2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems

Fast and Accurate Indoor Localization based on Spatially Hierarchical Classification

Duc A. Tran
Department of Computer Science
University of Massachusetts, Boston, MA 02125
Email: duc.tran@umb.edu

Cuong Pham
Microsoft Corporation
One Microsoft Way, Redmond, WA 98052
Email: cuong.pham@microsoft.com

***Abstract*—Location fingerprinting is a common approach to indoor localization. For good accuracy, the training set of sample fingerprints should be sufficiently large to be well-representative of the environment in terms of both spatial coverage and temporal coverage. As such, the computation required during the positioning phase can be expensive because we have to evaluate each new fingerprint against the training data repeatedly over time. It is desirable, therefore, to optimize computational efficiency, not just localization accuracy. Existing techniques are far from this goal due to their polarization toward one criterion but not both. We propose a substantially better technique based on the novel approach of modeling indoor localization as a classification learning problem where classes form a spatial hierarchy. Its performance is substantiated in our evaluation study.**

I. Introduction

Location information is valuable to a myriad of indoor applications of wireless networks. In a surveillance sensor network deployed in a building, it is crucial to know the location of an incident such as a fire or machine malfunction that has been caught by a sensor. There is an increasing demand for mobile apps providing navigation and other location-based services in hospitals, shopping malls, airport terminals, and campus buildings, to name a few. GPS is the most popular way to get location information but does not work indoors. Consequently, numerous efforts have been made towards alternative localization solutions.

An approach, sans GPS, is to leverage the correlation between received signal strength (RSS) and distance [1]. The distance between a transmitter and a receiver can be estimated based on how strong the received signal is observed; this procedure is called "ranging". If we have a number of reference points (RPs), e.g., Wi-Fi access points [2], FM broadcasting towers [3], or cellular towers [4], we can locate a device by estimating the distances between this device to the RPs based on RSS ranging and then using multilateration to compute the location. RSS ranging, however, is highly inaccurate due to noise interference [1]. Radio propagates differently in different directions due to obstacles such as walls, people, and furniture.

Hence arises the fingerprint approach, which is range-free. This approach consists of two phases: training (offline) and positioning (online). In the offline phase, the area is surveyed to build a map of sample "fingerprints" serving as training data. This map corresponds each surveyed location to a fingerprint which is a vector of RSS values received at this location from a set of RPs. In the online phase, when we need to compute a location in real time, the RSS vector of the device is evaluated against the fingerprint map to find the best location match. Recently, different efforts have suggested the use of other feature modalities, such as signal strength difference [5], sound [6] and geomagnetic field [7], in the fingerprint information. By combining these features where they apply, we can obtain a rich set of discriminative features for the fingerprint.

Having quality training data is crucial to good localization accuracy. For a large area, we should survey many locations to ensure good spatial coverage and at each location collect many fingerprint readings to ensure good temporal coverage (due to the dynamics of the environment over time). If possible and necessary, we can also augment the training data by using techniques such as compressive sensing [8] and semi-supervised learning [9] to utilize fingerprints that are available without location information.

As the training data size can be large, it is as important to improve the efficiency of the positioning phase, not just accuracy. Consider a location service offered to many mobile users in a large indoor environment, where a server has to compute the location for each user. The server can become congested because not only it has to receive fingerprint readings from every user (communication bottleneck) and evaluate this information against the fingerprint map (computation bottleneck), but also the server has to repeat these two tasks almost continuously as fingerprint information changes each time a user moves. Consequently, our focus in this paper is to devise a positioning algorithm that is not only accurate, but fast. This is also highly desirable even in the scenario where the location is computed locally on the user device without sending any data to the server. In this case, suitable for applications that protect user location privacy, the resource limitation of the mobile device precludes expensive computation.

Many fingerprint techniques rely on nearest-neighbor search (kNN) for estimating the location of the device given a new fingerprint [2], [10], [11]. Unfortunately, kNN search remains one of the most expensive operations in high-dimensional databases and so the positioning algorithm based on kNN is too slow due to the large quantities of fingerprint features and fingerprint samples. Therefore, an alternative fingerprint approach is to employ model-based supervised learning to

978-1-4799-6036-1/14 $31.00 © 2014 IEEE
DOI 10.1109/MASS.2014.122

IEEE computer society

Authorized licensed use limited to: INSTITUTO MILITAR DE ENGENHARIA. Downloaded on March 01,2021 at 16:19:31 UTC from IEEE Xplore. Restrictions apply.

compute a functional dependence between a fingerprint and a location. This dependence can be learned during the training phase using a generative classification model (e.g., Bayesian classifiers [12], [13]) or a discriminative classification model (e.g., Artificial Neural Networks (ANN) [14] and Support Vector Machines (SVM) [15]–[17]). The model-based classification approach is computationally more efficient because only a compact learning model of the fingerprint map is evaluated during the positioning phase. Today's techniques adopting this approach, however, have limited localization accuracy when compared to kNN [15], [18].

Our contribution in this paper is a classification-based fingerprint framework that can offer both improved localization accuracy and computational efficiency. Specifically, we model the localization problem as a classification problem where the classes form a spatial hierarchy. This is fundamentally different from the conventional classification-based fingerprint techniques which define classes as (quasi-)disjoint subdivisions resulted from a flat partitioning of the area. In the positioning phase, whereas the conventional approach queries membership on every class, the proposed approach requires asking only a logarithmic fraction of the number of classes thanks to their spatial hierarchy; hence, more efficient. We achieve this efficiency gain with even better localization accuracy, which is substantiated in an evaluation study we have conducted with two datasets, representing a case with rich training samples and the other case with much simpler training samples. In comparison to kNN, this study also finds our approach competitive in terms of accuracy and, especially, many times faster during the positioning phase.

