`: a regularization term for the complexity of the optimization problem in SSTCA

B. EMDT-WKNN

Our implementation of the EMDT-WKNN method mainly follows [6], with two customizations. First, we use the empirical consecutive mean square error method from [25] to select and smooth noise-dominant, high-frequency intrinsic mode functions (IMFs). We do not use the selection criteria in [6] because it often fails to identify even the most

high-frequency IMF in our dataset. Second, we change the equation in step 5 of WKNN in [6] from the original Equation (3) to the modified Equation (4), in which W_i' is the weight of each neighbor's contribution to the final prediction, d_i is the similarity between the RSSI fingerprints of the testing instance and those of a neighbor, and d_{ic}' is the physical distance from a neighbor to their centroid. We suspect the original Equation (3) is incorrect because when d_{ic}' or d_i is very small, i.e., the RSSI fingerprints are very similar between the testing instance and a neighbor, or a neighbor is very close to the centroid, W_i' would be larger than 1, which is incompatible with its role as weight in WKNN.

$$W_i' = \frac{\frac{1}{d_{ic}'} + \frac{1}{d_i}}{\sum_{i=1}^{K'} d_{ic}' + \sum_{i=1}^{K'} d_i} \tag{3}$$

$$W_i' = \frac{\frac{1}{d_{ic}'} + \frac{1}{d_i}}{\sum_{i=1}^{K'} \frac{1}{d_{ic}'} + \sum_{i=1}^{K'} \frac{1}{d_i}} \tag{4}$$

We tune the following hyperparameters to optimize the performance of the EMDT-WKNN method:

- `D`: a distance threshold to filter out the initial nearest neighbors that are too far from their centroid. If, for an RSSI pattern, no nearest neighbor is within `D` distance to the centroid, the RSSI pattern cannot be localized and it will be discarded.
- `n_neighbors`: the number of neighbors to consider in KNN
- `algorithm`: the algorithm used in KNN
- `leaf_size`: the number of leaves in BallTree or KDTree implementation of KNN

C. CLASSIFICATION

For general-purpose classification, we use Support Vector Classification (SVC), supplied by the `SVC` module in scikit-learn. We choose SVC with RBF as the kernel based on our prior experience with its high speed and good performance [3].

The hyperparameters tuned for SVC include:

- `gamma`: the kernel coefficient of RBF; higher γ makes the decision boundary curvier, which could lead to overfitting
- `C`: a regularization parameter; higher `C` imparts less leniency on misclassification, which could lead to overfitting

For classification that requires cost analysis (i.e., balancing the classification accuracy against the cost of discarding data, defined in more detail below), we use the `GaussianProcessClassifier` (GPC) module in scikit-learn with the default settings. We choose GPC for cost analysis because it provides genuine probability estimates for an observation belonging to different classes. Cost analysis in GPC is defined as follows:

Given a classification problem involving two classes, class 0 and class 1, and letting p be the probability of an observation belonging to class 0 and $1-p$ be the probability of it belonging to class 1, we have the following classification rule:

$$\text{predict} = \begin{cases} 0 & \frac{p}{1-p} > c_0 \\ 1 & \frac{p}{1-p} < c_1 \end{cases} \tag{5}$$

c_0 and c_1 are the cutoff values for classifying an observation as class 0 and class 1, respectively. In plain language, we will only classify an observation as class 0 if the probability of it belonging to class 0 is c_0 times higher than that of it belonging to class 1. Likewise, we will only classify an observation as class 1 if the probability of it belonging to class 1 is $\frac{1}{c_1}$ times higher than that of it belonging to class 0. The default values of c_0 and c_1 are both 1, which means there is no bias in classification. However, during cost analysis, we artificially increase c_0 and decrease c_1 to boost the accuracy rate (i.e., any observation that can overcome the threshold is more likely a correct classification) at the cost of discarding ambiguous observations that do not show an apparent affinity to either class (observations with $c_1 \leq \frac{p}{1-p} \leq c_0$). If we make c_0 extremely high and c_1 extremely low, we will obtain high accuracy with a high discard rate. Thus, cost analysis examines the balance between the amount of data one is willing to lose and the level of accuracy one wishes to achieve.

D. REGRESSION

For general-purpose regression, we use the Support Vector Regression (SVR) supplied by the `SVR` module in scikit-learn. We choose SVR with RBF as the kernel for the same reason as SVC, discussed above.

The hyperparameters tuned for SVC include:

- `gamma`: the kernel coefficient of RBF
- `C`: a regularization parameter
- `epsilon`: the ϵ value in the Epsilon-SVR model

E. HYPERPARAMETER TUNING

We conduct most of our hyperparameter tuning using the `GridSearchCV` module in scikit-learn with customized estimators and five-fold cross-validation. Due to the need to tune many hyperparameters (e.g., SSTCA with regression would require the tuning of $7 + 3 = 10$ hyperparameters), it is not realistic to tune all of them at once. Hence, independent hyperparameters are tuned separately. This means, for example, the hyperparameters of SSTCA are tuned with the hyperparameters of SVR unchanged, and vice versa. In practice, no more than three hyperparameters are tuned simultaneously in any model, significantly speeding up the tuning process.

The scoring metrics for classification are recall and false discovery. Let *tp*, *fp*, *tn*, and *fn* be the count of true positive, false positive, true negative, and false negative predictions in a confusion matrix. **Recall** is defined as $\frac{tp}{tp+fn}$ (the rate of positive observations captured by the model), whereas **false discovery** is defined as $\frac{fp}{fp+tp}$ (the rate of

negative observations classified as positive). Recall reflects the accuracy of a classifier, whereas false discovery reveals how often it commits a severe error. The goal of tuning on classification is to find a set of hyperparameters that maximize recall with the lowest false discovery.

The scoring metric for regression is the CDF at 4 m of error. We choose this metric because, empirically, 4 m of error is tolerable for outdoor localization on a city street to monitor pedestrian mobility patterns. Hyperparameter tuning on regression aims to maximize the CDF at 4 m of error.

Finally, to reduce the chance of overfitting, the best set of hyperparameters given by `GridSearchCV` might not be selected if another set exhibits only slightly worse performance (within 2% performance drop), yet has less extreme values (i.e., less likely to cause overfitting). For instance, consider three values for `gamma`: 100, 1, and 0.1, and the corresponding recall results: 0.98, 0.97, and 0.94 after hyperparameter tuning of SVC. Instead of choosing the first `gamma` as the best value (i.e., the one with the highest recall), we choose the second because it is smaller than the first, yet its performance is less than 2% lower. However, we do not select the third `gamma` despite it being even smaller because its performance is more than 2% lower than the second.

The entire hyperparameter tuning process is fully automated based on the scoring metrics and selection criteria.

IV. DATA COLLECTION

Most data collection procedures, including the physical location of the testbed, the software and hardware, the positions of the sensors, the cloud infrastructure to aggregate and store data, and the data pre-processing steps, follow our previous work [3]. Briefly, an empty parking lot with few obstructions on the Boca Raton campus of Florida Atlantic University is used as the testbed. Four sensors, labeled S1, S2, S3, and S4, are mounted on top of tripods 1.8 m from the ground. They are placed at the corners of a 16 m $\times$ 16 m square, with S3 designated as the origin $(0, 0)$ (see Fig. 4(A)). During data collection, an emitter mounted on a tripod 1 m high and capable of sending mock probe requests at a pre-determined frequency is placed on various RPs for 20 seconds each, during which time approximately 1,000 mock probe requests are sent. Each sensor captures approximately 70% of the mock probe requests, extracts their RSSI values, (mock) MAC addresses, and timestamps, and uploads the data to a database. The raw RSSI data collected after each session constitute a matrix of shape $N \times 4$, where N is the number of probe requests captured, and the four columns correspond to the RSSI measurements from the sensors, in the order of S1, S2, S3, and S4. Each row represents the RSSI pattern (which may contain missing values) of a single probe request emitted at an RP.

The raw data from each data collection session is examined for outliers. For all the rows under the same RP, we use the 3σ rule to identify outliers in each RSSI column (i.e., a variable is deemed an outlier if it deviates from the mean by more than three times the standard deviation). After all outliers are removed, we have the *cleaned raw data* with shape $N^* \times 4$, where N^* is the number of rows remaining after outlier exclusion. We compute the mean RSSI of each column within each RP to produce the *mean data*. The mean data is a matrix of shape $M \times 4$, where M is the number of RPs, and the four columns are the mean RSSI values corresponding to the four sensors (Fig. 3(A)). Each row now represents the mean RSSI pattern of an RP.

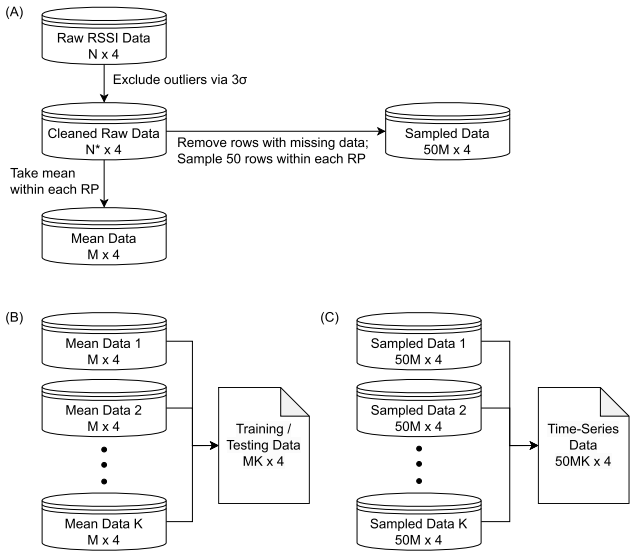


FIGURE 3. Relationship among datasets and their shape.

The *training data* for the models of our previous paper and the *InXModel* are obtained by concatenating and shuffling the mean data of multiple sessions. It is a matrix of shape $MK \times 4$, where K is the number of data collection sessions used for training purposes (Fig. 3(B)).

