LLOCUS: Learning-based Localization Using Crowdsourcing

Shamik Sarkar[1] (Shamik.Sarkar@utah.edu), Aniqua Baset[1], Harsimran Singh[1], Phillip Smith[1]
Neal Patwari[2], Sneha Kasera[1], Kurt Derr[3], and Samuel Ramirez[3]
[1]University of Utah, [2]Washington University in St. Louis, [3]Idaho National Labs

ABSTRACT

We present LLOCUS, a *novel* learning-based system that uses mobile crowdsourced RF sensing to estimate the location and power of unknown mobile transmitters in real time, while allowing unrestricted mobility of the crowdsourcing participants. We carefully identify and tackle several challenges in learning and localizing, based on RSS, in such a dynamic environment. We decouple the problem of localizing a transmitter with unknown transmit power into two problems, 1) predicting the power of a transmitter at an unknown location, and 2) localizing a transmitter with known transmit power. LLOCUS first estimates the power of the unknown transmitter and then scales the reported RSS values such that the unknown transmit power problem is transparent to the method of localization. We evaluate LLOCUS using three experiments in different indoor and outdoor environments. We find that LLOCUS reduces the localization error by 17-68% compared to several non-learning methods.

CCS CONCEPTS

• **Networks** → **Location based services**; **Mobile ad hoc networks**; *Mobile and wireless security*.

1 INTRODUCTION

Localization of a radio-frequency (RF) transmitter is a fundamental requirement in wireless and mobile networks. In addition to supporting typical benign applications including location-based services, localization can also help locate sources of malicious behavior, e.g., a wireless spectrum jammer. However, large scale placement and management of static receivers over wide geographical areas, for the purpose of localizing malicious jammers, is not feasible or too expensive. Ad hoc crowdsourcing, where a large number of nodes are recruited for a short duration at different locations and at different times to assist in localization in a dynamic manner, offers a *practical* approach to localization of such RF transmitters. The overhead of performing wide scale RF sensing gets shared among the crowd. In the crowdsourced RF sensing scenarios, if the crowdsourcing sensors/receivers are mobile, their received signal strength (RSS) measurements do not offer a consistent fingerprint, i.e., when a transmitter is to be located, the participating receivers may not be located at any of the locations where the receivers were present when the training measurements were made. Thus, despite the fact that fingerprint methods revolutionized RSS based indoor localization [21, 26], there is no fingerprint method, to date, which addresses localization when both the ends (the 'ends' being transmitters and receivers) are mobile. Instead, when 'both ends are mobile', localization algorithms resort to methods based on radio wave propagation path loss models (PLMs) [23], which we refer to as non-learning solutions.

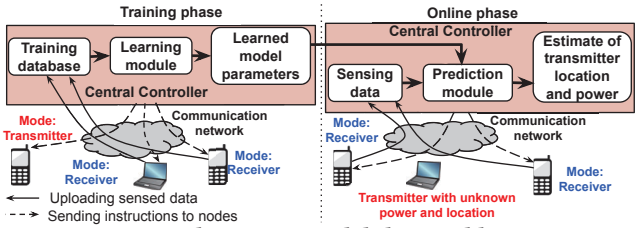


Figure 1: Crowdsourcing model that enables LLOCUS

In this work, we consider the realistic crowdsourced spectrum sensing scenario in which all the transmitters and the receivers can be mobile, and the transmitters, possibly spectrum violators trying to hide their locations, can modify their transmit power. We investigate an RSS-based machine learning approach for this scenario to overcome the problem of inconsistent fingerprints, and at the same time improve the localization accuracy with respect to the PLM based localization methods. In many scenarios, the peculiarities of the RF environment cannot be incorporated in a PLM. In our work, we learn environment-specific models using diverse training data that can accurately capture the RF environment. We exploit crowdsourced RF sensing on commodity mobile devices, whose feasibility has been demonstrated in recent works [2, 9], for diverse data collection across space and time. We stick to RSS-based localization as RSS measurements are easily available on commodity mobile devices, in contrast to *angle of arrival* measurements that require specialized hardware, and *time of arrival* measurements which require receiver synchronization [20].

We present LLOCUS, a novel supervised learning-based system that uses crowdsourced RF sensing to estimate the location and power of mobile transmitters in near real time, while allowing unrestricted mobility of the crowdsourcing nodes. Furthermore, when multiple transmitters transmit simultaneously, LLOCUS can detect the number of active transmitters and localize each of them. To the best of our knowledge, our work is the first to explore supervised learning for localization in this generalized mobility scenario. LLOCUS can be used for localizing unauthorized transmitters [7] with unknown power; locating incumbent or secondary transmitters for dynamic spectrum access [15, 22]; all using crowdsourced mobile devices rather than a static infrastructure deployed for this purpose.

Crowdsourcing model: A variety of RF sensing nodes, having unrestricted mobility, participate in crowdsourcing, as shown in Figure 1. These nodes interact with a cloud-based central controller (CC), via cellular network or WiFi, which sends instructions that

Mobihoc '20, October 11–14, 2020, Boston, MA, USA

ACM ISBN 978-1-4503-8015-7/20/10…$15.00
https://doi.org/10.1145/3397166.3409142

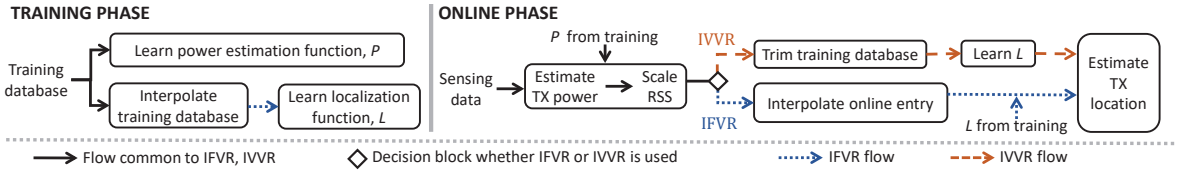


Figure 2: Overview of learning and prediction in LLOCUS

specify the operation mode (transmitter or receiver), frequency band, and duration of operation to the participants. These nodes continuously report their sensing information (measured RSS and device locations, obtained via GPS or any indoor localization system) to the CC. Since reporting of sensing information results in energy and bandwidth overhead, the participants must be incentivized for their contributions [10, 19]. However, we do not investigate any incentive framework in this paper. During data collection in the training phase, the CC operates one of the mobile nodes as a transmitter and the remaining nodes as receivers. To ensure that the training transmissions are spatially diverse, the role of transmitter is passed periodically to different participants. The collected data is combined into a training database, which is used by the learning module of LLOCUS, to learn parameters for models that characterize the RF environment. During the online phase the CC operates all the participating nodes as receivers. As soon as a new batch of sensing data arrives, using the current sensing data and the learned models, the prediction module of LLOCUS estimates the transmit power and location of the active transmitter.

Challenges in LLOCUS: There are three major challenges in our learning-based localization:
(a) *Learning when both ends are mobile*: Unlike traditional learning-based localization systems [21, 26], we cannot assume the presence of static reference devices in LLOCUS. This creates a challenge, which we call *both ends are mobile*, in forming the feature set for learning. Since the transmitters can be mobile, and the position as well as the number of the receivers change with time, the reported RSS values cannot be used directly as features for learning/prediction. Each RSS value is associated with a receiver location, thus, using the reported RSS values as features would lead to a different feature set for every batch of sensing data.
(b) *Sparsity of training*: Since the training transmitters are crowdsourced, the number of transmitter locations, for which we have RSS measured by the training receivers, depends on the training transmitters' mobility. Thus, the number of unique transmitter locations in the training database could be sparse with respect to the area of interest. However, our learning must be general enough to make accurate predictions even when the online transmitters' locations do not belong to the set of training transmitters' locations.
(c) *Transmit power variability*: If the training transmitters use a fixed transmit power, then the model learned from the training data will be useful in making predictions only if the online transmitters use the same power as the training transmitters. Since the online transmitters are unknown, we may have no prior knowledge about their transmit power. Thus, the *transmit power variability* problem, if not tackled carefully, may render the learned model inefficient.