The rest of the paper is organized as follows. §II provides a brief survey of the related work. §III describes our proposed technique in detail. Experimental results are discussed in §IV. The paper is concluded in §V with pointers to our future work.

II. Related Work

GPS-free localization in wireless networks has been a long-studied problem. There exist many techniques to date, which differ in the type of network environment (e.g., RFID [19], sensor networks [20], wireless LANs [13], vehicular ad hoc networks [21]), the modality of information used to infer location (e.g., infrared [22], radio [2], sound [6], geo-magnetic [7], light [23]), or the type of algorithmic method (e.g., range-based [24], range-free [25]). As our work is focused on range-free location fingerprinting, we here discuss techniques representative of this category. A broader survey of wireless localization techniques is presented in [26].

Radar [2] is the world's first Wi-Fi RSS-based indoor positioning system, which demonstrates the viability of using RSS information to locate a wireless device. This system works using a radio map, a lookup table that maps building locations to the corresponding RSS fingerprints empirically observed at these locations. The reference points are the Wi-Fi access points within the user's Wi-Fi range. The radio map is searched to find the closest RSS reading and the corresponding location will be used as the estimate for the user's location. MoteTrack [10] is a location fingerprint technique largely similar to Radar, but aimed at improving the robustness of the system through a decentralized approach to estimating the location. Instead of location computation's being performed at a central back-end server, this task is distributed among the reference nodes. Given a fingerprint at an unknown location, the location can be computed as the centroid of a number of nearest fingerprints in the radio map, using the Manhattan distance for similarity measure (heavier weight may be given to a training fingerprint that is nearer).

Radar and MoteTrack represent the fingerprint approach where kNN is used to determine the location. We can also employ an indirect learning approach to relate a fingerprint to a location. This approach is motivated by the stochastic nature of fingerprint information: fingerprints at the same location may vary over time and due to presence of physical obstacles they may have identical values even at two distant locations. Consequently, it is argued that location estimation may be more accurate if an intrinsic functional dependence between a fingerprint and a location exists and is utilized. A model for this dependence can be represented probabilistically using Bayesian inference [12], [13] or non-probabilistically using Artificial Neural Networks (ANN) [14] or Support Vector Machines (SVM) [15]–[17].

Bayesian-based techniques [12], [27] associate with each location fingerprint a conditional probability, $p(location \mid fingerprint)$. This probability is computed based on knowledge about the signal propagation model, $p(fingerprint \mid location)$, using Bayesian inference. A limitation of these techniques is that the signal propagation model must be given a priori, or the training set must be substantially large for a precise estimation of the fingerprint distribution at each surveyed location.

ANN-based techniques [14] compute a function mapping a fingerprint to a location in the form of a multi-layer perceptron network with a hidden layer. The synapses store parameters called "weights", initially unknown, that manipulate the data during the calculations. For training, the fingerprint values and surveyed locations in the training set serve as input and output of this network. After the training phase, appropriate values are obtained for the weights. Given a new fingerprint, it goes through linear calculations in the neural network with the obtained weights to finally result in a corresponding location at the output. ANN is computationally more efficient than Bayesian because of these simple calculations, whereas Bayesian requires expensive evaluation of the conditional probability distribution over all the surveyed locations.

SVM is a de facto standard classification tool in Machine Learning preferable to ANN for several reasons: (1) while ANN can suffer from multiple local minima, SVM provides a global and unique solution; (2) unlike ANN, the computational complexity of SVM is independent of the dimensionality of the input space (in our case, the number of reference points); and (3) SVM is less prone to overfitting than ANN is. When it comes to location fingerprinting, SVM has been shown to outperform both the Bayesian- and ANN- classifiers [15], [18],

Authorized licensed use limited to: INSTITUTO MILITAR DE ENGENHARIA. Downloaded on March 01,2021 at 16:19:31 UTC from IEEE Xplore. Restrictions apply.

hence recommended as the closest alternative to kNN when applied to location fingerprinting.

In setting up the localization problem as a classification problem, the conventional way is to pre-partition the localization area into a set of regions (e.g., a grid of cells), each serving as a class and a classification tool is used to assign a given fingerprint to one of these classes. If these classes are not chosen properly, both accuracy and efficiency may suffer, explaining why existing efforts applying SVM for fingerprinting offer limited accuracy compared to kNN; the latter is even faster during the positioning phase in the indoor experiment discussed in [15]. A recent study [17] attempts to boost the accuracy of SVM fingerprinting by incorporating a priori information such as an autocorrelation kernel or a complex output. This improvement comes with larger complexity, hence, at the cost of efficiency. We want to devise a more efficient classification-based fingerprint technique without sacrificing its localization accuracy.

III. Proposed Technique

We assume that location fingerprinting is needed for a 2D area, $\mathcal{D} = (0,1] \times (0,1]$ (our technique can easily be tuned to work with 3D). At each location $P \in \mathcal{D}$, its x- and y-coordinates are denoted by (x_P, y_P). A fingerprint is represented as a vector $\vec{f} = (f_1, f_2, ..., f_n)$ where f_i is a value corresponding to a feature of the fingerprint, e.g., RSSI from different Wi-Fi APs or RFID readers, readings from inertial measurement units (accelerometer, gyroscope, magnetometer), and any location-discriminative feature that is available with the device, etc. The sample fingerprint collection, $\mathcal{F} = \{(x_i, y_i, \vec{f_i})\}_{i=1}^{t}$, obtained during the training phase consists of t fingerprints sampled at locations in $\mathcal{D}$ with known coordinates. It is possible that more than one fingerprint may be collected for the same location. In the positioning phase, given a new fingerprint $\vec{f}$, we need to predict the corresponding location. We adopt the model-based classification approach for localization. Our central contribution is in the way we define classes for the classification and how we determine a fingerprint's location based on classification learning on these classes. Any classification tool, SVM, ANN, or Bayesian, can work with our technique.