However, the *time-series data* for the EMDT-WKNN method is acquired differently because instead of taking the average, we must line up individual rows of RSSI measurements in chronological order. For each session, we eliminate from the clean raw data any rows with missing measurements (i.e., a sensor that fails to capture a mock probe request) and randomly sample 50 rows from each RP to form the *sampled data*, which has shape $50M \times 4$ (Fig. 3(A)). The time-series data is formed by concatenating the sampled data of all training sessions in chronological order and has shape $50MK \times 4$, where the four columns are the raw RSSI values corresponding to the four sensors (Fig. 3(C)). Each row of the time-series data is an RSSI pattern (with no missing values) of a single probe request emitted from an RP.

The *test data* are acquired similarly as the training data, but from different data collection sessions designated for testing purposes.

As mentioned in Section I, a key component of our approach to addressing RSSI fluctuation is reformulating the problem of pedestrian localization. To do so, we establish a new data collection layout to determine how RPs and test

points (TPs) are positioned in the testbed. In the following two subsections, we define the old (broad data layout) and the new (split data layout) layouts used in this paper.

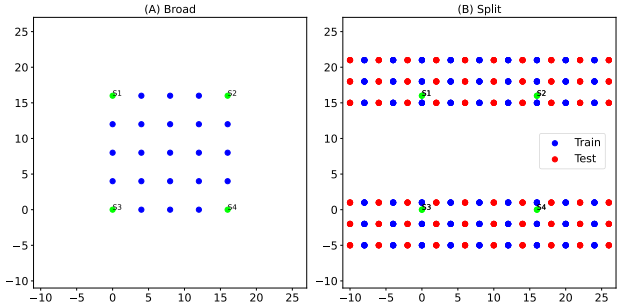


FIGURE 4. Data collection layout.

A. BROAD DATA LAYOUT

Fig. 4(A) shows the broad data layout. The green dots represent the sensors, and the blue dots represent RPs where mock probe requests are emitted. S3 is designated as the origin of the coordinate system. The broad layout is a descendant of the layout used in our previous work, fully covering the area bordered by the sensors. However, to facilitate scalability, we reduce the RP density by increasing the gap between adjacent RPs from 1 m in [3] to 4 m. There is no distinction in the broad layout between TPs and RPs, i.e., test data are collected at the exact physical locations as training data.

The training data under the broad data layout were collected on five separate days in March and April 2021, and the test data were collected in a single session in July 2021.

B. SPLIT DATA LAYOUT

Fig. 4(B) shows the split data layout, with the green, blue, and red dots representing the sensors, RPs, and TPs, respectively. S3 is still designated as the origin of the coordinate system. While the positioning of the sensors and the gap between RPs remain the same, we make three significant changes.

1) The split layout does not cover the center of the testbed.
2) The split layout extends horizontally beyond the sensor boundary.
3) The TPs do not overlap with the RPs.

These changes aim to address two major drawbacks of the broad data layout.

First, the distance between adjacent RPs in the broad data layout (Fig. 4(A)), despite being 4 m, may still be too small to unequivocally distinguish the RSSI patterns from one RP to its neighbors. The similarity between RP neighbors' RSSI patterns means that RSSI data collected at a TP might initially match one RP but later match its neighbor as time passes.

Second, the broad data layout does not resemble how pedestrians move about in a real-world streetscape. On Clematis Street, where our production environment is deployed (Fig. 5), pedestrians almost always walk on the sidewalk to the North and South sides of the street, as the center space is reserved for vehicles. Thus, we only need the localization model to function on the sidewalk rather than the entire area to monitor pedestrian mobility patterns. The broad data layout's full coverage of the testbed is unnecessary. Furthermore, since pedestrians can freely enter or leave any area, their movement is not restricted within the sensor boundaries. This is overlooked in the broad data layout, as no RP exists outside the sensor boundaries.

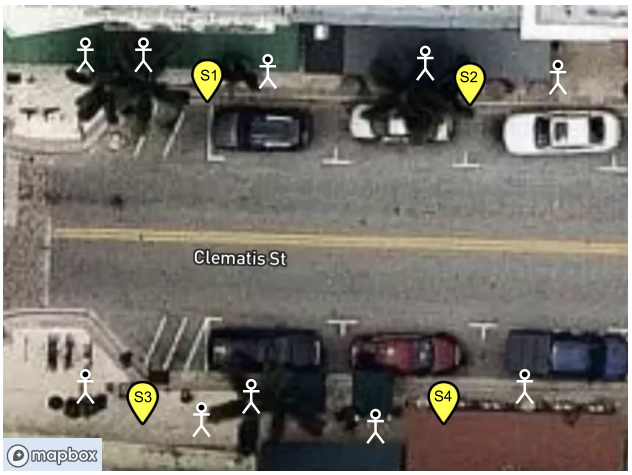


FIGURE 5. Pedestrian (stick figures) movement on Clematis Street with deployed sensors (yellow markers).

The first and second significant changes in the split data layout directly resolve the second drawback, as it resembles how pedestrians move (i.e., all RPs are located in the testbed's North and South strips). The RPs also extend horizontally beyond the sensor boundaries, the same way pedestrians enter and leave the sensor-bounded regions.

With the empty space in the center of the testbed, the split data layout partially addresses the first drawback (i.e., the distance between adjacent RPs is too small). Although the RPs within the North and South strip still suffer from the same RSSI similarity issue, the spatial gap in the middle means that it is much less likely for an RSSI pattern from the North to be mixed with that from the South. We will capitalize on this enhanced ability to distinguish the Northern from the Southern data in Section VII-D.

The third significant change in the split data layout provides the final advantage: the separation of the RPs from the TPs. This allows for a more rigorous, realistic examination of the performance of the localization models.

Given the many benefits of the split data layout, all the training and test data used in the paper, excluding those specific to the prior models, were collected under the new layout. The training data were collected across five days in January and February 2022, and the test data were collected across two days in March and April 2022, including a morning and an afternoon session per day.

V. RESCUING THE PRIOR MODELS

The impact of RSSI fluctuations on localization performance is demonstrated in Fig. 2, as discussed previously. A straightforward rescue attempt is to expand the training data to include RSSI data collected on different days. Fig. 6 shows the results of such an approach applied to the *fp*, *ml-clf*, and *ml-regr* models. The layout of the figure is the same as described before. Curves *fp-sgl*, *ml-clf-sgl*, and *ml-regr-sgl* represent the performance of the prior models trained on the same single-day data. Curves *fp-mul*, *ml-clf-mul*, and *ml-regr-mul* represent the performance of the prior models trained on the multi-day data. All the models are tested in the same way so that their performance can be compared directly. The results show that, across all models, the performance of the model trained on the multi-day data is better than the performance when trained on the single-day data, validating the assumption that by expanding the data variation exposed to the model, its robustness against RSSI temporal fluctuation improves.

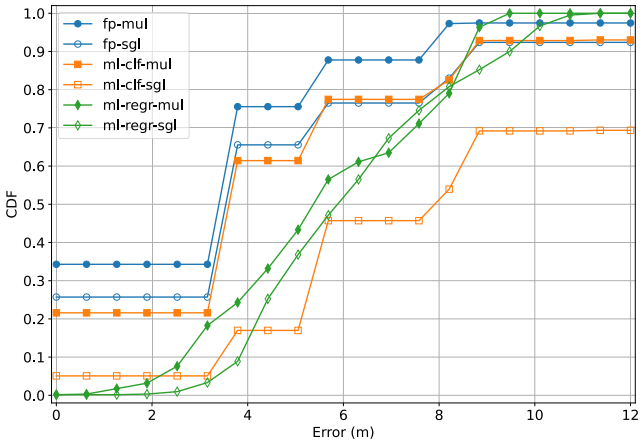


FIGURE 6. Localization performance comparison of the prior models: multi-day vs. single-day.

However, even with the multi-day training data, none of the prior models achieves the desired performance of ≤ 4 m of error in 90% of the cases. Therefore, the straightforward rescue is not good enough.

Another rescue attempt incorporates the three data transformation methods discussed in Section II-D (STD is not included because it has already been applied as part of data pre-processing in the prior models). The results are shown in Fig. 7(A), (B), and (C) for *fp*, *ml-clf*, and *ml-regr*, respectively. The figure layout is the same as described before. The blue curve represents the performance of the original model (i.e., data transformed by STD), whereas the orange, green, and red curves correspond to the performance after data transformation by KPCA, TCA, and SSTCA, respectively. Across all three prior models, some data transformation methods boost the localization performance. For instance, TCA and SSTCA boost *fp* performance, and TCA and KPCA improve *ml-clf* and *ml-regr* performance. This highlights the

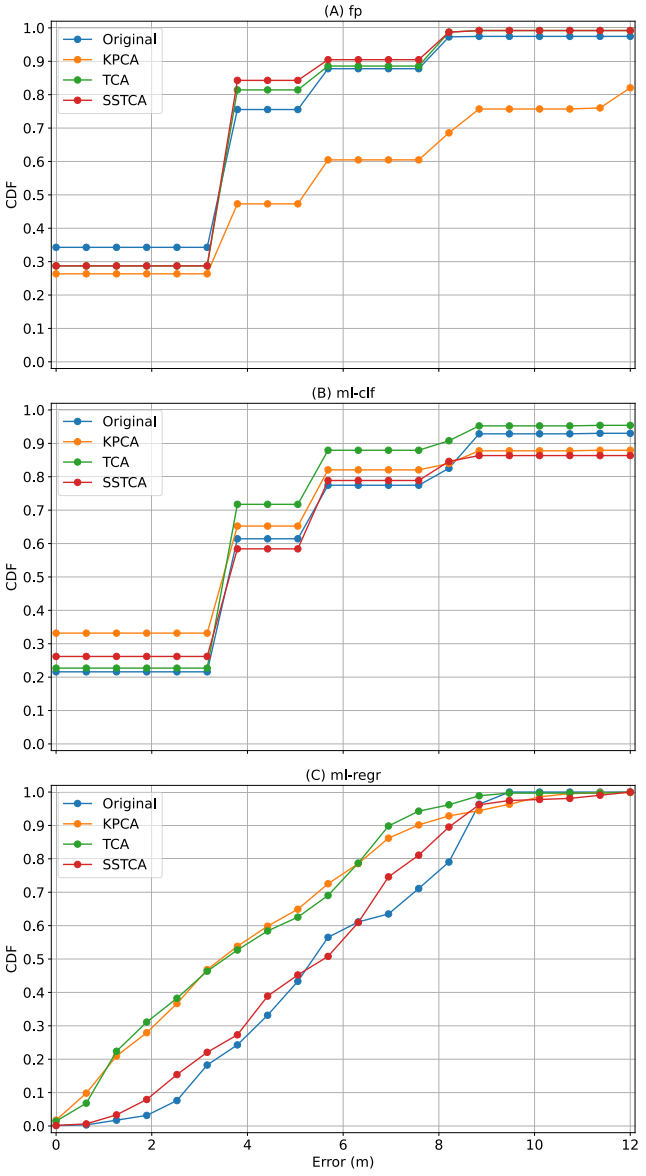


FIGURE 7. Localization performance comparison of the prior models with various data transformation methods.

usefulness of the data transformation methods, but they again fall short of the target of 4 m or less error in 90% of the cases.