Overview of LLOCUS: We decouple the problem of localizing a transmitter with unknown power into two problems, 1) predicting the power of a transmitter at an unknown location, and 2) localizing a transmitter with known power. In the online phase, LLOCUS first estimates the transmitter's power and then scales the RSS measurements, as shown in Figure 2, such that the *transmit power variability* problem is transparent to the localization method. We first explain our solution for the localization problem, and then describe how we estimate the transmitter's power.

For solving the *both ends are mobile* problem, we handle the receivers' mobility by interpolating every batch of reported RSS at a set of static, virtual locations that act like anchors. This set of interpolated RSS values can then be used as features for learning/prediction. We propose two approaches: interpolation at fixed virtual receiver locations (IFVR) and interpolation at variable virtual receiver locations (IVVR). We show that IVVR helps achieve higher localization accuracy than IFVR, while IFVR is more time efficient, as explained in Section 3.1. Based on the desired accuracy and computing resources available, one can choose to use LLOCUS with either IFVR or IVVR.

Next, to handle transmitter mobility, we explore two localization algorithms, a kNN prediction method, and a regularized radial basis interpolation method[1], that address the *sparsity of training* problem. These localization methods use the interpolated RSS values to learn a function (denoted by L in Figure 2) that relates the RSS at the virtual receiver locations to the transmitter's location. The learned function, L, is later used to localize the transmitters in the online phase. The localization methods are oblivious to the receivers' mobility by virtue of the RSS interpolation.

For estimating the power of an online transmitter, without knowing its location, we use the observation that, if the power of a transmitter changes, the average of RSS measured by the receivers near the transmitter would also change, irrespective of the transmitter's location. Thus, using the training database, LLOCUS learns a function (denoted by P in Figure 2) that, based on the measured RSS and locations of the online receivers, can predict the transmit power of an unknown transmitter. We use a support vector machine based regression method to learn this power estimation function.

For localizing simultaneous transmitters, we first estimate the number of simultaneously active online transmitters based on the measured RSS and locations of the online receivers. Then, we treat each of the detected transmitters as non-simultaneous, by defining an approximate region of presence around them, and localize each of them based on the RSS measurements inside their approximate region of presence. Thus, we can use L and P, learned based on non-simultaneous training transmissions, for estimating the location and power of the simultaneous online transmitters.

We evaluate LLOCUS using three experiments in different indoor and outdoor environments. In our evaluations LLOCUS reduces the localization error by 17-68% compared to a number of non-learning methods, and estimates the power of an unknown transmitter with an average error within 3 dB.

[1]Note that, in this paper, we use interpolation for two purposes. First, to obtain RSS values at virtual receiver locations, and second, for learning-based localization.

In summary, we make the following contributions:

- Develop a novel learning-based localization system, LLOCUS, that can learn and localize in crowdsourced environments with dynamic participants.
- Develop a novel learning-based method for estimating a transmitter's power using RSS measured by nearby receivers.
- Improve the state-of-art for localizing multiple transmitters that are active simultaneously.
- Thoroughly evaluate LLOCUS in diverse environments, and compare its performance with that of various other non-learning methods.
- Implement a prototype of LLOCUS that is capable of near real time operation.

2 RELATED WORK

Learning based localization: Localization of a mobile node via learning has been studied extensively in the context of WiFi fingerprinting, where the basic idea is to capture RSS fingerprints from static access points (AP) for all the locations in an indoor area. Subsequently, the location of a mobile node, in the same area, is obtained by searching for a match between the current RSS fingerprint and the ones collected previously by deterministic/probabilistic methods [21, 26]. However, this method requires extensive manual effort in collecting the training fingerprints. To circumvent this problem, researchers have investigated ways to make the system work even if the fingerprints are spatially sparse [16, 18]. They have also used crowdsourcing for collecting the training fingerprints [3, 17]. Irrespective of the approach, all of these learning-based localization methods ultimately depend on the RSS to/from the nearby "static" APs. In our problem, both the transmitter and the receivers are mobile; be it the training or the online phase. Thus, none of the existing approaches can be adopted directly to solve our problem.
Crowdsourced localization: Crowdsourced RF sensing [1, 2, 8, 19, 25] uses specially equipped vehicles, networked spectrum analyzers, and mobile devices as the crowdsourced nodes; where each of these possesses an RF sensing capability. The idea explored in these works is to crowdsource distributed RF sensing, and based on the sensing data extract several important pieces of information, including the location of the RF transmitters. However, none of these works explore learning-based localization in a general setting, indoor or outdoor, with unrestricted transmitter and receiver mobility.

3 METHODOLOGY OF LLOCUS

We first define our notation and then describe LLOCUS in detail.

Notation: During the data collection at time t, a transmitter located at (x_t^T, y_t^T) transmits on frequency f_t^T, with power p_t^T. Note we use T for variables related to the transmitter, and $\intercal$ for transpose. At each time instant, the training transmitter reports $t, p_t^T, (x_t^T, y_t^T)$, and each of the training receivers reports $t, z_{i,t}, (x_{i,t}, y_{i,t})$ to the CC. Here, $z_{i,t}$ (dBm) is the RSS measured by receiver i at time t, $(x_{i,t}, y_{i,t}) \in R_t$ is receiver i's location, and R_t is the set of locations where participating receivers are present at time t. The CC combines the set of current reported data in an entry and appends it to the training database. Using this training database, LLOCUS learns the functions L and P (shown in Figure 2). During the online phase, at time t', each participating receiver i reports to the CC: $t', z_{i,t'}$, $(x_{i,t'}, y_{i,t'}) \in R_{t'}$. The CC feeds the online data to P and L to estimate the power $p_{t'}^T$ and location $(\hat{x}_{t'}, \hat{y}_{t'})$ of the transmitter.

3.1 Both ends are mobile

The localization function L, shown in Figure 2, is the unknown target function that we want to learn, based on the set of training measurements: $\{(x_t^T, y_t^T), R_t, \{z_{i,t}\}; \forall (x_i, y_i) \in R_t\}$. Due to the mobility of the crowdsourced nodes, the tuples $(z_{i,t}, (x_{i,t}, y_{i,t}))$ correspond to different receiver locations for different time instants. Thus, we cannot simply use them as features for learning L. To tackle this issue, we map the tuples, $(z_{i,t}, (x_{i,t}, y_{i,t}))$, to a set of features that are common across all the training as well as the online entries. We make use of RSS interpolation to accomplish this feature mapping, as described below. In this section and Section 3.2, we assume that the training and the online transmitters have the same power. We relax this assumption in Section 3.3 where we present our solution for *transmit power variability*.

3.1.1 Interpolation at fixed virtual receiver locations (IFVR). In this feature mapping approach, we predetermine a set of 'fixed' locations, I, in the area of interest. For each of the entries in the training database, we interpolate the measured RSS values, $z_{i,t}$ at $(x_{i,t}, y_{i,t}) \in R_t$, to the set of locations in I, and obtain an interpolated training database. This process is shown in Figure 2 as 'interpolate training database'. We denote the interpolated RSS value at $(x_{j,t}, y_{j,t}) \in I$ by $z_{j,t}$. During the online phase at time t', when a new set of RF sensing data arrives at the CC, we also interpolate the reported RSS values, $z_{i,t'}$, measured by the current online receivers at $(x_{i,t'}, y_{i,t'}) \in R_{t'}$ to the locations in I (shown as 'interpolate online entry' in Figure 2). Now, we have a set of features, the interpolated RSS values at the set of locations in I, that are common across all the training and online entries. We identify $|I|$ uniformly spaced locations in the area for determining I.

3.1.2 RSS interpolation algorithms. We need an algorithm to interpolate the RSS field at the set of locations in I. We start our investigation with two RSS interpolation methods, kriging interpolation and inverse distance weighting, that are known to perform well on the RSS field [6, 11].

Kriging interpolation: In this method, RSS is modeled as an intrinsically stationary random field, i.e., the mean and covariance functions are spatially invariant. Kriging computes the interpolated RSS at $(x_{j,t}, y_{j,t}) \in I$ as, $z_{j,t} = \sum_i w_{i,t} z_{i,t}, \forall (x_{i,t}, y_{i,t}) \in R_t$. The optimal weights, $\{w_{i,t}\}$, are obtained as described in [6]. For the RSS field to be intrinsically stationary, it is required that $\mathbf{E}[z_{i,t}] - \mathbf{E}[z_{i+h,t}] = 0$, which is not the case for RSS, in general. To overcome this problem, we use detrending, as given in [6].