A. Conventional Classification for Location Fingerprinting

The common way of applying model-based classification to location fingerprinting is to partition the area into a number of regions, each representing a class to feed into a multi-class classifier. Typically, grid partitioning is used and a class is a unit cell of this partition. If the area is partitioned as a $n_X \times n_Y$ grid, there are $n_X n_Y$ classes, $C_{1,1}, ..., C_{i,j}, ..., C_{n_X,n_Y}$, each being a cell of size $1/n_X \times 1/n_Y$:

$$C_{i,j} = \left\{ P \in \mathcal{D} \mid x_P \in \left[\frac{i-1}{n_X}, \frac{i}{n_X} \right] \wedge y_P \in \left[\frac{j-1}{n_Y}, \frac{j}{n_Y} \right] \right\}.$$

Using a multi-class classifier, a fingerprint can be classified into one of these classes and the center of the corresponding cell is the location estimate. Alternatively, instead of estimating both x- and y-coordinates simultaneously, we can estimate them independently. For the x-dimension, the area is partitioned into n_X equally-sized column stripes, each serving as a class; hence, n_X classes, $\{C_1^X, C_2^X, ..., C_{n_X}^X\}$:

$$C_i^X = \left\{ P \in \mathcal{D} \mid x_P \in \left[\frac{i-1}{n_X}, \frac{i}{n_X} \right] \right\}.$$

Using a multi-class classifier, we can classify the x-coordinate into one of these stripes. Similarly, to estimate the y-coordinate, we apply a separate classification procedure where n_Y classes are obtained by partitioning the area into equally-sized row stripes, $\{C_1^Y, C_2^Y, ..., C_{n_Y}^Y\}$,

$$C_j^Y = \left\{ P \in \mathcal{D} \mid Y_P \in \left[\frac{j-1}{n_Y}, \frac{j}{n_Y} \right] \right\}.$$

For the same quantization resolution (n_X, n_Y) (the smallest area a location can be estimated in is of size $1/n_X \times 1/n_Y$), this method uses only $(n_X + n_Y)$ classes, compared to $n_X n_Y$ of the grid method. We refer to these two methods as STRIPE and GRID, respectively.

Both STRIPE and GRID rely on a flat partitioning of the area; the classes represent regions that are disjoint. As such, comes the following dilemma. On the one hand, since each region in the partition is the unit of quantization to locate a new fingerprint, we want a high partition resolution to reduce the quantization error. On the other hand, with these techniques, a high resolution results in not only a large number of classes to be evaluated (negatively affecting efficiency) but also overwhelmingly more negative samples in each class than positive samples (negatively affecting training quality). Although different techniques may differ in how they define the class regions, which are not precisely disjoint squares or stripes (e.g., circles used in [28]), they all are based on flat partitioning and incur the same drawback as just mentioned.

B. Spatially Hierarchical Classification

We propose that classes are formed by a spatial hierarchy of regions, instead of a flat partitioning of regions. For the same quantization resolution (n_X, n_Y), our classes are defined as follows:

- $m_X = n_X - 1$ classes for the x-dimension, $\{C_1^X, C_2^X, ..., C_{m_X}^X\}$, where

$$C_i^X = \left\{ P \in \mathcal{D} \mid x_P \in \left[\frac{i}{n_X}, 1 \right] \right\}$$

- $m_Y = n_Y - 1$ classes for the y-dimension, $\{C_1^Y, C_2^Y, ..., C_{m_Y}^Y\}$, where

$$C_j^Y = \left\{ P \in \mathcal{D} \mid y_P \in \left[\frac{j}{n_Y}, 1 \right] \right\}$$

Visually speaking, each x-class C_i^X represents the region formed by all the locations that lie to the right of the vertical line $x = \frac{i}{n_X}$, while each y-class C_i^Y represents all the locations that lie above the horizontal line $y = \frac{i}{n_Y}$.

Authorized licensed use limited to: INSTITUTO MILITAR DE ENGENHARIA. Downloaded on March 01,2021 at 16:19:31 UTC from IEEE Xplore. Restrictions apply.

Our classes are not disjoint. Specifically, they form the following ordered lists,

$$C_1^X \supset C_2^X \supset ... \supset C_{m_X}^X$$
$$C_1^Y \supset C_2^Y \supset ... \supset C_{m_Y}^Y$$

and, consequently, we can apply a binary search like algorithm to find the best classification for each fingerprint (to be discussed later).

Compared to STRIPE and GRID, given the same quantization granularity, say if each region to approximate a location is a cell of the $n_X \times n_Y$ grid, our technique requires $(n_X - 1)$ classes for the x-dimension and $(n_Y - 1)$ classes for the y-dimension, for a total of $(n_X + n_Y - 2)$ classes. Also, since each class represents a larger region (a range of columns instead of a single column or a range of rows instead of a single row), the distribution of positive and negative samples in the training for each class is more balanced than that in STRIPE and GRID.

1) Training: Given the sample fingerprint collection $\mathcal{F}$ and using a binary classification tool (e.g., SVM, native Bayes, decision trees, etc.), each of the $(n_X + n_Y - 2)$ classes described above is trained separately. As a result, we obtain a learning model that predicts whether an arbitrary fingerprint $\vec{f} = (f_1, f_2, ..., f_n)$ belongs to a given class C,

$$isMember(\vec{f}, C) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } \vec{f} \text{ is predicted to be in } C \\ 0 & \text{otherwise} \end{cases}$$

For example, if SVM is used, this model contains for each class a set of support vectors and their corresponding coefficients, which is used to compute the function $isMember()$. The details of this are omitted due to limited space and we assume that $isMember()$ is given as a result of the training phase.