Since neither the expansion of the training data nor the data transformation methods can propel the prior models to the desired performance, it is doubtful whether more performance can be squeezed from them. Significant changes are needed to improve localization performance further under RSSI fluctuations. In the next section, we discuss the drawbacks of the data collection layout used in the prior models and explore a new layout that is more resistant to RSSI variation.

VI. CONCEPTUALIZING A NEW MODEL

With the localization problem reformulated and a new data collection layout established, we made a final attempt at

rescuing the prior models by examining their performance under the split data layout. The results are shown in Fig. 8, with the same layout and labeling as described before. Since none of the prior models can achieve the desired localization performance, we are convinced that a new, more robust model is needed.

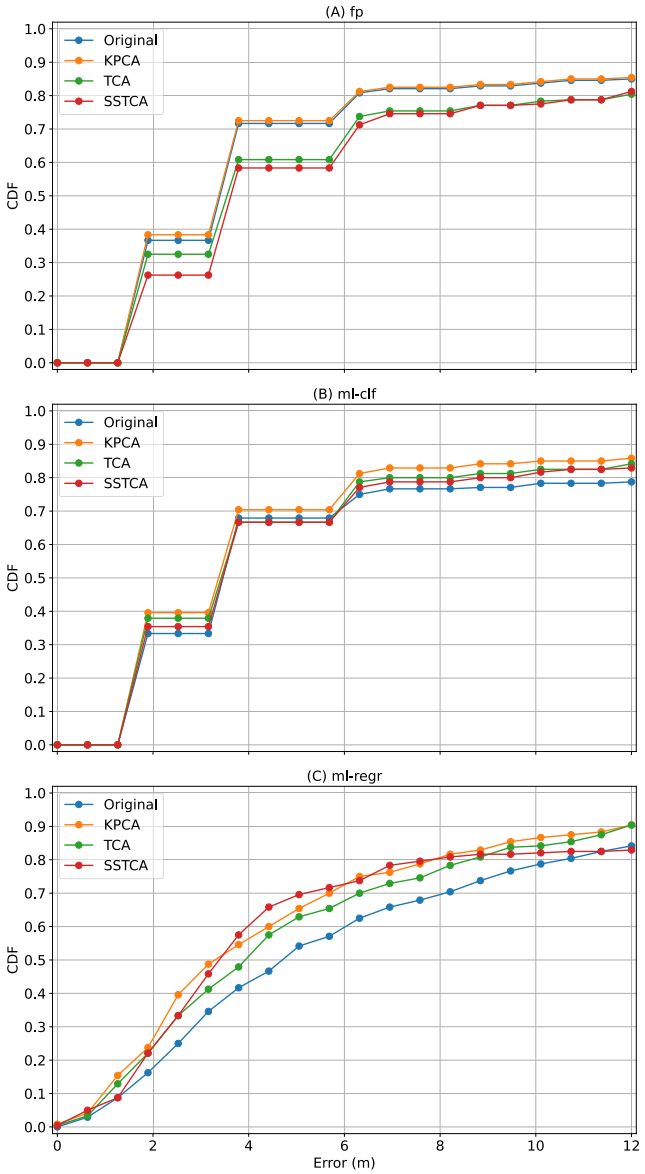


FIGURE 8. Localization performance comparison of the prior models on the split data layout with various data transformation methods.

We conceptualize the new model based on qualitative reasoning and quantitative evidence in the following two subsections.

A. QUALITATIVE REASONING

For an RSSI-based localization model to perform well, it must capture a time-invariant, location-sensitive pattern in the RSSI patterns. The lackluster performance of the prior models on the split data layout (Fig. 8) suggests that if such a pattern does exist, it most likely manifests itself over a subset of the training and test data.

To qualitatively determine the existence of a time-invariant, location-sensitive pattern, we plot the mean RSSI value of each sensor at each RP as shown in Fig. 9. The green dots represent the location of the four sensors, and the positions of the inset bar plots represent the locations of each RP. Within each inset bar plot, the blue, orange, green, and red bars represent the mean RSSI values recorded by S1, S2, S3, and S4, respectively. Essentially, each inset bar plot illustrates the mean RSSI pattern of the RP. From these RSSI patterns, we can make four observations.

1) Within the North or South strip, there is little RSSI variation in the North-South direction (see blue rectangles). Such homogeneity indicates that a model to predict the y-coordinate within the North or South strip would be difficult to train. If such a model were produced, it would yield poor accuracy in predicting the distance between a pedestrian and the front of a shop. Fortunately, this is not a significant issue. Understanding pedestrians' North-South positions within a sidewalk does not add much value to understanding their mobility patterns. On the contrary, we can take advantage of this observation by assuming that the y-coordinates of RPs and TPs within the North or South strip are the same. Consequently, instead of predicting two coordinates, we only need to predict the x-coordinate within the North or South strip.
2) Within the North or South strip, there is a noticeable, location-sensitive pattern in the East-West direction among RSSI measurements associated with the sensors on the same side, as one would expect. For instance, among the Southern RSSI patterns in the red triangle, mean RSSI values obtained by S3 and S4 increase as the RP approaches these sensors and decrease as it moves away. The same pattern can be observed in the rows in the North, relative to S1 and S2. This suggests that predicting only the x-coordinate in the North or South strip may be feasible.
3) The location-sensitive RSSI pattern mirrors itself on either side of each sensor. This means the same RSSI pattern could be associated with two RPs, one inside and the other outside the sensor boundary (see an example in the fuchsia rectangles). To avoid such confusion, it would benefit a model if one of the options, such as the one outside the sensor boundary, were not included in the training and test data.
4) Despite a large gap between the North and South RPs, there is still RSSI overlap, as demonstrated by the RSSI patterns enclosed in the grey rectangle, where the distinction between the North and South RSSI patterns is not clear. This is unsurprising because as the RPs move away from the sensors, the RSSI measurements are more noise-dominant. It would benefit a model

if these noise-dominant RSSI patterns were excluded from the training and test data.

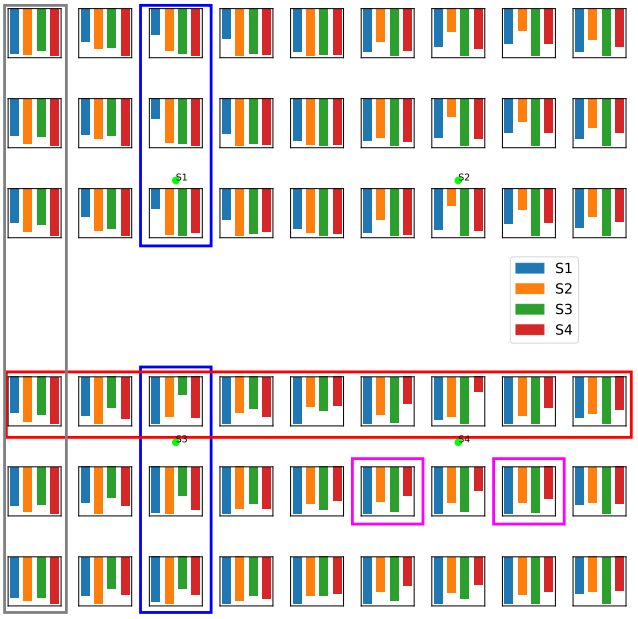


FIGURE 9. Mean RSSI patterns of each RP in the training data.

From these observations, we can sketch an outline of the new model: It eliminates data collected from RPs outside the East-West sensor boundary, separates the remaining data to North and South, and then (only) predicts x-coordinates. It is important to note that the eliminated data are not truly lost because sensors are deployed at regular intervals along the entire street in our urban production environment, which means the neighboring sensor quadruplets will pick up any data excluded by the current sensor quadruplets. The only exception is at the very ends of the street, but data loss at those locations is acceptable because the lost data are also outside the street's physical boundary.

Nevertheless, quantitative evidence is needed to validate this qualitative sketch. We discuss the quantitative evidence next.

B. QUANTITATIVE EVIDENCE

The qualitative sketch of the new model essentially restricts the training and test data to only those collected within the East-West sensor boundaries and predicts only x-coordinates. To validate this sketch, we compare the localization performance of the *ml-regr* model (one of the models in our prior work [3]) as the data are increasingly restricted. Since this is a proof-of-concept argument, we presume (pretend) that the locations of all RPs and TPs are already available. We expect the performance to peak at the restriction level indicated by the qualitative sketch.

Fig. 10 summarizes the localization performance. The training and test data are derived from the split data layout. The top plot includes the CDF curves of localization error, where the blue, orange, green, and red curves correspond to distinct data restriction scenarios *full-x-y*, *ns-sep-x-y*, *ns-sep-x*, and *ns-sep-x-in*, respectively. The definition of each scenario is given below.

- **full-x-y**: All data are included. Two models are trained, one for the x- and the other for the y-coordinate.
- **ns-sep-x-y**: All data are included, but the North and South data are separated. Within each half, two models are trained, one for the x- and the other for the y-coordinate.
- **ns-sep-x**: All data are included, but the North and South data are separated. Within each half, one model is trained for the x-coordinate only. During the evaluation, localization error is only computed for the x-coordinate.
- **ns-sep-x-in**: Only the data inside the East-West sensor boundaries are considered. These data are further separated into North and South. Within each half, one model is trained for the x-coordinate only. During the evaluation, localization error is only computed for the x-coordinate.

The four smaller plots at the bottom of the figure show the distribution of the RPs and illustrate the meaning of the four restriction scenarios.