Inverse distance weighting (IDW): In this method, we compute the interpolated RSS at $(x_{j,t}, y_{j,t})$ as, $z_{j,t} = \frac{\sum_i z_{i,t}/d_{ij}^c}{\sum_i 1/d_{ij}^c}, \forall (x_{i,t}, y_{i,t}) \in R_t$. Here d_{ij} is the distance between $(x_{i,t}, y_{i,t})$, $(x_{j,t}, y_{j,t})$ and c is a constant, obtained by k-fold cross validation [24]. The strategy is to estimate the RSS as a weighted average of the measured RSS values. The weighing is done such that the RSS values measured close to the target location, $(x_{j,t}, y_{j,t})$, have higher weight compared to the RSS values measured far away from $(x_{j,t}, y_{j,t})$.

For evaluating the above RSS interpolation methods, we interpolate the RSS field to the locations for which we have RSS values

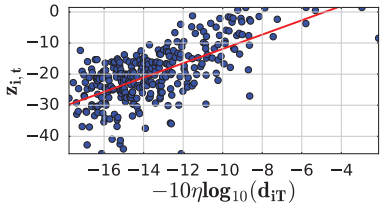


Figure 3: Linear regression in TxAIDW

measured by physical receivers. During this evaluation, we observe that sometimes the interpolated RSS deviates significantly from the measured RSS. This happens primarily due to the randomness in the RF environment. To address this issue, we take a more fundamental approach based on the following observation. In general, the RSS interpolation methods perform the interpolation without the knowledge of the transmitter's location. Thus, the estimated RSS in the above methods depends only on the observed RSS values, and the locations of observations. However, in the context of LLOCUS, the training database also has the transmitters' actual locations. We propose a new RSS interpolation method, transmitter assisted IDW (TxAIDW), that exploits the availability of transmitters locations. In TxAIDW, we use the training data for learning a function that estimates a coarse value of RSS at the target location. Then, we use the current RSS measurements at the receivers near the target location and further refine the coarse estimate, to counter the randomness in the RF environment. We show via evaluations, in Section 4, that TxAIDW has lower RSS estimation error, compared to kriging and IDW. The details of TxAIDW are presented below.

TxAIDW: We exploit the availability of transmitter's location, (x_t^T, y_t^T), for producing the coarse estimate of RSS, $\hat{z}_{j,t}$, at $(x_{j,t}, y_{j,t})$. For that, we model the measured RSS, $z_{i,t}$, as [23]:

$$z_{i,t} = z_0 - 10\eta \log_{10}(d_{iT}) \tag{1}$$

where z_0 is the measured RSS at a location close to the transmitter, η is the radio wave path loss exponent, and d_{iT} is the distance between $(x_{i,t}, y_{i,t})$, (x_t^T, y_t^T). Next, we use the least square regression [24] to learn the linear curve, defined by (1), that has best fit with all the RSS measurements in the training database, as shown in Figure 3. The learned linear curve is defined by its slope, η, and intercept, z_0. Then, $\hat{z}_{j,t}$ is obtained by using the learned curve, and d_{jT}, the distance between (x_t^T, y_t^T) and $(x_{j,t}, y_{j,t})$; i.e., $\hat{z}_{j,t} = z_0 - 10\eta \log_{10}(d_{jT})$. Since the RSS values often deviate from (1), as shown in Figure 3, we refine $\hat{z}_{j,t}$ by estimating the randomness in the RF environment, at $(x_{j,t}, y_{j,t})$. For that, we identify the deviation from (1) in the current measured RSS values close to the target location (within d_{ref} meters from $(x_{j,t}, y_{j,t})$), and use the IDW method on these deviation values to estimate the deviation at $(x_{j,t}, y_{j,t})$. Thus, the final estimated RSS at $(x_{j,t}, y_{j,t})$ is:

$$z_{j,t} = \hat{z}_{j,t} + \frac{\sum_i (\hat{z}_{i,t} - z_{i,t})/d_{ij}^c}{\sum_i 1/d_{ij}^c} \quad \forall (x_{i,t}, y_{i,t}) \in R_t \ s.t. \ d_{ij} < d_{ref}$$

where $\hat{z}_{i,t}$ is the coarse estimate of RSS at $(x_{i,t}, y_{i,t})$. The intuition behind our refinement strategy is that, the randomness in the RF environment close to $(x_{j,t}, y_{j,t})$ would possibly indicate the deviation in RSS at $(x_{j,t}, y_{j,t})$, from (1).

Using TxAIDW, we can interpolate the RSS values in the training entries to the set of location in I. In IFVR we also interpolate the RSS values in online entry to the location in I. However, for the online entry we do not have the transmitter's location. Thus, without the transmitter's location we have to use less efficient methods, kriging or IDW, for interpolating the online entry. To overcome this limitation of IFVR, we propose another method for feature mapping below, that does not require RSS interpolation for the online entry.

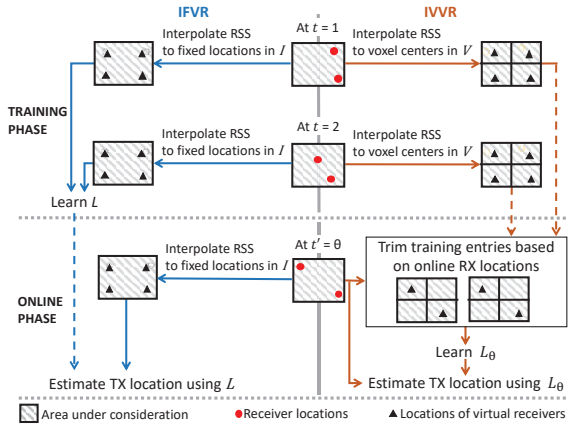


Figure 4: Differences in IFVR (left) and IVVR (right)

3.1.3 Interpolation at variable virtual receiver locations (IVVR). We first divide the area of interest into a rectangular grid. Let the set of voxel centers of the resulting voxels be V. For each entry in the training database, we interpolate the RSS values measured by the training receivers to the set of locations in V, resulting in an interpolated training database, similar to the one obtained for IFVR, except V need not be same as I. In the online phase, when a set of RF sensing readings arrive at the CC, we extract a trimmed training database from the interpolated training database (shown as 'trim training database' in Figure. 2). The entries in the trimmed training database consist of interpolated RSS values for the voxel centers corresponding to the voxels inside which the current online receivers are located. We choose a small value of the voxel length so that the virtual receiver locations in the trimmed training database are same as the receiver locations in the online entry. For every new online entry, we use a different trimmed training database depending on the online receivers' locations. As the locations of the virtual receivers vary across predictions, we call this approach interpolation at variable virtual receiver locations.

3.1.4 IVFR vs. IVVR. Figure 4 shows the differences between IFVR and IVVR using a simple example, with two training entries, $t = 1, 2$, and one online entry, $t' = \theta$. Once the feature mapping is done, the next task is to learn the localization function, L (shown in Fig. 2), that will predict the locations of online transmitters. If IFVR is used, LLOCUS learns the function L, only once, during the training phase. In the online phase, we interpolate the reported RSS values to the locations in I, feed the interpolated RSS values to L, and get an estimate of the transmitter's location. On the other hand, with IVVR, LLOCUS re-learns the function L for every prediction in the online phase, as shown in Figure 2, 4. In Figure 4, L_θ represents the learned function at $t' = \theta$. The methods for learning L are presented in the next section. Due to the differences in the two feature mapping methods, they have different prediction time, t_p, defined as the time required to make a prediction once a set of sensed data arrives at the CC. With IVVR, LLOCUS learns repeatedly to reduce dependence

on RSS interpolation (no RSS interpolation for online entries), and, in turn, achieves lower localization error at the cost of higher t_p.