2) Positioning: Suppose that we need to compute the coordinates, (x_P, y_P), of a device at an unknown location P, for which a fingerprint $\vec{f}$ has been obtained. For this purpose, we compute each coordinate separately as follows:

- To compute x_P, we find the smallest class $C_{i^*}^X$ such that $P \in C_{i^*}^X$; i.e.,

$$i^* = \max \{i \mid isMember(\vec{f}, C_i^X) = 1\}.$$

- To compute y_P, we find the smallest class $C_{j^*}^Y$ such that $P \in C_{j^*}^Y$; i.e.,

$$j^* = \max \{j \mid isMember(\vec{f}, C_j^Y) = 1\}.$$

Because $P \in C_{i^*}^X$ and $P \notin C_{i^*+1}^X$, we have $x_P \in \left[\frac{i^*}{n_X}, \frac{i^*+1}{n_X}\right]$. Similarly, we have $y_P \in \left[\frac{j^*}{n_Y}, \frac{j^*+1}{n_Y}\right]$. Our coordinate estimate for the location P will be

$$x_P \approx \frac{i^* + 0.5}{n_X},\ y_P \approx \frac{j^* + 0.5}{n_Y}.$$

In the case that $C_{i^*}^X$ is not found (i.e., no x-dimension class contains P), then x_P is estimated as $\frac{0.5}{n_X}$. Similarly, in the case $C_{j^*}^Y$ is not found (i.e., no y-dimension class contains P), then y_P is estimated as $\frac{0.5}{n_Y}$.

To find the smallest class $C_{i^*}^X$ we apply binary search on the ordered list $C_1^X \supset C_2^X \supset ... \supset C_{m_X}^X$. If it is known that $P \in C_{m_X/2}^X$, the class in the middle of this ordered list, we only need to search for i^* in the range $[\frac{m_X}{2}, m_X]$; otherwise, we search the range $[1, \frac{m_X}{2} - 1]$. The search repeats recursively for the resultant range, in which the first query is always evaluated with the class in the middle of this range. Consequently, the number of x-dimension classes we need to query is $\log m_X$. Similarly, the number of y-dimension classes we need to query to find $C_{j^*}^Y$ is $\log m_Y$. The total number of queries, therefore, is $\log m_X + \log m_Y = \log (n_X - 1) + \log (n_Y - 1)$. In contrast, the corresponding cost for STRIPE and GRID is $n_X + n_Y$ and $n_X n_Y$, respectively, if the same quantization granularity is used.

If the prediction $isMember()$ in our positioning algorithm is correct for each query, the location error for the x-coordinate is at most $\frac{1/2}{n_X}$ and for the y-coordinate at most $\frac{1/2}{n_Y}$. However, there is a prediction error associated with each class and so the prediction for a class C may state that $P \in C$ whereas the ground truth is $P \notin C$. Consequently, the location error should be larger. A theoretical analysis of this error is presented below.

C. Error Analysis

Let us focus on one dimension, say x-dimension, and analyze the location error along this dimension. For ease of presentation, assume that n_X is of the form $n_X = 2^m$ and so $m_X = 2^m - 1$. The decision sequence is represented by a m-bit binary number $b_1 b_2 ... b_m$, where $b_i = 0$ if P is predicted not belonging to the class evaluated in the i^{th} query and $b_i = 1$ otherwise. For example, if $m = 3$ ($m_X = 7$ classes) and $b_1 b_2 b_3 = 010$, our positioning algorithm predicts that $P \notin C_4^X$, $P \in C_2^X$, and $P \notin C_3^X$, resulting in class C_2^X being returned as the smallest class containing P. Let ϵ be the maximum individual location prediction error probability among all x-dimension classes. In other words, ϵ is the maximum error probability for the prediction function $isMember()$ among all classes, when applied to an arbitrary (training or non-training) fingerprint. Note that this ϵ is not the training accuracy during the training phase.

First, consider the special case that the ground-truth location is in class $C_{m_X}^X$. That means the correct decision sequence is $\underbrace{11...1}_{m}$. However, the actual decision sequence is $b_1 b_2 ... b_m$ and so the corresponding location error is at most

$$\frac{1}{2^m}\left(\overline{\underbrace{11...1}_{m}} - \overline{b_1 b_2 ... b_m} + \frac{1}{2}\right) = 1 - \frac{\overline{b_1 b_2 ... b_m}}{2^m} - \frac{1}{2^{m+1}}$$

where $\overline{b_1 b_2 ... b_m}$ represents the decimal value of the binary number $b_1 b_2 ... b_m$. If there are i errors made in the decision sequence $b_1 b_2 ... b_m$, then i bits in this sequence must be '0' and the other $m - i$ bits must be '1'. The probability for this case to happen is $\binom{m}{i}\epsilon^i(1-\epsilon)^{m-i}$. The expected value of

Authorized licensed use limited to: INSTITUTO MILITAR DE ENGENHARIA. Downloaded on March 01,2021 at 16:19:31 UTC from IEEE Xplore. Restrictions apply.