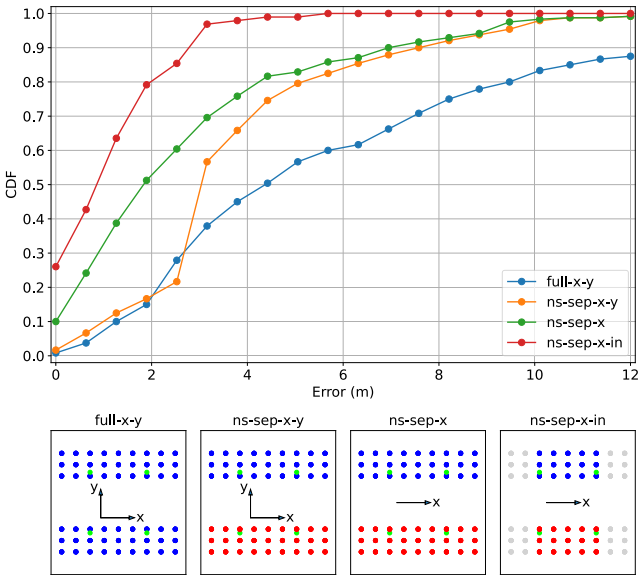


FIGURE 10. Localization performance of the *ml-regr* model as data are increasingly restricted.

It is apparent that the localization performance improves as data are separated (*full-x-y* vs. *ns-sep-x-y*), the y-coordinate is ignored (*ns-sep-x-y* vs. *ns-sep-x*), and data outside the East-West sensor boundaries are excluded (*ns-sep-x* vs. *ns-sep-x-in*). The best performance, *ns-sep-x-in*, corresponds precisely to the qualitative sketch of the new model, and it reaches ≤ 4 m of error in 90% of the cases.

Granted, this performance will not be replicated in real life because *ns-sep-x-in* is gifted with perfect a priori knowledge of the TP locations, which guarantees that the data restrictions are perfect. In reality, the data restrictions will not be perfect,

as the locations of the test data are unavailable—they are what we want to predict in the first place. If some test data outside the sensor boundary fail to be excluded, they might produce large prediction errors. Nevertheless, the performance of *ns-sep-x-in* shows that if we can successfully identify RSSI patterns, both in training and testing, that originate from RPs and TPs within the East-West sensor boundaries and split them in the North-South direction, the remaining x-coordinate regression is likely to yield good performance. Essentially, we have reformulated the original localization problem into two more manageable sub-problems: (1) identifying the data within the East-West sensor boundaries in the North or South strip and (2) predicting their x-coordinates. We call the first sub-problem **inlier classification** and the second **inlier regression**.

To address these two sub-problems, we create a new multi-stage Inlier X-coordinate Model (*InXModel*), as shown in Fig. 11. In the following sections, we detail the steps depicted in the *InXModel* flow chart.

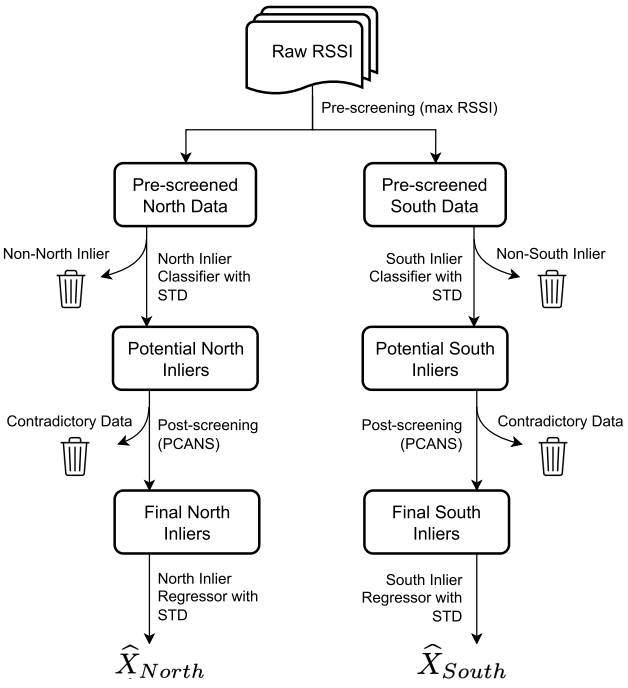


FIGURE 11. Flow chart of the *InXModel*.

VII. INLIER CLASSIFICATION

This section first defines the meaning of inlier, outlier, and two error types in inlier classification. We then show the result of naive inlier classification, explore methods to boost its performance, and demonstrate the final performance of the improved inlier classifier.

A. INLIER AND ERROR DEFINITIONS

Inliers refer to the RSSI data originating from the RPs within the East-West sensor boundaries. In Fig. 12, any RSSI data collected from the green blocks are considered inliers. Similarly, any RSSI data originating from the red or grey blocks are considered outliers. Notice that the North inliers lie within the boundary of S1 and S2, whereas the South inliers lie between S3 and S4.

Suppose the blue dots represent the predicted North inliers, and the red dots represent the predicted South inliers. If the blue and red dots land in the green blocks, they are considered correct predictions. Otherwise, they are errors.

There are two types of errors in inlier classification. The first type is an error on the same side (**err-same**), which refers to a situation where a predicted inlier is actually an outlier but resides on the same side of the prediction. In Fig. 12, if the blue and red dots land in the red blocks, they are considered err-same. The second type is an error on the opposite side (**err-opp**), which refers to a situation where a predicted inlier is actually an outlier and resides on the opposite side of the prediction. In Fig. 12, if the blue and red dots fall anywhere in the grey blocks, they are considered err-opp.

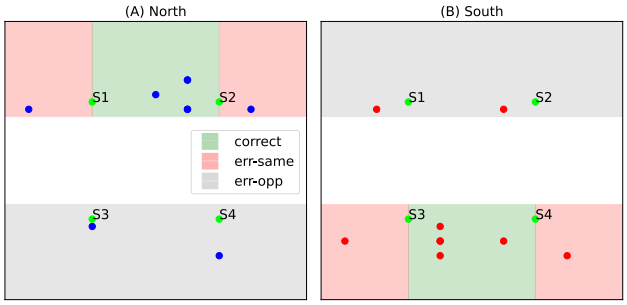


FIGURE 12. Inliers, outliers, and error definitions in inlier classification.

Err-opp is more significant than err-same because it always leads to substantial localization errors. In a real-world application, err-opp misplaces a pedestrian from the North sidewalk to the South or vice versa, significantly damaging the observed mobility pattern. Err-same is not as serious, especially if the actual position of the misclassified inlier is near the red-green border. Therefore, we invest much effort to reduce err-opp, even at the cost of discarding data (discussed in detail in Section VII).

B. NAIVE INLIER CLASSIFICATION

We create two models for naive inlier classification, one for the North and the other for the South. The input for both inlier classifiers is the entirety of the training data, modified by one of the four data transformation methods presented in Section III-A. The training label is a binary value, where 1 indicates that the training data are inliers and 0 indicates outliers. The output is also a binary value, where 1 means that a testing RSSI pattern is predicted to be an inlier, and 0 means it is predicted as an outlier. We use SVC, described in Section III-C, as the underlying classifier and follow the hyperparameter tuning process described in Section III-E.

The results of the North and South naive inlier classification across the four data transformation methods are shown

in Fig. 13(A) and (B). Detailed metrics are available in Table 1. The blue, orange, green, and red bars represent the performance of STD, KPCA, TCA, and SSTCA, respectively. Recall is the percentage of the actual inliers recovered by the inlier classification (see Section III-E for details). For instance, the recall of naive North inlier classification with the STD method is $40/48 = 0.83$. Err-same and err-opp rates are the error rates computed as the count of err-same and err-opp instances over the total count of predicted inliers. For instance, the err-opp rate of naive South inlier classification with the TCA method is $4/(39 + 17 + 4) = 0.067$.

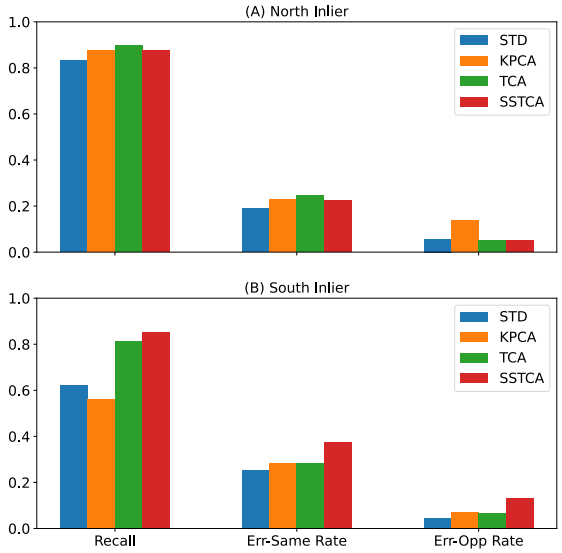


FIGURE 13. Metrics of naive inlier classification.

Ideally, we want recall to be high, err-same rate to be low, and err-opp rate to be zero. However, low recall is not by itself a fundamental deal-breaker. It means that only a small percentage of the actual inliers are captured by the classifier. Given that we capture millions of Wi-Fi probe requests in the production environment, we can afford low recall but still obtain sufficient inlier data to analyze pedestrians' mobility patterns. A moderate err-same rate is also acceptable, as explained in Section VII-A. Thus, err-opp rate is the only metric we must keep as low as possible. The err-opp rate represented in Fig. 13 is unacceptably high; further measures are needed to reduce it.

C. INLIER CLASSIFICATION WITH PRE-SCREENING

We implement a basic pre-screening process on the test data: We classify a testing RSSI pattern as North if the maximum RSSI value belongs to S1 or S2, or South if the maximum RSSI value belongs to S3 or S4. For example, if an RSSI pattern is $(-40, -30, -50, -60)$, it will be pre-screened to the North because the maximum RSSI, -30, belongs to S2. However, if an RSSI pattern is $(-40, -50, -20, -60)$, it will be pre-screened to the South due to S3 having the maximum RSSI value of -20. The rationale behind pre-screening is that the RSSI measurements by the sensors on the same side as the

TABLE 1. Results of naive inlier classification.