3.2 Sparsity of training

In this section, we investigate learning algorithms that can address the *sparsity of training* problem described in Section 1. We present two algorithms for learning the localization function, L, and describe how the learned function is used for making predictions. By virtue of the feature mapping methods described in Section 3.1, the same set of features, interpolated RSS at the virtual receiver locations are presented to the localization algorithms; for both learning and prediction. For the ease of explaining, we assume that the set of the features, used by the learning algorithm, is $\{z_{j,t}\}; \forall (x_{j,t}, y_{j,t}) \in A_{t'}$. In reality, $A_{t'} = I$ and $A_{t'} = R_{t'}$ for IFVR and IVVR, respectively. We use $\mathbf{z}_{t'}$ for the vector of RSS values at t', $[z_{1,t'}, z_{2,t'}, ...]$, and M is the set of time instants corresponding to the entries in the training database. When LLOCUS is used with IVVR, the learning of L happens repeatedly for every prediction, as described in Section 3.1.4. To ensure near real-time continuous operation of LLOCUS, we must have an algorithm that learns fast. Consequently, we cannot use iterative algorithms whose learning (convergence) time is non-deterministic. Next, we describe two algorithms that maintain the above criteria.

kNN localization (kNN): The kNN algorithm has the property that with enough training examples, the error of this method converges to that of the best possible predictor [24]. This property implies that when the training examples are spatially dense, this method will perform well. Moreover, the kNN method requires minimal time for learning, making it suitable for IVVR. To predict the transmitter's location, we determine k RSS vectors in the set $\{\mathbf{z}_t\}; \forall t \in M$ that are nearest to $\mathbf{z}_{t'}$ in the feature space. Then, we use the weighted average of the labels/targets, i.e., (x_t^T, y_t^T), corresponding to the chosen k RSS vectors, as the predicted location of the transmitter.

Regularized Radial basis interpolation (RRBI): The kNN localization is efficient for a high spatial density of training examples; however, in reality the training examples are often sparse. Addressing this sparsity of training examples is an important objective of LLOCUS. Interpolation via radial basis function method [4] has been considered for WiFi based indoor localization in [18]; where the training fingerprints are sparse, collected at room level granularity. Moreover, the learning algorithm in this method uses matrix operations that can be performed in a non-iterative fashion. Since this framework fits into our *sparsity of training* problem, we consider this as a candidate solution. Henceforth, we call this method localization via radial basis interpolation (RBI). Although the RBI method has the word 'interpolation' in its name, it is different from the RSS interpolation algorithms of Section 3.1.2. The RSS interpolation methods produce RSS as an output, but the RBI method produces a physical coordinate (two dimensional) as output.

While evaluating the RBI method, we observe that it performs no better than the kNN method. We further investigate this issue and identify that there are two contextual differences between WiFi fingerprinting for indoor localization, for which the RBI method has been developed, and our generalized framework in LLOCUS. First, in case of WiFi fingerprinting the training fingerprints are based on RSS measurements to/from static physical APs; whereas, in LLOCUS the training fingerprints are based on the estimated RSS values, via RSS interpolation, at the virtual receiver locations. Since the RSS interpolation methods are not perfect, the training fingerprints will always have some noise due to estimation error. This issue may lead to L learning a strong dependence on a few particular features that have a significant amount of noise in the estimated RSS. The second difference is that the RBI method has been developed for indoor environments. However, LLOCUS is not limited to any specific RF environment. Consequently, the Gaussian kernel used in RBI may not be the best kernel in all RF environments.

To make the RBI method usable in the context of LLOCUS, we propose a regularized, and more generalized version of the RBI method, which we call regularized RBI (RRBI). In RRBI, we use least square regularization [24] to generalize the learning of L. The least square regularization prevents the learning algorithm from learning strong dependence on a few features, i.e., it reduces the influence of individual features. In RRBI we also consider three kernel functions, Gaussian, inverse quadratic, and inverse multiquadric, that are relevant in this framework [4], and identify the best one, with respect to the RF environment, using cross validation.

In the RRBI method, we estimate the transmitter's location as:

$$(\hat{x}_{t'}, \hat{y}_{t'}) = \left(c_x + \sum_{k \in M} \alpha_k K\big(||\mathbf{z}_{t'} - \mathbf{z}_k||\big), c_y + \sum_{k \in M} \beta_k K\big(||\mathbf{z}_{t'} - \mathbf{z}_k||\big) \right)$$

where $\{\mathbf{z}_k\}$ is the feature vector (interpolated RSS at the virtual receivers locations) corresponding to the k^{th} entry in the training database. $\{\alpha_k\}$, $\{\beta_k\}$ are the set of weights of this learning algorithm, $K(\cdot)$ is the kernel function, and c_x, c_y are the mean of $\{x_t^T\}, \{y_t^T\}; \forall t \in M$, respectively. Essentially, the RRBI method estimates the transmitter's location by computing the weighted average of the distances between the features in the online entry and the features in the training entries in a high dimensional space. The kernel function, $K(\cdot)$, specifies the similarity between $\mathbf{z}_{t'}$ and $\mathbf{z}_k$ in a higher dimensional feature space [24], compared to the input space. This feature transformation makes the input space more expressive; often it makes a linearly non-separable input space to be linearly separable in the higher dimensional feature space. This, in turn, simplifies the learning problem.

To determine the weights, $\{\alpha_k\}$, we estimate the transmitter's location for all the entries in M, using the above equation, and minimize the squared error:

$$\min_{\boldsymbol{\alpha}} \sum_{t \in M} \left(x_t^T - c_x - \sum_k \alpha_k K\big(||\mathbf{z}_t - \mathbf{z}_k||\big) \right)^2 + \frac{\lambda}{2|M|} ||\boldsymbol{\alpha}||^2 \quad (2)$$

where $\boldsymbol{\alpha}$ is the vector of $\{\alpha_k\}; \forall k \in M$. The second part of the cost function of (2) is due to the least square regularization. λ is a hyper-parameter that can be obtained by cross validations. Taking partial derivative of the objective in (2) with respect to $\boldsymbol{\alpha}$ produces a set of linear equations that can be arranged in matrix form as,

$$\Big(\mathbf{K}^\intercal \mathbf{K} + (\lambda/|M|)\mathbf{I}\Big)\boldsymbol{\alpha} = \mathbf{K}^\intercal \mathbf{x}^T \quad (3)$$

where $\mathbf{K}[t,k] = K\big(||\mathbf{z}_t - \mathbf{z}_k||\big)$, $\mathbf{I}$ is the identity matrix, and $\mathbf{x}^T$ is the vector of $\{x_t^T - c_x\}; \forall t \in M$. Solving (3) produces the optimal weights for $\boldsymbol{\alpha}$. A similar method is used for determining $\{\beta_k\}$; that we omit for brevity. In Section 4, we show that RRBI performs better than kNN and RBI, in presence of the *sparsity of training* problem.

3.3 Transmit power variability

In Section 3.1 and 3.2, we assumed the online transmitters' power, $p^T_{t'}$, to be same as the training transmitters' power, p^T_t. Thus, the localization algorithms described in Section 3.2 would work properly if $p^T_{t'} = p^T_t$. To make the localization algorithms work with online transmitters of unknown power, LLOCUS first estimates the power of the online transmitter ('Estimate TX power' in Figure 2) and then scales the reported RSS values ('Scale RSS' in Figure 2) such that it appears to the localization algorithm that $p^T_{t'} = p^T_t$. Thus, we must have a way to estimate the online transmitter's power.

A PLM can be used to estimate the power of a transmitter, once its location has been estimated, as follows:

$$\hat{p}^T_{t'} = p + \frac{1}{|R_{t'}|}\sum_i \big(z_{i,t'} - (z_0 - 10\eta \log d_{iT})\big); \forall (x_{i,t'}, y_{i,t'}) \in R_{t'}$$

where d_{iT} is the distance between the estimated location of the transmitter and the receiver at $(x_{i,t'}, y_{i,t'})$. We start with the default transmit power, p, and add the average difference between the RSS measured by the online receivers, and what the PLM would expect the RSS to be at those receiver locations, for a transmitter at the estimated transmitter location, with power p. We call this method, transmitter power estimation using PLM (TPE-PLM). However, in LLOCUS, we must estimate the online transmitter's power, using the measured RSS and locations of the online receivers, before its location has been estimated (see Figure 2).