$\overline{b_1 b_2 ... b_m}$ over all possible choices for $b_1 b_2 ... b_m$ that has i '0' bits is $(1 - \frac{i}{m})(2^m - 1)$. Consequently, the expected value of $\overline{b_1 b_2 ... b_m}$ over all possible choices of $b_1 b_2 ... b_m$ is

$$\begin{aligned}
B &= \sum_{i=0}^{m} \binom{m}{i} \epsilon^i (1-\epsilon)^{m-i} (1 - \frac{i}{m})(2^m - 1) \\
&= (2^m - 1) \left[1 - \sum_{i=0}^{m} \binom{m}{i} \frac{i}{m} \epsilon^i (1-\epsilon)^{m-i} \right] \\
&= (2^m - 1) \left[1 - \sum_{i=1}^{m} \binom{m}{i} \frac{i}{m} \epsilon^i (1-\epsilon)^{m-i} \right] \\
&= (2^m - 1) \left[1 - \sum_{i=1}^{m} \binom{m-1}{i-1} \epsilon^i (1-\epsilon)^{m-i} \right] \\
&= (2^m - 1) \left[1 - \sum_{i=0}^{m-1} \binom{m-1}{i} \epsilon^{i+1} (1-\epsilon)^{m-i-1} \right] \\
&= (2^m - 1) \left[1 - \epsilon \sum_{i=0}^{m-1} \binom{m-1}{i} \epsilon^i (1-\epsilon)^{m-i-1} \right] \\
&= (2^m - 1)(1 - \epsilon).
\end{aligned}$$

The expected location error is therefore bounded by

$$f(m) = 1 - \frac{B}{2^m} - \frac{1}{2^{m+1}} = \frac{1}{2^{m+1}} + \epsilon \left(1 - \frac{1}{2^m} \right).$$

Note that this bound is for the special case where the ground-truth location of P is in class $C^X_{m_X}$. Now we consider the general case, where the ground-truth location may be anywhere in $\mathcal{D}$ with equal probability. We represent our upper bound for the expected location error as a function $g(m)$ of parameter m. Without loss of generality, we assume that $x_P \geq 1/2$.

When $m = 1$, $g(1) = \epsilon \times 3/4 + (1 - \epsilon) \times 1/4 = (1 + 2\epsilon)/4$. For $m \geq 2$, given a decision sequence, there are two possibilities:

- The first bit is 1: In the first membership query, the classifier for $C^X_{m_X/2}$ correctly predicts that P is in this class. In this case, which happens with probability $1 - \epsilon$, the expected error is

$$A_1 = \frac{1}{2} g(m-1).$$

- The first bit is 0: In the first membership query, the classifier for $C^X_{m_X/2}$ incorrectly predicts that P is not in this class. In this case, which happens with probability ϵ, the expected error is

$$\begin{aligned}
A_0 &= \frac{1}{2^m} \left(\frac{1\underbrace{\overline{00...0}}_{m-1} + \underbrace{\overline{11...1}}_{m}}{2} - 0\underbrace{\overline{11...1}}_{m-1} \right) + \frac{1}{2} f(m-1) \\
&= \frac{1}{4} + \frac{1}{2^{m+1}} + \frac{1}{2} f(m-1) \\
&= \frac{1}{4} + \frac{1}{2^{m+1}} + \frac{1}{2} \left[\frac{1}{2^m} + \epsilon \left(1 - \frac{1}{2^{m-1}} \right) \right] \\
&= \frac{1}{4} + \frac{1}{2^m} + \epsilon \left(\frac{1}{2} - \frac{1}{2^m} \right).
\end{aligned}$$

We have the following recurrence relation for function g:

$$\begin{aligned}
g(m) &= (1-\epsilon) A_1 + \epsilon A_0 \\
&= \frac{1-\epsilon}{2} g(m-1) + \epsilon \left[\frac{1}{4} + \frac{1}{2^m} + \epsilon \left(\frac{1}{2} - \frac{1}{2^m} \right) \right] \\
&= \frac{1-\epsilon}{2} g(m-1) + \frac{\epsilon + 2\epsilon^2}{4} + \frac{\epsilon - \epsilon^2}{2^m}
\end{aligned}$$

Let $a = \frac{1-\epsilon}{2}$, $b = \frac{\epsilon + 2\epsilon^2}{4}$, and $c = \epsilon - \epsilon^2$. We have

$$\begin{aligned}
g(m) &= a g(m-1) + b + \frac{c}{2^m} \\
&= ... \\
&= a^{m-1} g(1) + \frac{b(1 - a^{m-1})}{1-a} + c \sum_{i=0}^{m-2} \frac{a^i}{2^{m-i}} \\
&= a^{m-1} g(1) + \frac{b(1 - a^{m-1})}{1-a} + \frac{c}{2^m} \sum_{i=0}^{m-2} (2a)^i \\
&= a^{m-1} g(1) + \frac{b(1 - a^{m-1})}{1-a} + \frac{c(1 - (2a)^{m-1})}{2^m (1 - 2a)}.
\end{aligned}$$

Since $g(1) = (1 + 2\epsilon)/4$, we obtain the following closed form for g:

$$g(m) = \frac{\epsilon(1+2\epsilon)}{2(1+\epsilon)} + \frac{1-\epsilon}{2^m} - \frac{(1-\epsilon)^{m-1}(1-\epsilon)}{2^{m+1}(1+\epsilon)}. \quad (1)$$

Hence, the result below.

Proposition III.1 (Bound on 1-D Location Error)**.** *Assuming the uniform distribution for the user location, the expected location error along a single dimension (e.g., x-dimension or y-dimension) is bounded by*

$$E = \frac{\epsilon(1+2\epsilon)}{2(1+\epsilon)} + \frac{1-\epsilon}{2^m} - \frac{(1-\epsilon)^{m-1}(1-\epsilon)}{2^{m+1}(1+\epsilon)}. \quad (2)$$

Here, $(2^m - 1)$ classes are defined for this dimension and ϵ is the maximal individual prediction error of these classes.