	Total Inliers	North 48	South 48
STD	Correct	40	30
	Err-Same	10	11
	Err-Opp	3	2
	Recall	0.83	0.62
	Err-Same Rate	0.19	0.26
	Err-Opp Rate	0.06	0.05
KPCA	Correct	42	27
	Err-Same	15	12
	Err-Opp	9	3
	Recall	0.88	0.56
	Err-Same Rate	0.23	0.29
	Err-Opp Rate	0.14	0.07
TCA	Correct	43	39
	Err-Same	15	17
	Err-Opp	3	4
	Recall	0.90	0.81
	Err-Same Rate	0.25	0.28
	Err-Opp Rate	0.05	0.07
SSTCA	Correct	42	41
	Err-Same	13	31
	Err-Opp	3	11
	Recall	0.88	0.85
	Err-Same Rate	0.22	0.37
	Err-Opp Rate	0.05	0.13

RP are generally larger than those measured by the sensors on the opposite side. Thus, most pre-screened North (South) data are truly from the North (South). Since there are fewer data points from the opposite side, to begin with, and if we only feed the pre-screened North (South) data to the North (South) inlier classifier, the chance of committing err-opp would decrease.

It is tempting to also apply pre-screening to the training data such that the North (South) inlier classifier is trained only on the pre-screened North (South) training data. Unfortunately, this would drastically diminish the power of the inlier classifiers because the training data size would be halved after pre-screening. Therefore, to preserve classification capacity, both the North and South inlier classifiers are trained on the entirety of the training data as before. Only during evaluation do we use pre-screening and supply the pre-screened North (South) test data to the North (South) inlier classifier. In a sense, one can view the North (South) inlier classifier as a "double-checker" of the pre-screening process.

The performance of inlier classification with pre-screening is summarized in Fig. 14 and Table 2. The layout and labeling of the figure and table are the same as described before.

Although some sacrifices occur in recall (e.g., the correct instances of STD South, KPCA South, and TCA South all drop compared to naive inlier classification), we have drastically reduced the err-opp rate. In particular, there are no instances of err-opp among all South inlier classifiers

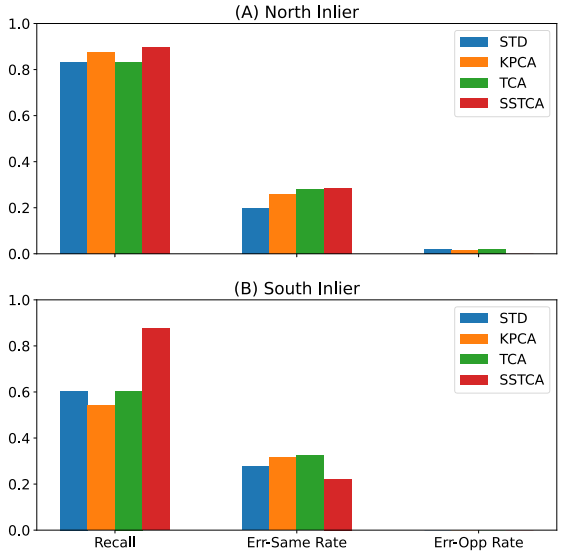


FIGURE 14. Metrics of inlier classification with pre-screening.

TABLE 2. Results of inlier classification with pre-screening.

		North	South
	Total Inliers	48	48
STD	Correct	40	29
	Err-Same	10	11
	Err-Opp	1	0
	Recall	0.83	0.60
	Err-Same Rate	0.20	0.28
	Err-Opp Rate	0.02	0.00
KPCA	Correct	42	26
	Err-Same	15	12
	Err-Opp	1	0
	Recall	0.88	0.54
	Err-Same Rate	0.26	0.32
	Err-Opp Rate	0.02	0.00
TCA	Correct	40	29
	Err-Same	16	14
	Err-Opp	1	0
	Recall	0.83	0.60
	Err-Same Rate	0.28	0.33
	Err-Opp Rate	0.02	0.00
SSTCA	Correct	43	42
	Err-Same	17	12
	Err-Opp	0	0
	Recall	0.90	0.88
	Err-Same Rate	0.28	0.22
	Err-Opp Rate	0.00	0.00

and only one each for the North inlier classifiers with STD, KPCA, and TCA.

The success of pre-screening begs the question: Can we do even better to remove the remaining err-opp?

D. INLIER CLASSIFICATION WITH POST-SCREENING

Err-opp occurs when a North (South) data point is classified as a South (North) inlier. A post-screening process may reduce this error by conducting a North-South classification on the initially predicted inliers and flagging those that receive contradictory predictions from the two classifications. In other words, if some data are initially classified as North (South) inliers by the North (South) inlier classifier but later classified as non-North (non-South) by a post-screening North-South classifier, the contradiction means that these data are dubious and should be discarded. The expectation is that err-opp will reduce even further after removing the dubious data.

The feasibility of post-screening depends on a classifier that can reliably distinguish North and South inliers. Such a classifier is possible because the inliers in the training data can be readily separated into North and South once they are projected to the first two principal components (Fig. 15). By combining PCA with a Gaussian Process Classifier (GPC), we train a PCA-based North-South classifier (PCANS) on inliers. Section III-C explains the reason for choosing GPC as the underlying classification algorithm and explains the cost analysis procedure (i.e., balancing the classification accuracy against the cost of discarding data).

Regarding PCANS, we define c_0 as the cutoff for classifying an RSSI pattern as North and c_1 as the cutoff for classifying an RSSI pattern as South (refer to Section III-C for the definition of c_0 and c_1). c_0 and c_1 can be tuned like hyperparameters based on the allowance of discard rate (any RSSI pattern that cannot cross either cutoff is discarded). In the following analysis, the discard rate is set at 5%. We will discuss the discard rate further in Section VII-E.

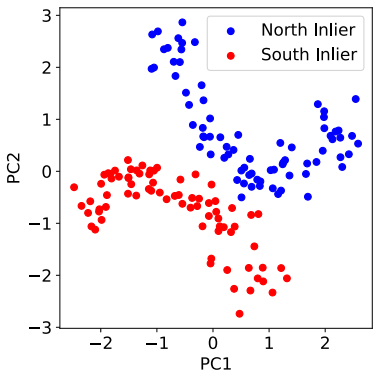


FIGURE 15. Separation of North and South inliers in the training data along principal components 1 and 2.

We pass the output of inlier classification with pre-screening (both the North and South versions) to a tuned PCANS. Any RSSI pattern resulting in contradictory predictions between the inlier classifier and PCANS is discarded. The results of inlier classification with post-screening are summarized in Fig. 16 and Table 3. The layout and labeling of the figure and table are the same as described before.

With a few more sacrifices in recall, we have eliminated all instances of err-opp. Comparing the classification performance across the four data transformation methods, we observe that SSTCA performs best in the South but has the

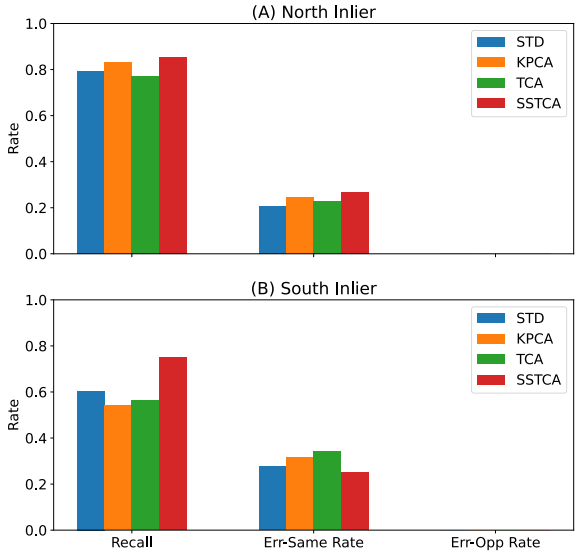


FIGURE 16. Metrics of inlier classification with post-screen.

TABLE 3. Results of inlier classification with pre-screening and post-screening.

		North	South
	Total Inliers	48	48
STD	Correct	38	29
	Err-Same	10	11
	Err-Opp	0	0
	Recall	0.79	0.60
	Err-Same Rate	0.21	0.28
	Err-Opp Rate	0.00	0.00
KPCA	Correct	40	26
	Err-Same	13	12
	Err-Opp	0	0
	Recall	0.83	0.54
	Err-Same Rate	0.25	0.32
	Err-Opp Rate	0.00	0.00
TCA	Correct	37	27
	Err-Same	11	14
	Err-Opp	0	0
	Recall	0.77	0.56
	Err-Same Rate	0.23	0.34
	Err-Opp Rate	0.00	0.00
SSTCA	Correct	41	36
	Err-Same	15	12
	Err-Opp	0	0
	Recall	0.85	0.75
	Err-Same Rate	0.27	0.25
	Err-Opp Rate	0.00	0.00

highest err-same in the North. STD accumulates the lowest err-same and has a decent recall in the North, yet its recall in the South is lackluster. Therefore, picking a clear winner at the current stage is difficult. We will make the decision after the overall localization performance under each data transformation method is revealed.

To further investigate where the predicted inliers and errors come from, Fig. 17 shows the distribution of the predicted inliers before (blue bars) and after PCANS (orange bars). The x-axis represents the x-coordinates of the testbed, and the y-axis represents the count of inliers, with the top half for the North inliers and the bottom half for the South inliers. The total number of inliers at each x-coordinate is 12 because the test data consists of four sessions with three RPs at each x-coordinate. The bars inside the green, red, and grey blocks indicate correct classification, err-same, and err-opp, respectively.