One method for estimating the online transmitter's power, that we call MAX-RSS, is to identify the maximum measured RSS and declare the same as the transmitter's power. Intuitively, this method may not work well when the density of the crowdsourcing nodes is low. In such cases, the maximum RSS might be measured at a significant distance from the transmitter's location. Thus, the transmitter's power can be much higher than the measured maximum RSS. We show via evaluations, in Section 4, that our intuition is correct. Instead of using just the maximum RSS, we can use all the RSS measurements to produce a more accurate estimate of the transmitter's power. EZ [12], uses a genetic algorithm (GA) for estimating location, transmit power, and path loss exponent of the PLM, of an unknown transmitter. However, we determine that the GA, being an iterative method, requires ≈ 3 minutes (on a server-class machine) for estimating the transmitter's location and power. Thus, EZ is not applicable for real time operations, when the online transmitters' location and power can change with time.

In absence of an accurate method for estimating the transmitter's power, based on location and RSS measurements of the online receivers, we propose a new method that can achieve high accuracy and low time complexity by learning from the training data. To learn a function that estimates the power of an unknown transmitter, we must have the training database with different values of transmit power. To simplify the data collection, we conduct the training with constant transmit power, $p^T_t = p$, and then scale the RSS values in the training database to emulate transmit power diversity. For example, we can make the power of the transmitter at time t to be p_1 by scaling the measured RSS values as $(z_{i,t} - p + p_1)$. We refer to this idea of scaling the RSS values to emulate a different transmit power as RSS scaling. This RSS scaling is justified because if the power of a transmitter changes by δ dB, the measured RSS at all the receivers would also change by δ dB, when the RF environment is unchanged [5]. Using the scaled RSS values, LLOCUS learns a function, P, that can estimate the online transmitter's power, $\hat{p}^T_{t'}$. We frame the problem of learning P as a regression problem. For identifying features, to be used in P, we note that P must produce $\hat{p}^T_{t'}$ using the measured RSS and locations of the online receivers. Based on this observation, we use the following method for feature extraction. We analyze the average RSS measured by the receivers close to the transmitter. For any transmission, we identify the location of the receiver that measures maximum RSS and consider a circle of radius R_p around it. All the receivers within this circle are considered 'close' to the transmitter. For each training entry, we calculate the average of RSS measured by the receivers close to the transmitter. This local average RSS, denoted by μ_t, carries information about the difference between $p^T_{t'}$ and p. We use this one dimensional feature vector in a support vector machine [24] regressor, with the radial basis kernel, for learning P.

During the online phase, at time t', when a new entry is received by the CC, we compute $\mu_{t'}$ and feed it to P for estimating $\hat{p}^T_{t'}$. Then, using $\hat{p}^T_{t'}$, we apply RSS scaling on the measured RSS values, $\{z_{i,t'}\}$, such that the power of the online transmitter is p. Finally, we feed the scaled online entry to L for predicting the transmitter's location, as shown in Figure 2. In Section 4, our evaluations show that our method has lower power estimation error than other methods described earlier in this section.

Limitation: Transmit power estimation in LLOCUS exploits the underlying assumption that in a crowdsourcing scenario a transmitter will always have a sufficient number of receivers close to it irrespective of the transmitter's location. Thus, if the receivers are unevenly distributed, the above method may not lead to a very accurate estimate of the power of an unknown transmitter.

3.4 Localizing simultaneous transmitters

Until now, we have considered localizing non-simultaneous transmitters, i.e., multiple transmitters can be present in the area, but their transmissions do not overlap in time. However, under certain undesirable situations multiple transmitters can be active simultaneously, e.g., an RF jammer may transmit alongside a legitimate transmitter for the purpose of disrupting communications. Therefore, in LLOCUS, we incorporate the capability of locating simultaneously active transmitters (SATs), as shown in Figure 5. LLOCUS first estimates the number of SATs, and then performs location estimation for each of the identified transmitters. The primary challenge in localizing SATs is that the measured RSS values at the crowdsourcing receivers are non-deterministic summation of the power from the individual transmitters. To address this challenge, LLOCUS treats each of the detected transmitters as non-simultaneous by defining an approximate region of presence around each of them. The approximate region of presence around a transmitter is defined as the region where the RSS measured by each of the sensing nodes is primarily dominated by the power of the transmitter located inside that region of presence. This implies that the approximate region of presence of the SATs must be non-overlapping such that the sensing nodes can be segregated in disjoint subsets, where nodes within a single subset are influenced by only one transmitter. Thus, in our approach we avoid interference at the sensing nodes by appropriately selecting the non-overlapping regions of presence, which is

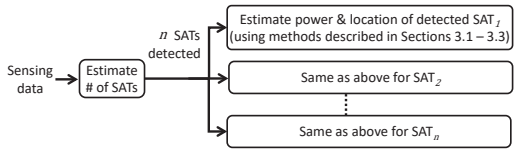


Figure 5: Workflow of LLOCUS for localizing SATs

an important contribution of our work. Once the non-overlapping regions of presence around the SATs are identified, the receivers inside these regions use the methods described in Sections 3.1 - 3.3 for estimating the location and power of the transmitters associated with each of the regions of presence.

For a robust estimation of the number of SATs, we build upon strategies used in SPLOT [19]. For the purpose of explaining our enhancement, we first briefly describe SPLOT. SPLOT first identifies the locations, $\boldsymbol{\theta}_t$, where the measured RSS is above a pre-defined threshold, say z_{ref} dBm (minimum RSS measured by a receiver when there is a transmitter, with transmit power p dBm, in proximity). Next, it identifies a neighborhood for each of the locations $\theta_a \in \boldsymbol{\theta}_t$, defined as a circle of radius R with θ_a at the center. If two locations in $\boldsymbol{\theta}_t$ are within each other's neighborhood, then the location with lower measured RSS is removed from $\boldsymbol{\theta}_t$. The cardinality of $\boldsymbol{\theta}_t$ after this screening procedure is declared as the number of SATs at time t. The neighborhood of each of the locations in the remaining $\boldsymbol{\theta}_t$ is considered to be the approximate region of presence of the nearest active transmitter. Subsequently, for each of the remaining locations in $\boldsymbol{\theta}_t$, SPLOT uses the RSS measurements in its neighborhood to localize the transmitter in proximity, using the matrix inversion method[19]. A reasonable value of R depends on the density of the crowdsourcing nodes and the expected minimum separation between any pair of SATs.

Our improvement over SPLOT is in adaptively selecting the neighborhood for the locations in $\boldsymbol{\theta}_t$. Using a fixed value for R may lead to many false positives (transmitters) in certain scenarios. E.g., when there is only one active online transmitter operating at a power level higher that p, i.e., $p^T_{t'} > p$, using fixed values of R and z_{ref} might result in $|\boldsymbol{\theta}_t| > 1$ after the screening procedure, leading to incorrectly declaring more than one SAT. To overcome this limitation, while screening the initial set of locations in $\boldsymbol{\theta}_t$, for each $\theta_a \in \boldsymbol{\theta}_t$, we estimate the power of the transmitter nearest to θ_a, and based on the estimated transmit power, we choose a value of R_{θ_a} (adaptive value of R) that defines the neighborhood of θ_a. Thus, the screening of $\boldsymbol{\theta}_t$ in LLOCUS is defined by the following procedure. First, we sort the locations in $\boldsymbol{\theta}_t$ based on the descending order of the RSS values measured at these locations. Let the sorted set of locations be $\vec{\boldsymbol{\theta}}_t$. Then, we sequentially process the locations in $\vec{\boldsymbol{\theta}}_t$ starting from the beginning. Assuming that the current location being processed is $\theta_a \in \vec{\boldsymbol{\theta}}_t$, for identifying the neighborhood of θ_a, we apply P (see Section 3.3) at θ_a and estimate the power of the transmitter in proximity. Recall from Section 3.3 that input to P is the local average RSS. Here, we use θ_a as the center for calculating the local average RSS. Since P only uses measured RSS close to θ_a, the estimated power is that of the transmitter in proximity of θ_a. If the estimated transmit power, $\hat{p}^T_{t,a}$, is more than p, then choose R_{θ_a} to be $R \times 10^{\left(p-\hat{p}^T_{t,a}\right)/10\eta}$. Here R_{θ_a} is selected such that if a transmitter transmits at power $\hat{p}^T_{t,a}$ dBm, the RSS value measured by a receiver at a distance of R_{θ_a} meters is approximately z_{ref}. Thus, if $p^T_t > p$, we use a bigger radius and reduce the number of false positives. Next, among all the remaining locations in $\vec{\boldsymbol{\theta}}_t$, that have not been processed yet, ones that are in the neighborhood of θ_a are removed. Hence, our method for estimating the power of online transmitters, described in Section 3.3, enables the adaptive selection of neighborhood in LLOCUS. This completes the processing of θ_a. In the next iteration, we repeat the same procedure for the next element in the remaining $\vec{\boldsymbol{\theta}}_t$. For estimating $p^T_{t,a}$, we use P instead of MAX-RSS, EZ, or TPE-PLM, for reasons explained in Section 3.3.