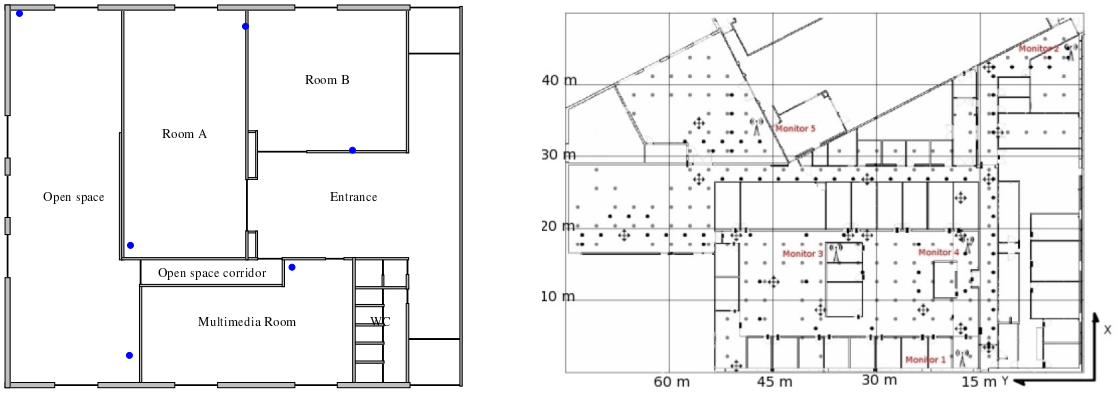
As a corollary, the corresponding bound for the overall 2-D location error is $E_{2D} = \sqrt{E_X^2 + E_Y^2}$ where E_X and E_Y are the bounds for the x-dimension and y-dimension, respectively.

The bound E is a strictly decreasing function of the number of classes and a strictly increasing function of the ϵ prediction error. Specifically,

$$\lim_{\epsilon \to 0} E = \frac{1}{2^{m+1}}, \quad \lim_{m \to \infty} E = \frac{\epsilon(1+2\epsilon)}{2(1+\epsilon)}.$$

However, m and ϵ are not independent of each other. Indeed, for the same training set, if m increases, there are fewer training samples in each class, thus reducing the prediction error ϵ. It is also noted that we do not have a constructive method to compute ϵ. That said, the significance of this proposition is its asymptotical guarantee; the location error can be made small by choosing a sufficiently large m and given this choice of m, having a sufficiently small ϵ (which can be achieved by training on a sufficiently large number of fingerprint samples).

Authorized licensed use limited to: INSTITUTO MILITAR DE ENGENHARIA. Downloaded on March 01,2021 at 16:19:31 UTC from IEEE Xplore. Restrictions apply.



(a) Trento dataset's map: 30m×20m (courtesy of [15])

(b) Colorado dataset's map: 75m×50m (courtesy of [18]).

Fig. 1. Indoor maps of the datasets

IV. Evaluation Study

We evaluate with two datasets:

- Colorado dataset: This dataset is from an indoor experiment used in [18] (University of Colorado). RSSI fingerprints are collected at 179 sample locations in a WLAN with five 802.11 reference points, called "monitors" in their study (Figure 1(b)). An omni-directional transmitter serves as the transmitter and the five monitors as the receivers. Although the signal transmission is omni-directional, fingerprints are collected for four different heading orientations of the transmitter (east/west/south/north). We use a 1% subset of the original dataset as the training collection. There are 8.8 training fingerprints for an average sample location (2.2 on average for each orientation); hence, a total of 1,576 training fingerprints. We test the prediction using a test file containing 77,516 fingerprints.
- Trento dataset: This dataset is from an indoor experiment used in [15] (University of Trento), containing a collection of 257 RSSI fingerprints at 257 sample locations in a WLAN with six Wi-Fi access points (Figure 1(a)). The sample locations are regular-grid points of the floor. Each fingerprint is measured at a sample location by a person carrying a PDA, as a receiver receiving signals from the access points. The PDA always points north. A random half of this collection (128 samples) is used for training and the other half (129 samples) for testing purposes. We use this dataset, which is much smaller than the Colorado dataset, because we want to evaluate the case where there are very few samples (only one per surveyed location).

It is noted that both of these datasets are obtained from areas with signal barriers (walls, obstacles) and holes (non-accessible rooms). Using these datasets, we compare the hierarchical classification based fingerprint approach to the conventional approaches, Stripe and Grid, and the kNN fingerprint approach. Since SVM is arguably the best practice as a binary classification tool, we use it in this evaluation. For ease of presentation, we name the classification-based techniques in comparison by SH-SVM (our technique, SH = Spatially Hierarchical), Grid-SVM, and Stripe-SVM. The parameters for kNN are k and $distance\_type$, where k (1 or 5) represents the number of nearest-neighbor fingerprints used to predict the unknown location and $distance\_type$ ("E" or "M") represents the type of distance used (Euclidean or Manhattan). The parameters for the SVM-based techniques are n_X and n_Y, which represent the quantization region, a cell in the $n_X \times n_Y$ grid. Hence, with SH-SVM, $m_X = (n_X - 1)$ and $m_Y = (n_Y - 1)$. In the results reported here, the range for these two parameters is $n_X \times n_Y \in \{10 \times 10, 20 \times 20, ..., 100 \times 100\}$. For ease of presentation, a notation such as Grid-SVM(20, 20) represents the version of Grid-SVM if the grid is 20×20. Similar notations are also used for Stripe-SVM and SH-SVM.