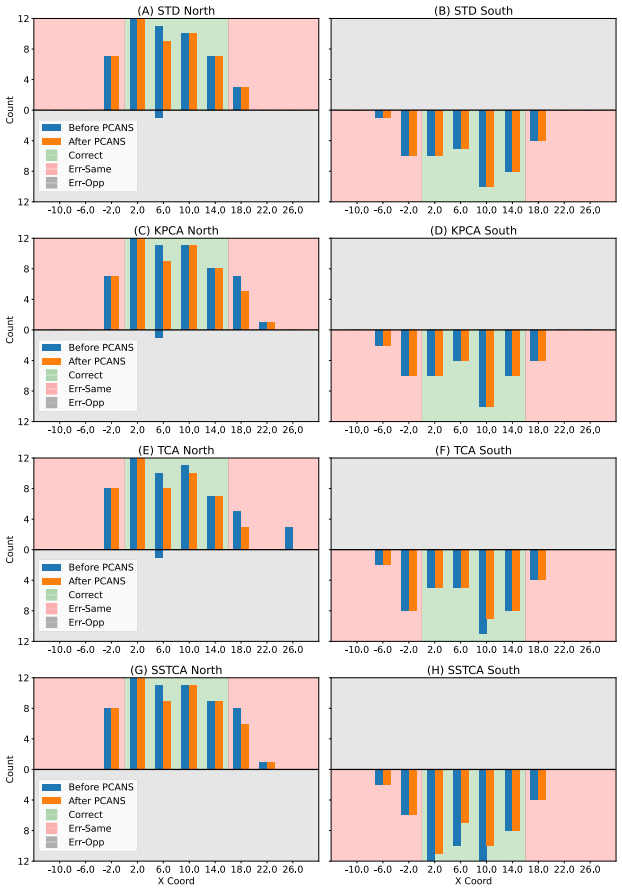


FIGURE 17. Distribution of predicted inliers before and after PCANS.

Fig. 17(A), (C), and (E) show the effect of PCANS on removing err-opp at $x = 6.0$. Fig. 17(C), (E), and (G) demonstrate the correction of err-same in the red blocks on the right. Sacrifices in the recall are exhibited in the green blocks of all the panels, except Fig. 17(B) and (D).

Despite being tolerated, the unusually low recall in STD, KPCA, and TCA in the South inlier classification deserves attention. Fig. 17(B), (D), and (F) reveal that the loss in the recall is mainly attributed to low inlier capture at $x = 2.0$ and $x = 6.0$ in the South. This suggests that the RSSI patterns of the South test data at these positions might not match the RSSI patterns of the South training data at the nearby positions of $x = 0.0$, $x = 4.0$, and $x = 8.0$.

Fig. 18 shows the mean RSSI pattern of all RPs with the same x-coordinates in the South, with panels (A), (B), and (C) corresponding to the training data at $x = 0.0$, $x = 4.0$, and $x = 8.0$, and panels (D) and (E) corresponding to the test data at $x = 2.0$ and $x = 6.0$. Each bar plot's x-axis represents the four sensors, and the y-axis represents the mean RSSI values. Comparing the test data in panel (D) with the training data in panels (A) and (B), we observe that weaker RSSI measurements of S2 and S3 might contribute to the low recall at $x = 2.0$. Similarly, comparing test data in panel (E) with training data in panels (B) and (C), we see that weaker RSSI measurements of S2 and S3 might again be the culprit of the low recall at $x = 6.0$. We speculate that local interference or slight changes in sensor setup (e.g., height and orientation) at each data collection session might cause weaker RSSI measurements at S2 and S3. However, these problems might not be as big a concern in the production environment. For local interference, its negative effect can be mitigated by incorporating data from additional sensors along the street. For sensor setup, there should be no change in its orientation and height once deployed on a light pole. That said, it remains to be seen whether a similar deficiency in the recall will manifest with data collected in the production environment.

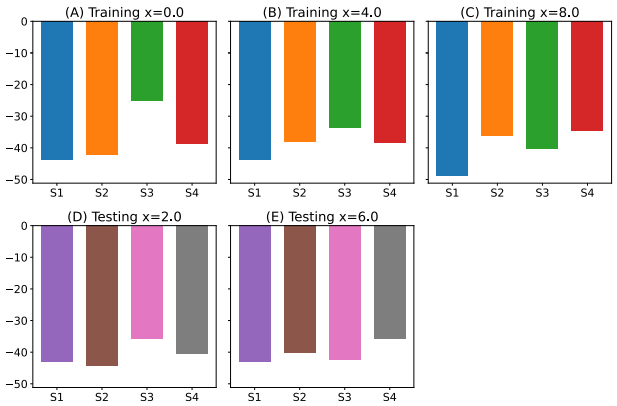


FIGURE 18. Mean RSSI patterns of x-coordinates in the South where recall is low.

E. MECHANISM OF PCANS

Fig. 19 shows how PCANS reduces err-opp at the cost of a reduced recall. The four panels correspond to the four data transformation methods. Within each panel, the x- and y-axis represent the first and second principal components on which data are projected. The blue (red) contours are computed likelihoods from the training data (the darker the color gradient, the higher the likelihood), such that any data point falling inside the contour would be predicted by PCANS as North (South), and any data point landing outside the contour would be considered dubious and discarded. The blue (red) dots are the predicted North (South) inliers by the North (South) inlier classification with pre-screening. Any blue (red) dots circled in red (blue) indicate that it is actually from the South (North) or an err-opp.

PCANS can eliminate instances of err-opp if the data points to land in the blank space (i.e., dubious) or the wrong contour (i.e., contradiction). Fig. 19(A), (B), and (C) demonstrate the first case, where a blue dot circled in red lands in the blank space between the contours. Since this data point is deemed not convincingly North or South by PCANS, it is discarded. This is how PCANS removes the single instance of err-opp in North STD, KPCA, and TCA (compare Table 2 with Table 3).

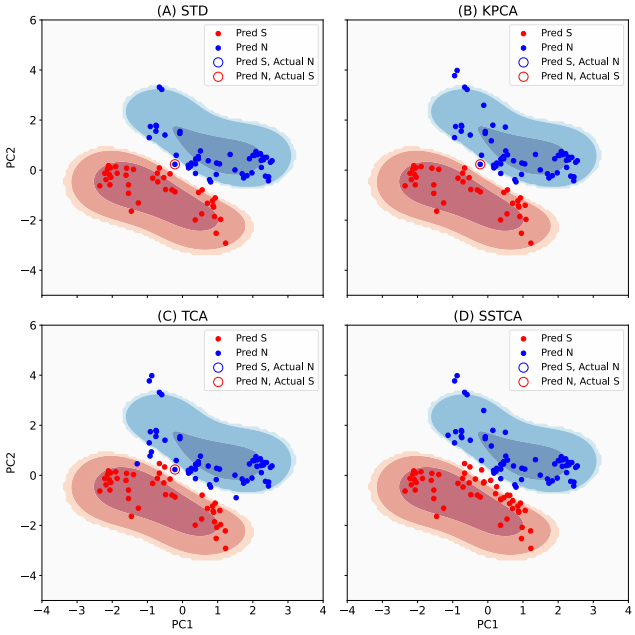


FIGURE 19. Mechanism of how PCANS reduces err-opp at the cost of lowering recall.

The sacrifice in recall is also on display in all four panels. Any time a non-circled blue (red) dot falls in the blank space or wrong contour, it is eliminated by PCANS, equivalent to discarding a correctly predicted inlier. For instance, Fig. 19(D) shows that PCANS removes many red and a few blue dots in the blank space between the contours. This corresponds to the drop in correct classifications from 42 to 36 in the South and 43 to 41 in the North for SSTCA (compare Table 2 with Table 3).

In Section VII-D, we mentioned that PCANS is tuned with a discard rate of 5% for post-screening. Discard rate plays a vital role in controlling the aggressiveness of data elimination. Increasing the discard rate shrinks the blue and red contours, which exposes more blank space and makes it easier to discard data points. Since most err-opp instances are dubious data points landing somewhere between the contours, a high discard rate helps eliminate them. However, the recall will also suffer from smaller contours. Ideally, we want to balance err-opp elimination and recall preservation with the help of a plot like Fig. 19. Unfortunately, in practice, such a figure is not available because we do not know the location of the test data—the

location is what we want to predict in the first place. While we picked 5% via empirical intuition, a more analytical approach should be used to estimate the optimal PCANS discard rate for the production environment. To do so, one needs to collect multiple sessions of labeled validation data, preferably over a long time horizon and under various environmental conditions, and leverage those to acquire the best estimate of the optimal PCANS discard rate.

VIII. INLIER REGRESSION

The second sub-problem conceptualized in Section VI is inlier regression, which aims to predict the x-coordinate of inliers in the North and South. We train a North (South) inlier regressor on the North (South) inlier RSSI patterns and their x-coordinates. The same four data transformation methods are applied to the data before they are fed to the SVR-based regressor. The hyperparameters and the tuning process are explained in Section III-D and Section III-E.

Fig.s 20(A) and (B) show the performance of the North and South inlier regressors, respectively, in the form of CDF curves of error, with the same layout and labeling as described before. In the North, all four data transformation methods achieve the desired performance of ≤ 4 m of error in 90% of the cases, with STD and KPCA tied as the best performers. In the South, all but SSTCA fulfill the desired performance, with STD and KPCA again tied as the best performers. Thus, based on the localization performance, both STD and KPCA are desirable.

Fig. 20(C) illustrates the time cost of tuning the hyperparameters for each data transformation method on a Macbook Pro with an M1 Pro chip and 32 GB of RAM. The x-axis lists the four methods, the y-axis represents the time in seconds, and the blue and orange bars correspond to the tuning cost of the North and South regressor. Between the winners of the previous round, STD is almost 100 times faster than KPCA. Therefore, the overall winner of the data transformation method for inlier regression is STD.

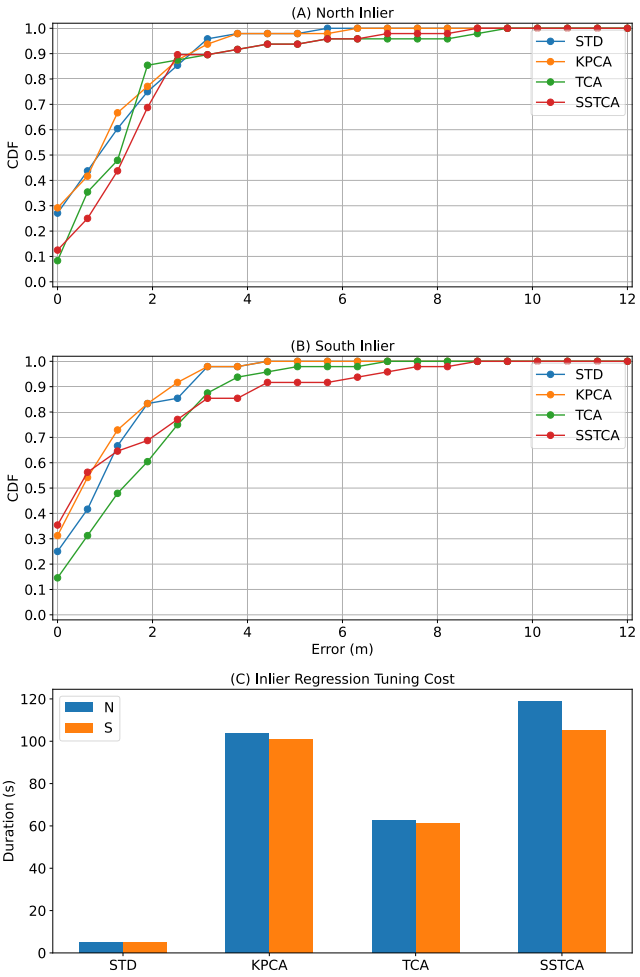


FIGURE 20. Performance of inlier regression across the four data transformation methods and their hyperparameter tuning costs.