Limitation: LLOCUS can not detect two SATs if they are very close to each other. When that happens, it is not possible to segregate the set of receivers in disjoint subsets where each subset if primarily affected by only one transmitter.

3.5 Time complexity of predictions

Recall from Section 3.1.4, t_p is the time required to make a prediction, once an online entry arrives at the CC. Assuming $|M| > |R_{t'}|$, $|M| > |I|$, we summarize the main results but leave out the details for brevity. For IFVR, t_p is $O\left(|M| \times |I|\right)$, and for IVVR, t_p is $O(|M|^3)$.

4 EVALUATION AND RESULTS

In the section, we first evaluate LLOCUS for transmitters whose transmissions do not overlap in time. We evaluate LLOCUS for simultaneously active transmitters (SATs) in Section 4.2.

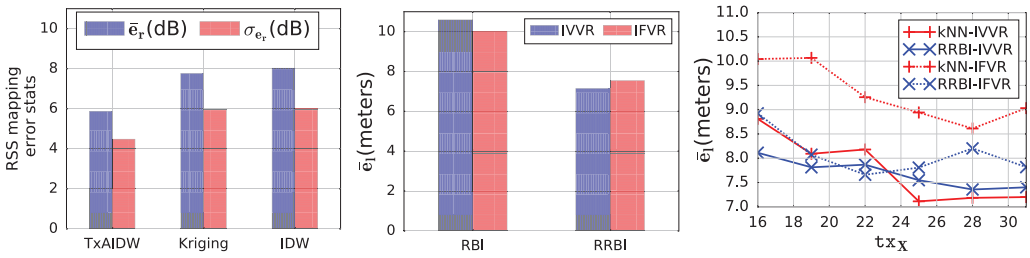
4.1 LLOCUS for single transmitter localization

We evaluate LLOCUS using three different datasets, collected in three experimental settings that span indoor and outdoor environments, with varying areas and density of crowdsourcing nodes.

Description of our experimental settings and datasets:
Dataset A: Outdoor park area: For collecting this dataset, we conduct a measurement experiment in an 85 m × 65 m outdoor area. We use two volunteers for this experiment, one of them as a transmitter (walkie-talkie with transmitter power of 1W) and the other as a receiver (an RTL-SDR attached to a mobile device). To create a crowdsourced environment, the transmitter transmits continuously from one location, and the receiver roams around randomly, inside the park, for 3 minutes. An android application running on the mobile receiver records one reading, tuple of RSS and location (GPS coordinates), per second. This essentially creates a scenario, where a transmission is heard by up to 180 (one reading/second for 3 minutes) crowdsourcing nodes. This procedure is repeated for 44 different randomly chosen transmitter locations.

Dataset B: Indoor hallways: We consider an indoor area of four connected hallways for this dataset. While the whole area is 2500 m^2, all the nodes move only in the four hallways. This dataset is collected by creating an actual crowdsourcing environment with multiple users and using the same equipment as in dataset A. The transmitter moves around the four hallways in a cyclic pattern, and eight receivers move randomly in the four hallways.

Dataset C: Outdoor uneven area: For this experiment, we use another outdoor area of 35 m × 75 m. Similar to dataset B, this dataset is also created by conducting a crowdsourcing experiment with the same equipment. Here, we use static transmitters but mobile receivers. Three users acting as transmitters are placed in the area with at least 25 m distance between any pair, and the



(a) Comparison of RSS interpolation methods (b) Improvement in RRBI over RBI (c) Comparison between RRBI and kNN

Figure 6: Evaluations of the constituent methods in LLOCUS using dataset A. For all the plots $N = 20$ and $|I| = 16$ for IFVR.

transmitters transmit in a round-robin fashion. Eight users acting as receivers roam inside the area randomly.

Creating train-test sets: Among all the transmitter locations in the available data, we randomly select $\texttt{tx}_X$ locations as training transmitter locations. For each training transmitter location, $\texttt{rx}_X$ randomly selected locations among the available receiver locations, are used as training receiver locations. For the test set, we randomly select another set of $\texttt{tx}_Y$ transmitter locations as online transmitter locations, and for each online transmitter location, we randomly select $\texttt{rx}_Y$ receiver locations as online receiver locations.

Evaluation criteria: **(i)** $\bar{e}_l$: Mean of the euclidean distances between the actual locations and the estimated locations of the $\texttt{tx}_Y$ online transmitters. **(ii)** $\bar{e}_p$: Mean of the absolute differences between the actual power and the estimated power of the $\texttt{tx}_Y$ online transmitters. We vary the power of online transmitters randomly between $(-10, 10)$ dB from the power of training transmitters. **(iii)** $\bar{e}_r, \sigma_{e_r}$: Mean and standard deviation of the absolute differences between the actual RSS and the estimated RSS at K target locations where the RSS field is interpolated. Each of the values in $\bar{e}_l, \bar{e}_p$, and $\bar{e}_r$ are further averaged over N iterations, where each iteration corresponds to different partition of the available data in train and test datasets. Each partition produces a random 70% - 30% split of the available data in train and test datasets. The purpose of averaging over N iterations is to smooth out any performance outlier due to a bad train-test split of the available data.

Existing methods for comparison: We compare LLOCUS with the following three well-known non-learning methods.

Maximum likelihood estimation (MLE) [20]: A non-learning method that uses the maximum likelihood estimation to estimate the transmitter's location, based on the PLM and the reported RSS values.

Echolocation (EL) [13]: A non-learning method that creates an ordered sequence of RSS values. Then for every location in the area of interest, it creates an ordered sequence of the distances to the receivers. Finally, it estimates the location of the transmitter as the one that has maximum match between the ordered sequence of RSS and that location's ordered sequence of distances.

Matrix Inversion (MI) [19]: Discretizes the area of interest in voxels, estimates the transmit power field of each voxel based on the PLM and the reported data, and declares the voxel center with maximum transmit power field as the location of the transmitter.

4.1.1 Results. We present the evaluation results in this section.

Selecting constituent methods for LLOCUS: In Figure 6, we evaluate the methods for RSS interpolation and localization presented in Section 3 using dataset A. Figure 6(a) compares the RSS interpolation methods in terms of $\bar{e}_r$, σ_{e_r}. In Figure 6(a), $\texttt{tx}_X \in U(20, 30)$, $\texttt{rx}_X \in U(10, 30)$, and $\texttt{rx}_Y = 5$. We use the notation $a \in U(b, c)$ to denote a is randomly chosen between (b, c). We observe from this figure, both $\bar{e}_r$ and σ_{e_r} for TxAIDW is 2 dB lower than that of kriging and IDW interpolation. This justifies using TxAIDW for RSS interpolation in LLOCUS.