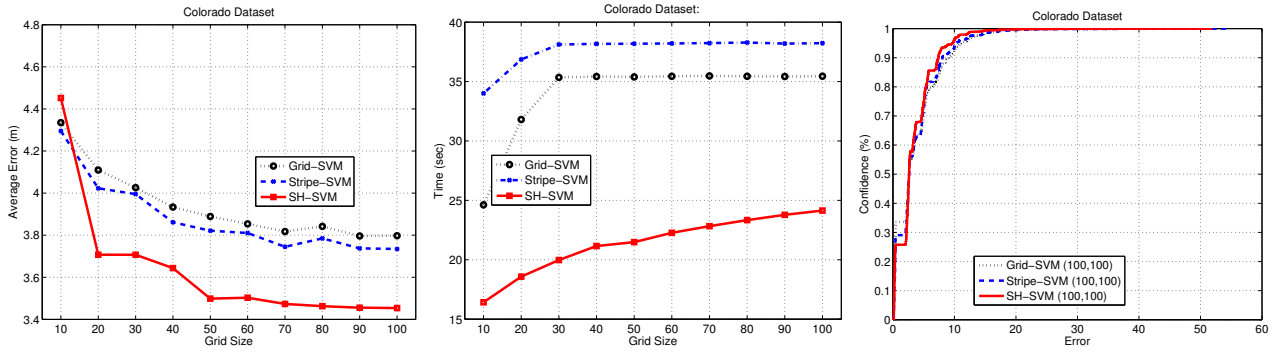
The metrics for comparison are location error ("error" in short), measured as the distance between the predicted location and the ground-truth location, and computational efficiency ("time" in short), measured as the time it takes to make prediction on the given test file. In our evaluation, we use the $libsvm$ software package (http://www.csie.ntu.edu.tw/∼cjlin/libsvm) for SVM training. The programs (training and testing) run on a MacBook Pro laptop running OS X 10.8.1 with 8GB/1600MHz of DDR3 and 2.3 GHz Intel Quadcore i7 processor.

A. Results on the Colorado Dataset

1) SH-SVM vs. Grid-SVM and Stripe-SVM: For this comparison, we compute the error and time metrics of each technique given the same location quantization granularity, i.e., same grid size. The results are shown in Figure 2.

As the quantization resolution increases from 10×10 to 100×100, all three techniques (SH-, Grid-, and Stripe-SVM) improve their location error (Figure 2(a)). On the other hand, the computation time (Figure 2(b)) worsens, which is understandable because the number of classes in each technique increases with the quantization resolution. With resolutions richer than 30×30, this time increase stops for Grid-SVM and Stripe-SVM, which is possibly because the classes become so

Authorized licensed use limited to: INSTITUTO MILITAR DE ENGENHARIA. Downloaded on March 01,2021 at 16:19:31 UTC from IEEE Xplore. Restrictions apply.



(a) Error (b) Time (c) Error distribution

Fig. 2. SH-SVM vs. Grid-SVM vs. Stripe-SVM: Colorado dataset

small that the prediction model is very fast to evaluate.

Grid-SVM and Stripe-SVM are comparable to each other, the former slightly better in terms of time but the latter slightly better in terms of error. They, however, cannot compete with SH-SVM in both metrics. The case $(n_X, n_Y) = (100, 100)$ results in the best error for all the three techniques. In this case, while the average error of Grid-SVM and Stripe-SVM exceeds 3.8m, SH-SVM offers a slightly better error of 3.5m (7+% better); the CDF plot of the error is given in Figure 2(c). Also in the case $(n_X, n_Y) = (100, 100)$, in terms of computation time, it takes more than 35sec time for Grid-SVM and Stripe-SVM to localize all the test fingerprints, whereas it takes 24sec for SH-SVM, which is 30+% faster. Clearly, SH-SVM is the best technique.

2) SH-SVM vs. kNN: To compare to SH-SVM to kNN, first, we analyze kNN in four cases of k and $distance\_type$: 1NN(E), 5NN(E), 1NN(M), and 5NN(M). Here, "E" is for Euclidean and "M" for Manhattan. The average error and time are plotted in Figure 3. It is observed that the Euclidean distance and Manhattan distance provide similar results, time- or error- wise. In either case, a larger k will incur smaller error (Figure 3(a)) and longer time (Figure 3(b)), which is not surprising. The best error (3.5m) is achieved by 5NN(E) and the best time (400sec) is achieved by 1NN(M).

We now compare SH-SVM to the above best-case versions of kNN. In the training phase we have found that the 100×100 quantization resolution should give the best error for SH-SVM, which is indeed substantiated in Figure 2(a). We thus use this resolution for SH-SVM in comparison with kNN. As seen in Figure 3, SH-SVM is clearly better. Compared to 5NN(E) (best-error version of kNN), SH-SVM offers a slightly smaller error (less than 3.5m), yet 76$\times$ faster. Compared to 1NN(M) (best-time version of kNN), SH-SVM is 12% more accurate and 16$\times$ faster.

This study strongly indicates that, for the Colorado dataset, our proposed technique is substantially better than Grid-SVM and Stripe-SVM and kNN-based techniques. Comparing the others, Grid-SVM and Stripe-SVM are also faster than kNN (albeit to a lesser extent) but slightly less accurate.

B. Results on the Trento Dataset

1) SH-SVM vs. Grid-SVM and Stripe-SVM: For the Trento dataset, the results are plotted in Figure 4, which shows that SH-SVM is better than both Grid- and Stripe-SVM in terms of accuracy and, especially in terms of computation time, SH-SVM is more than 3$\times$ faster than the others.