IX. INXMODEL

Combining the final version of inlier classification (pre-screening, plus North and South inlier classification, plus post-screening) and inlier regression (North and South inlier regressor with STD), we present the multi-stage *InXModel* (see Fig. 11 for the flow chart). The overall performance of the *InXModel* across the four data transformation methods is shown in Fig. 21(A), where the layout and labeling of the figure remain the same as described before. The data transformation is for inlier classification only. All four data transformation methods reach the goal of ≤ 4 m of error in 90% of the cases, with STD scoring the highest. Fig. 21(B) shows the hyperparameter tuning costs of the four methods, with the tuning carried out on the same machine, as noted before. STD is again the fastest. Therefore, like inlier regression, STD is the best data transformation method for inlier classification.

With STD at the helm for both inlier classification and regression, the *InXModel* achieves a mean absolute localization error of 1.78 m, a CDF probability of 94.3% for errors ≤ 4 m, and a data loss rate of 30.2%. Given an RSSI pattern, the *InXModel* will predict whether it belongs to the North or South and its x-coordinate relative to the sensor designated as the origin (in our case, the origin is S3).

X. COMPARISON WITH EMDT-WKNN

Reference [6] proposes the Empirical Mode Decomposition with soft Thresholding and Weighted K-Nearest Neighbors method (EMDT-WKNN). The approach is used to construct a fingerprinting radio map with smoothed RSSI data collected over several months, followed by weighted K-nearest neighbors to localize an RSSI pattern (see Section II-C for a summary of the concept, and Section III-B for the implementation details). Although the study focuses on indoor localization, its experimental procedure and goal of reducing the impact of RSSI temporal fluctuations align with ours. Hence, we implement a customized version of the EMDT-WKNN method, tune its hyperparameters (see

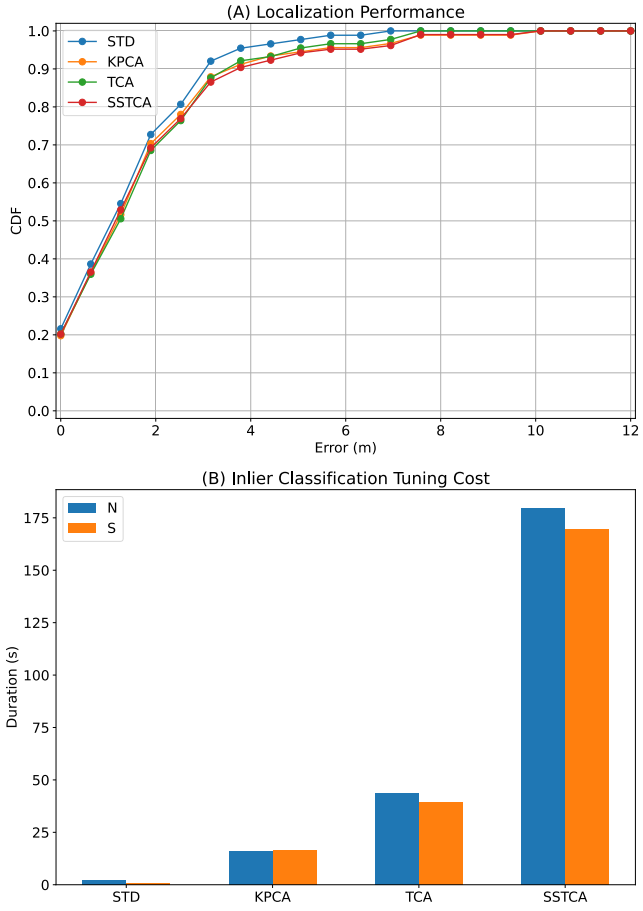


FIGURE 21. Performance of the *InXModel* across the four inlier classification data transformation methods and their hyperparameter tuning cost.

Section III-B for details), and compare its performance with the *InXModel*. The result is shown in Fig. 22 with the same layout as described above. The blue and orange curves correspond to the *InXModel* (with STD for both inlier classification and regression) and the EMDT-WKNN method.

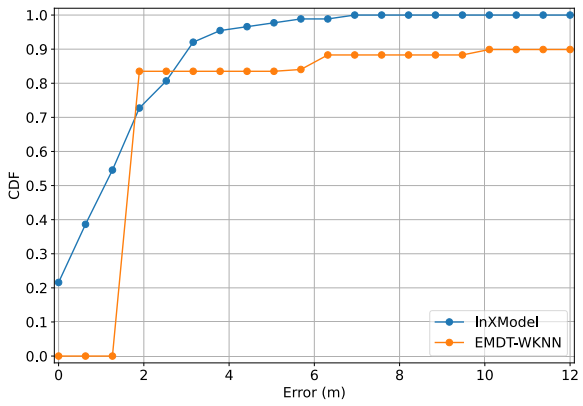


FIGURE 22. Localization performance comparison: *InXModel* vs. EMDT-WKNN.

The EMDT-WKNN method yields a lower data loss rate of 21.7%, but it performs worse than the *InXModel*, only achieving $\leq$ 4 m of error in 83.5% of the cases. Note that the data loss rate for the EMDT-WKNN method refers to the percentage of discarded data over the entire test data (see Section III-B for how data are discarded in EMDT-WKNN), which differs from the inlier-based data loss rate of the *InXModel*; the EMDT-WKNN method does not distinguish between inliers and outliers

Further analysis of the performance of the EMDT-WKNN method reveals the distribution of the unacceptable errors (i.e., errors larger than 4 m) in Table 4. The EMDT-WKNN method accumulates 31 instances of unacceptable error, among which 22, or 71%, are err-opp. Among the err-opp, more than half (13 out of 22) are from the outliers, despite the outliers having less share of unacceptable error than the inliers (14 vs. 17). Thus, the inability to control err-opp in the outliers is the main reason for the lackluster performance of the EMDT-WKNN method.

TABLE 4. Distribution of unacceptable error in the performance of The EMDT-WKNN method.

Type	Total	Error >4 m	Err-Opp
Inlier	74	17	9
Outlier	114	14	13

The EMDT-WKNN method does not properly handle outliers because outliers rarely appear in indoor localization. In an indoor environment, APs or sensors are usually installed close to the physical boundary of the structure, making most of the RPs and TPs inliers. Data from the inliers tend to have higher quality than those from the outliers due to less ambiguity and noise. We have seen in Fig. 15 and Fig. 20 that if outdoor localization is restricted to inliers, it is possible to achieve good performance. Unfortunately, outliers are inevitable, and they pose an inherent difficulty in RSSI-based outdoor localization.

Another factor contributing to the poor performance of the EMDT-WKNN method is that our experimental setup involves only four sensors. In [6], the number of APs is 10, which means their RSSI pattern is 10-dimensional; ours is only four. The considerable reduction in available information in each RSSI pattern may negatively affect the method's performance.

Finally, the RP density in our dataset (four-meter intervals between RPs) is half of that in [6] (two-meter intervals between RPs). Given that the performance of fingerprinting is heavily dependent on the density of the radio map, it is not surprising that the EMDT-WKNN method performs poorly against our dataset.

We speculate that the EMDT-WKNN method might fare better on data collected in our production environment. While the RP density will not increase due to scalability concerns, the production environment has many more sensors. A probe request in the production environment is typically captured by

six or more sensors nearby, which might provide sufficient information for each RSSI pattern. Outliers can also be addressed if we adapt some of the screening methods used in the *InXModel* to the EMDT-WKNN method. In summary, despite the poor showing with our dataset, we think the EMDT-WKNN method may have a better chance with the RSSI data from the production environment.

XI. CONCLUSION

In this paper, we discuss temporal fluctuations in RSSI in an outdoor environment and demonstrate their negative effect on passive outdoor localization models proposed in our prior work. We then address the signal fluctuation issue, step by step, by reformulating the localization problem, establishing a new data layout, and conceptualizing and implementing a multi-stage *InXModel*. The *InXModel* uses STD as the data transformation method and consists of a three-step inlier classification stage (pre-screening, plus inlier classification, plus post-screening) and a subsequent inlier regression stage. The performance of the *InXModel* achieves our goal of ≤ 4 m of error in 94.3% of the cases and has an acceptable data discard rate of 30.2%. We compare the *InXModel* with a recently reported EMDT-WKNN method explicitly designed to handle RSSI temporal fluctuation indoors and find that the latter has a poorer performance. We speculate that having to contend with outliers, not having sufficient dimensions in each RSSI pattern, and not having a radio map with sufficiently high RP density are the causes of the poor performance of the EMDT-WKNN method. These factors are also likely why a good indoor localization model cannot readily translate its performance to an outdoor setting.

Limitations. There are several limitations in this paper. First, our data collection efforts and experimental evaluation are limited exclusively to the 2.4 GHz band. This choice was motivated both by the ubiquitous use of 2.4 GHz access networks in public spaces and the hardware limitations of our streetscape testbed. Second, despite accounting for RSSI temporal fluctuation, our data do not consider signal fluctuations caused by obstructions or interference, as they are all collected in a semi-controlled area with no additional devices nearby or line-of-sight obstructions. Further, all of the (mock) probe requests emitted during the training and testing sessions originate from the same emitter, which means our dataset also fails to take into account RSSI fluctuations attributed to hardware variation. In future research, we will include these confounding factors in the dataset by collecting RSSI data directly in the production environment and using multiple emitters. We will examine the potential impact of these factors and investigate ways to mitigate them.