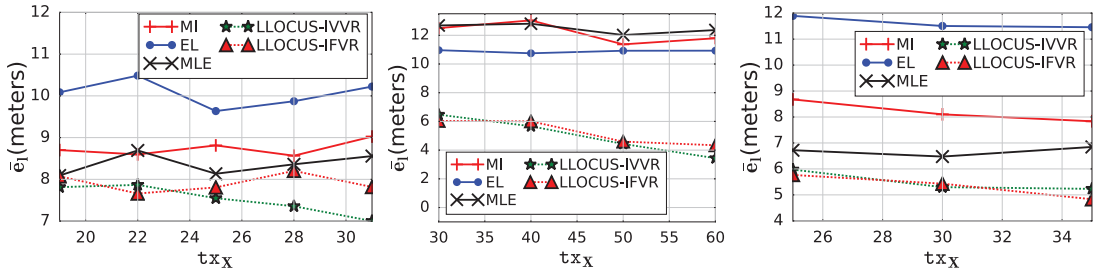
Figure 6(b) shows that $\bar{e}_l$ of our proposed RRBI method is better than the RBI method by around 3 meters, for both IFVR and IVVR. For producing Figure 6(b), we use $\texttt{tx}_X = 30$, $\texttt{rx}_X \in U(15, 30)$, and $\texttt{rx}_Y \in (15, 20)$. Figure 6(c) compares $\bar{e}_l$ for the kNN method and the RRBI method, as $\texttt{tx}_X$ is varied. In this figure, $\texttt{rx}_X \in U(15, 30)$, and $\texttt{rx}_Y \in U(15, 20)$. We observe that, in presence of the *sparsity of training* problem, i.e., when $\texttt{tx}_X$ is low, RRBI outperforms kNN. As $\texttt{tx}_X$ is increased, the performance of kNN gradually converges with that of RRBI. The above observation is true for both feature mapping methods. This implies in areas with a low density of crowdsourcing nodes, RRBI is better than kNN, and in areas with a high density of crowdsourcing nodes, RRBI is as good as kNN, if not better. Thus, based on the observations from Figure 6(b), (c), we select RRBI as the localization method in LLOCUS. All the subsequent evaluations for LLOCUS use TxAIDW for RSS interpolation and RRBI for localization.

LLOCUS vs non-learning methods: In Figure 7, we evaluate $\bar{e}_l$ for LLOCUS and the non-learning methods using our datasets, as $\texttt{tx}_X$ is varied. In Figure 7(a), $\texttt{rx}_X \in U(15, 30)$ and $\texttt{rx}_Y \in U(15, 20)$. Since Figure 7(b), (c) corresponds to actual crowdsourcing setup, readings from all the available receivers are used, i.e., $\texttt{rx}_X = 8$, $\texttt{rx}_Y = 8$. We make the following observations from this figure.

First, LLOCUS always outperforms the non-learning methods, irrespective of the feature mapping method used. Moreover, increasing $\texttt{tx}_X$ reduces $\bar{e}_l$ for LLOCUS due to a higher density of training examples. However, $\texttt{tx}_X$ has no effect on the non-learning methods, since they do not use the training data. Using the right-most data points in Figure 7, the improvement in LLOCUS (with IVVR) over the non-learning methods, in terms of $\bar{e}_l$, for the three datasets are at least 17%, 68%, and 24%, respectively.

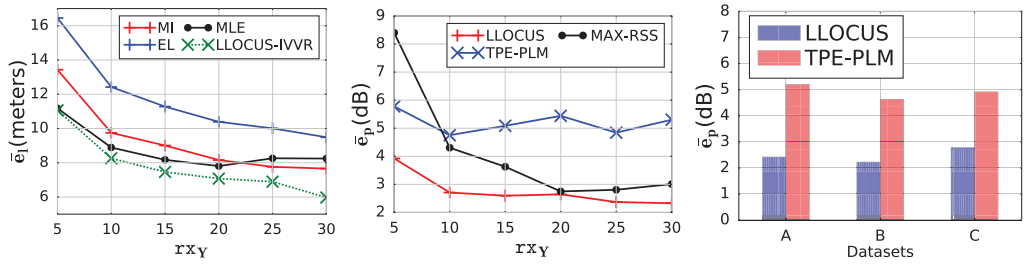
Second, the improvement in LLOCUS, over the non-learning methods, is much more prominent in dataset B. This is primarily due to the fact that, the transmitters in dataset B are always in the hallways. Thus, although the total area in dataset B is 2500 m^2, the spatial sparsity of the training transmitters is much lesser, compared to the other two datasets.

Third, LLOCUS is almost always better with IVVR, than IFVR. For IFVR, we try different values of $|I|$ and identify that, using a value of $|I|$ in the range 16 - 25 provides best performance. In general, the value of $|I|$ can be obtained by applying cross validation.



(a) Dataset A (b) Dataset B (c) Dataset C

Figure 7: Localization error of LLOCUS using dataset A, B, and C. For all the plots $N = 20$ and $|I| = 16$ for IFVR.



(a) Dataset A (b) Dataset A (c) Transmit power estimation error

Figure 8: Miscellaneous evaluations of LLOCUS. For all three plots $N = 20$.

Finally, among the non-learning methods the MLE algorithm performs better than the others, in general. Thus, we use the location estimate produced by MLE to estimate the power of the transmitter when we compare $\bar{e}_p$ for LLOCUS and TPE-PLM, later in Figure 8.

Effect of receiver density: While Figure 7 shows the impact of tx_X on $\bar{e}_l$, the localization error also depends on rx_Y, as shown in Figure 8(a). In Figure 8(a), we use $tx_X = 30$, and $rx_X \in U(10, 30)$. We observe that increasing rx_Y reduces $\bar{e}_l$ for both LLOCUS and the non-learning methods. This is due to the fact that more online receivers provide more RF information about the transmitter's location. However, the rate of improvement in LLOCUS is faster than the non-learning methods, as rx_Y increases.

Estimation of transmit power: Using $tx_X = 30$ and $rx_X \in U(10, 30)$, Figure 8(b) shows that the density of the crowdsourcing nodes also affect the estimation of transmit power. We observe that, increasing rx_Y reduces $\bar{e}_p$ for both LLOCUS and MAX-RSS (described in Section 3.3). However, when rx_Y is low, the MAX-RSS method has a high error. Since the density of the crowdsourcing nodes is not controllable, based on the observations from Figure 8(b), we do not use the MAX-RSS method in LLOCUS.

Figure 8(b) also shows the improvement in LLOCUS, over TPE-PLM (described in Section 3.3), in terms of $\bar{e}_p$. In Figure 8(c), we show that, this improvement exists for all the datasets. In Figure 8(c), we use $tx_X = 30, 60$, and 35 for dataset A, B, and C, respectively. Values of rx_X and rx_Y are chosen as in Figure 7. We observe that $\bar{e}_p$ for LLOCUS is less than 3 dB across all three datasets. Moreover, $\bar{e}_p$ for LLOCUS is 2-3 dB lesser than the $\bar{e}_p$ based on the PLM.

Table 1: Prediction time (msecs) for different algorithms

Dataset	EZ	EL	MLE	LOCUS-IVVR	LLOCUS-IFVR
A	5.5×10^5	61.8	30.0	38.1	0.1
B	6.9×10^5	2.1	3.5	167.1	0.2
C	5.75×10^5	4.5	8.6	58.8	0.1

Figure 8(b), (c) collectively justifies using our method, proposed in Section 3.3, for estimating the power of online transmitters.

Prediction time: In Table 1, we present the prediction time, t_p, for different methods. We observe that t_p for LLOCUS is significantly lower when IFVR is used, compared to the case when IVVR is used. This observation is in congruence with our time complexity analysis in Section 3.5. Next, we observe that, t_p for LLOCUS with IVVR can be much higher compared to the t_p for EL and MLE; whereas with IFVR t_p for LLOCUS is much lower than that of EL and MLE. However, with both IFVR and IVVR, t_p for LLOCUS is less than a second. Finally, we observe that the EZ method, explained in Section 3.3, has a very high t_p.