Finally, the *InXModel* is restricted to a sensor layout where the two sensors on the North are directly opposite to the two on the South. However, in reality, the sensor layout might not follow this rectangular pattern, and unplanned sensor shutdowns can further complicate the situation. In future research, we will study the flexibility and scalability of the *InXModel* when the sensor layout changes. Since the EMDT-WKNN method is more resistant to dynamic changes in sensor availability due to its fingerprinting nature, we will also explore better adaptations of this method for outdoor localization.

ACKNOWLEDGMENT

The authors would like to acknowledge support from 1) the City of West Palm Beach, the Knight Foundation, and the Community Foundations of Palm Beach and Martin Counties; 2) Chris Roog, the Executive Director of the Community Redevelopment Agency in the City of West Palm Beach; and 3) Florida Atlantic University's I-SENSE Team. This work derives from Fanchen Bao's dissertation [1].

REFERENCES

[1] F. Bao, "RSSI-based passive localization in complex outdoor environments using Wi-Fi probe requests," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Florida Atlantic Univ., Boca Raton, FL, USA, 2023.

[2] N. Singh, S. Choe, and R. Punmiya, "Machine learning based indoor localization using Wi-Fi RSSI fingerprints: An overview," *IEEE Access*, vol. 9, pp. 127150–127174, 2021.

[3] F. Bao, S. Mazokha, and J. O. Hallstrom, "Mobility intelligence: Machine learning methods for received signal strength indicator-based passive outdoor localization," *Adv. Sci., Technol. Eng. Syst. J.*, vol. 7, no. 6, pp. 269–282, Dec. 2022. [Online]. Available: https://www.astesj.com/v07/i06/p31/

[4] Q. Dong, F. Zhu, Y. Cai, L. Fang, and M. Lu, "Analysis of RSSI feasibility for sensor positioning in exterior environment," in *Proc. Wireless Telecommun. Symp. (WTS)*, Apr. 2021, pp. 1–7.

[5] S. J. Pan, V. W. Zheng, Q. Yang, and D. H. Hu, "Transfer learning for WiFi-based indoor localization," in *Proc. AAAI Conf. Artif. Intell.*, 2008, p. 6.

[6] R. Zhou, F. Meng, J. Zhou, and J. Teng, "A Wi-Fi indoor positioning method based on an integration of EMDT and WKNN," *Sensors*, vol. 22, no. 14, p. 5411, Jul. 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/14/5411

[7] S. Mazokha, F. Bao, J. Zhai, and J. O. Hallstrom, "MobIntel: Sensing and analytics infrastructure for urban mobility intelligence," *Pervas. Mobile Comput.*, vol. 77, Oct. 2021, Art. no. 101475. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574119221001097

[8] K. Shin, R. McConville, O. Metatla, M. Chang, C. Han, J. Lee, and A. Roudaut, "Outdoor localization using BLE RSSI and accessible pedestrian signals for the visually impaired at intersections," *Sensors*, vol. 22, no. 1, p. 371, Jan. 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/1/371

[9] A. Firdausi and G. P. N. Hakim, "RSSI indoor outdoor personal localization: A study to found targeted social engineering victim by attacker via wireless methods," *J. Robot. Control (JRC)*, vol. 2, no. 4, pp. 328–331, 2021. [Online]. Available: https://journal.umy.ac.id/index.php/jrc/article/view/9775

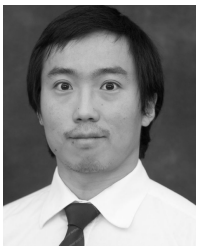
[10] S. Barai, D. Biswas, and B. Sau, "Sensors positioning for reliable RSSI-based outdoor localization using CFT," in *Proc. IEEE Int. Symp. Sustain. Energy, Signal Process. Cyber Secur. (iSSSC)*, Dec. 2020, pp. 1–5.

[11] D. Li, Y. Lei, X. Li, and H. Zhang, "Deep learning for fingerprint localization in indoor and outdoor environments," *ISPRS Int. J. Geo-Inf.*, vol. 9, no. 4, p. 267, Apr. 2020. [Online]. Available: https://www.mdpi.com/2220-9964/9/4/267

[12] Y. Lin, K. Yu, L. Hao, J. Wang, and J. Bu, "An indoor Wi-Fi localization algorithm using ranging model constructed with transformed RSSI and BP neural network," *IEEE Trans. Commun.*, vol. 70, no. 3, pp. 2163–2177, Mar. 2022.

[13] P. Ssekidde, O. S. Eyobu, D. S. Han, and T. J. Oyana, "Augmented CWT features for deep learning-based indoor localization using WiFi RSSI data," *Appl. Sci.*, vol. 11, no. 4, p. 1806, Feb. 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/4/1806

[14] N.-S. Duong and T.-M. Dinh, "Indoor localization with lightweight RSS fingerprint using BLE iBeacon on iOS platform," in *Proc. 19th Int. Symp. Commun. Inf. Technol. (ISCIT)*, Sep. 2019, pp. 91–95.
[15] K. K. Mamidi and K. Prasad, "ALTAR: Area-based localization techniques using AoA and RSS measures for wireless sensor networks," in *Proc. Int. Conf. Contemp. Comput. Informat. (IC3I)*, Dec. 2019, pp. 160–168.
[16] G. Guo, R. Chen, F. Ye, X. Peng, Z. Liu, and Y. Pan, "Indoor smartphone localization: A hybrid WiFi RTT-RSS ranging approach," *IEEE Access*, vol. 7, pp. 176767–176781, 2019.
[17] M. T. Hoang, B. Yuen, X. Dong, T. Lu, R. Westendorp, and K. Reddy, "Recurrent neural networks for accurate RSSI indoor localization," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10639–10651, Dec. 2019.
[18] M. Atashi, M. Salimibeni, P. Malekzadeh, M. Barbulescu, K. N. Plataniotis, and A. Mohammadi, "Multiple model BLE-based tracking via validation of RSSI fluctuations under different conditions," in *Proc. 22th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2019, pp. 1–6.
[19] Q. Chang, Q. Li, Z. Shi, W. Chen, and W. Wang, "Scalable indoor localization via mobile crowdsourcing and Gaussian process," *Sensors*, vol. 16, no. 3, p. 381, Mar. 2016. https://www.mdpi.com/1424-8220/16/3/381
[20] W. Zhao, S. Han, R. Q. Hu, W. Meng, and Z. Jia, "Crowdsourcing and multisource fusion-based fingerprint sensing in smartphone localization," *IEEE Sensors J.*, vol. 18, no. 8, pp. 3236–3247, Apr. 2018.
[21] S.-h. Jung, B.-c. Moon, and D. Han, "Unsupervised learning for crowdsourced indoor localization in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 11, pp. 2892–2906, Nov. 2016.
[22] L. Oliveira, D. Schneider, J. De Souza, and W. Shen, "Mobile device detection through WiFi probe request analysis," *IEEE Access*, vol. 7, pp. 98579–98588, 2019.
[23] G. Li, E. Geng, Z. Ye, Y. Xu, J. Lin, and Y. Pang, "Indoor positioning algorithm based on the improved RSSI distance model," *Sensors*, vol. 18, no. 9, p. 2820, Aug. 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/9/2820
[24] Z. Huang, L. Xu, and Y. Lin, "Multi-stage pedestrian positioning using filtered WiFi scanner data in an urban road environment," *Sensors*, vol. 20, no. 11, p. 3259, Jun. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/11/3259
[25] A.-O. Boudraa and J.-C. Cexus, "EMD-based signal filtering," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 6, pp. 2196–2202, Dec. 2007.
[26] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, Feb. 2011.
[27] H. Zou, Y. Zhou, H. Jiang, B. Huang, L. Xie, and C. Spanos, "Adaptive localization in dynamic indoor environments by transfer kernel learning," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2017, pp. 1–6.
[28] P. Li, H. Cui, A. Khan, U. Raza, R. Piechocki, A. Doufexi, and T. Farnham, "Deep transfer learning for WiFi localization," in *Proc. IEEE Radar Conf. (RadarConf)*, May 2021, pp. 1–5.
[29] F. Pedregosa, S. Varoquaux, A. Gramfort, V. Michel, and B. Thirion, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Dec. 2011.



FANCHEN BAO received the B.S. degree in biological sciences from Peking University, China, in 2012, and the M.S. degree in artificial intelligence and the Ph.D. degree in computer science from Florida Atlantic University, in 2022 and 2023, respectively.

From 2019 to 2023, he was a Research Assistant with the Institute for Sensing and Embedded Network Systems Engineering, Florida Atlantic. His research interests include reducing the impact of RSSI signal fluctuation on localization performance and passive Wi-Fi localization in city streetscapes.

STEPAN MAZOKHA received the B.Sc. degree in computer engineering from Taras Shevchenko National University of Kyiv, in 2016, and the M.Sc. degree in computer science from Florida Atlantic University, in 2018, where he is currently pursuing the Ph.D. degree.

Since 2017, he has been a Graduate Research Assistant with the Institute for Sensing and Embedded Network Systems Engineering (I-SENSE), Florida Atlantic. He is the author of seven peer-reviewed publications. His research interests include the Internet of Things, mobility intelligence in city streetscapes, Wi-Fi localization, and device re-identification.

JASON O. HALLSTROM received the B.S. degree in systems analysis and the M.A. degree in economics from Miami University, in 1998, and the M.S. and Ph.D. degrees in computer and information science from The Ohio State University, in 2004.

From 2004 to 2010, he was an Assistant Professor with the School of Computing, Clemson University. He was named as an Associate IDEaS Professor, in 2010. Additionally, he was the Director of Technology for Clemson's Institute of Computational Ecology, from 2011 to 2014, and the Deputy Director, from 2013 to 2014. Since 2015, he has been the Founding Executive Director of the Institute for Sensing and Embedded Network Systems Engineering (I-SENSE), Florida Atlantic University, and a Professor with the Department of Electrical Engineering and Computer Science. His work has contributed to more than 140 publications, 200 presentations, and resulted in multiple awards, including the National Science Foundation CAREER Award, in 2008. His research interests include embedded network systems, internet-scale sensing infrastructure, the Internet of Things, software engineering, and computer science education.