4.2 LLOCUS for SATs

For this section, we reconsider the indoor area used in dataset B, and extend our experiment to collect data for 2-3 SATs; with at least 25 meters separation between any pair of SATs. Then, we reuse dataset B, and extend the test set by including entries that correspond to 2 - 3 SATs; while keeping the training sets unchanged. We use three evaluation criteria in this section: **(i)** $\bar{r}_m$: Mean of the ratio of the number of online transmitters not detected, and the total number of active online transmitters, be it simultaneous or not, across the test dataset, **(ii)** $\bar{r}_f$: Mean of the ratio of the excess number of online transmitters declared, and the total number of active online transmitters, across the test dataset, **(iii)** Penalized localization error, $\bar{\epsilon}_p$ (meters), calculated in two steps; first, we find the permutation between the estimated locations of the SATs and the actual locations of SATs that results in minimum average

Table 2: Comparison of LLOCUS and SPLOT

Experimental setup	SPLOT			LLOCUS		
	$\bar{r}_m$	$\bar{r}_f$	$\bar{\epsilon}_p(mts.)$	$\bar{r}_m$	$\bar{r}_f$	$\bar{\epsilon}_p(mts.)$
SATs, no power variation	0.04	0.1	10.94	0.05	0.1	6.09
No SAT, power variation	0.01	0.38	12.7	0.01	0.1	4.5

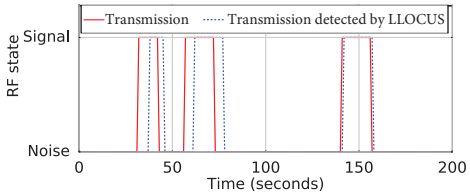


Figure 9: Latency of our LLOCUS prototype

localization error for that online entry. Then, we add a penalty of $g = 5$ meters for each excess or missed transmitter. Since $\bar{\epsilon}_p$ depends on the chosen value of g, it is a relative metric, not an absolute one. The evaluation results are shown in Table 2. Here, we use $\texttt{tx}_X$, $\texttt{rx}_X$, and $\texttt{rx}_Y$ as in Figure 8(c), for dataset B, and $z_{ref} = -4$ dBm.

In the first row of Table 2, we compare the performance of LLOCUS and SPLOT in the presence of SATs, but when no power variation is used in the online phase. We observe that for this case, $\bar{r}_m$ and $\bar{r}_f$ is similar for both the algorithms. However, $\bar{\epsilon}_p$ is lower for LLOCUS, compared to SPLOT, due to the use of our learning-based method. The second row of Table 2 corresponds to the absence of SATs, but when power variation is used in the online phase. In this case, LLOCUS has a much lower $\bar{r}_f$ compared to SPLOT, and this penalizes SPLOT further, in terms of $\bar{\epsilon}_p$. The lower $\bar{r}_f$ in LLOCUS results from the selection of the neighborhoods in an adaptive way.

4.3 Implementation

We build an end-to-end prototype for LLOCUS, that uses a cloud-based compute server as the CC, with a web interface for sending commands to the crowdsourced participants. For the convenience of the participants, instead of the tethered RTL-SDR that we used for collecting dataset A, B, and C, we use a low-power, portable, and untethered SDR, that we have developed [14], with both RF transmit and receive capability, as the crowdsourced client. We build an Android app that runs on the participants' smartphones in the background, and simultaneously maintains a socket with the CC and a Bluetooth connection to the SDR; thus it acts as an intermediary between the CC and the SDRs. LLOCUS can also be used with other mobile devices and in various frequency bands. However, user level access of the RF transceiver, other than the WiFi bands, is not available in current consumer grade mobile devices. We anticipate that future consumer mobile devices will have SDR capabilities to support dynamic spectrum access [22] and hence, will be more amenable to LLOCUS. Using six of our custom built SDRs as crowdsourced nodes, we manually set one SDR to transmit at known instants for a predetermined duration while running LLOCUS. Figure 9 shows that our LLOCUS prototype is able to detect all three transmissions with a latency of up to 5 seconds. The latency in our prototype system is due to two reasons: the mobile devices are loosely synchronized in time, and our current implementation pulls the RF sensing data. We expect that pushing the RF sensing data to the CC and synchronizing the devices would reduce the latency to ≈ 1 second.

5 CONCLUSIONS

We presented LLOCUS, a learning-based system that uses crowdsourcing for localizing mobile transmitters with unknown transmit power. We identified three important challenges in learning from a crowdsourcing system with mobile transmitters and receivers, and described how LLOCUS addresses these challenges. We evaluated LLOCUS using three different datasets and developed a prototype implementation to demonstrate its higher performance and applicability. Our evaluations showed that LLOCUS can learn quickly, as it needs to collect and perform training on a small number of training examples, and LLOCUS can learn well, enough to outperform the non-learning methods. Thus, LLOCUS is deployable in dynamic crowdsourced environments with minimal overhead to a central controller. LLOCUS may not perform very well when the crowdsourcing nodes are not well distributed in the spatial region of interest. Additionally, LLOCUS cannot accurately localize simultaneously active transmitters when they are very close to each other. These limitations will be addressed in future work.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1564287.

REFERENCES

[1] A. Dutta and M. Chiang. 2016. "See Something, Say Something" Crowdsourced Enforcement of Spectrum Policies. *IEEE Trans. on Wireless Communications* (2016).
[2] A. Nika et al. 2016. Empirical Validation of Commodity Spectrum Monitoring. In *ACM SenSys.*
[3] A. Rai et al. 2012. Zee: Zero-Effort Crowdsourcing for Indoor Localization. In *ACM MobiCom.*
[4] C. Bishop. 1995. *Neural networks for pattern recognition.* Oxford university press.
[5] Y. Chen and A. Terzis. 2010. On the Mechanisms and Effects of Calibrating RSSI Measurements for 802.15.4 Radios. In *Springer EWSN.*
[6] A. Chakraborty et al. 2017. Specsense: Crowdsensing for Efficient Querying of Spectrum Occupancy. In *IEEE INFOCOM.*
[7] A. Fragkiadakis et al. 2013. A survey on security threats and detection techniques in cognitive radio networks. *IEEE Communications Surveys & Tutorials* (2013).
[8] A. Iyer et al. 2011. SpecNet: Spectrum Sensing Sans Frontieres. In *USENIX NSDI.*
[9] D. Pfammatter et al. 2015. A Software-defined Sensor Architecture for Large-scale Wideband Spectrum Monitoring. In *ACM IPSN.*
[10] D. Yang et al. 2012. Crowdsourcing to Smartphones: Incentive Mechanism Design for Mobile Phone Sensing. In *ACM MobiCom.*
[11] H. Singh et al. 2018. Privacy Enabled Crowdsourced Transmitter Localization Using Adjusted Measurements. In *IEEE PAC.*
[12] K. Chintalapudi et al. 2010. Indoor Localization Without the Pain. In *ACM MobiCom.*
[13] K. Yedavalli et al. 2005. Ecolocation: A Sequence Based Technique for RF Localization in Wireless Sensor Networks. In *IEEE IPSN.*
[14] P. Smith et al. 2019. Sitara: Spectrum Measurement Goes Mobile Through Crowdsourcing. In *IEEE MASS.*
[15] S. Liu et al. 2009. Non-interactive Localization of Cognitive Radios Based on Dynamic Signal Strength Mapping. In *IEEE WONS.*
[16] S. Sorour et al. 2012. RSS Based Indoor Localization with Limited Deployment Load. In *IEEE GLOBECOM.*
[17] H. Wang et al. 2012. No Need to War-drive: Unsupervised Indoor Localization. In *ACM MobiSys.*
[18] J. Krumm and J. Platt. 2003. Minimizing Calibration Effort for an Indoor 802.11 Device Location Measurement System. *Microsoft Research, November* (2003).
[19] M. Khaledi et al. 2017. Simultaneous Power-Based Localization of Transmitters for Crowdsourced Spectrum Monitoring. In *ACM MobiCom.*
[20] N. Patwari et al. 2003. Relative Location Estimation in Wireless Sensor Networks. *IEEE Trans. on Signal Processing* (2003).
[21] P. Bahl and V. Padmanabhan. 2000. RADAR: An In-building RF-based User Location and Tracking System. In *IEEE INFOCOM.*
[22] Q. Zhao and B. Sadler. 2007. A Survey of Dynamic Spectrum Access. *IEEE Signal Processing Magazine* (2007).
[23] T. Rappaport et al. 1996. *Wireless communications: principles and practice.* Prentice hall PTR New Jersey.
[24] S. Shalev-Shwartz and S. Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge university press.
[25] T. Zhang et al. 2014. A Vehicle-based Measurement Framework for Enhancing Whitespace Spectrum Databases. In *ACM MobiCom.*
[26] M. Youssef and A. Agrawala. 2005. The Horus WLAN Location Determination System. In *ACM MobiSys.*