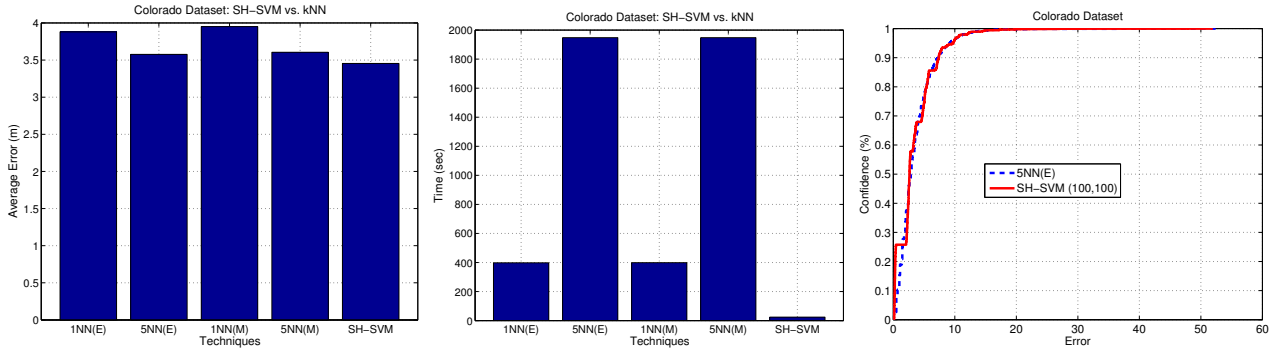
We observe that the performance of these techniques do not seem to improve when the quantization granularity is dense enough; specifically, denser than 20$\times$20 as shown in Figure 4(a) and Figure 4(b). It is noted that the error of Grid-SVM gets worse much more quickly than the others. This supports our argument that having too small a class affects negatively the quality of training, which occurs with Grid-SVM, and that SH-SVM by having larger classes should result in better error. Figure 4(c) plots the CDF of the error in the best scenario for each technique, namely Grid-SVM(10, 10), Stripe-SVM(100,100), and SH-SVM(100, 100), clearly illustrating that SH-SVM is the most accurate. As an example, 80% of the tests has error less than 5m, whereas this percentage is 70% for Stripe-SVM and 60% for Grid-SVM.

2) SH-SVM vs. kNN: We use SH-SVM(100, 100) for this comparison. Similar to the results of the Colorado dataset, SH-SVM outperforms kNN in both error and time metrics, albeit to a lesser extent. The error CDF plotted in Figure 5(c) shows that SH-SVM is more accurate than the best of kNN, which is 5NN(M). On average, 5NN(M) offers a 3.5m error whereas SH-SVM offers 3.2m (Figure 5(a)). It is noted that the Trento floor plan is about 4$\times$ smaller than the Colorado floor plan and so this error gap is more significant compared to the gap with the Colorado dataset. In terms of time (Figure 5(b)), SH-SVM is 1.4$\times$ faster than the fastest version of kNN ("1NN(M)") and more than 6$\times$ faster than the most accurate version of kNN ("5NN(M)").

C. Remarks

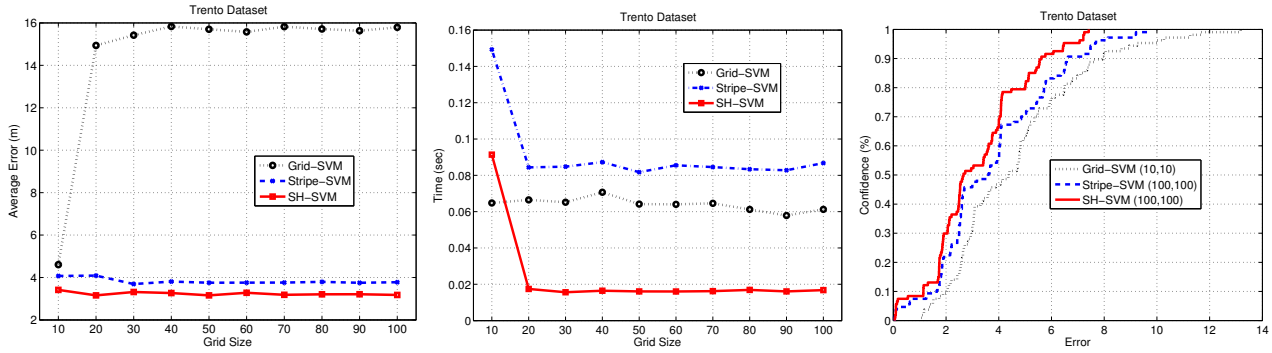
Our study with the two different datasets, small and large, has shown that (1) Grid-SVM and Stripe-SVM are comparable with each other; the former is slightly faster but slightly less accurate; (2) they are substantially less accurate than kNN and

Authorized licensed use limited to: INSTITUTO MILITAR DE ENGENHARIA. Downloaded on March 01,2021 at 16:19:31 UTC from IEEE Xplore. Restrictions apply.



(a) Error (b) Time (c) Error Distribution

Fig. 3. SH-SVM vs. kNN: Colorado dataset



(a) Average error (b) Time (c) Error distribution

Fig. 4. SH-SVM vs. Grid-SVM vs. Stripe-SVM: Trento dataset

worse than SH-SVM in both criteria; (3) kNN is the slowest, extremely slow with large training data; (4) SH-SVM is many times faster than kNN, more accurate with small training data and as accurate with large training data.

V. Conclusions

State-of-the-art solutions to indoor fingerprint-based localization remain either too slow or inaccurate, thus ineffective for large-scale deployments. One can improve accuracy by obtaining more training samples with richer fingerprint features. This, however, does not help mitigate the computation time in the positioning phase; it could even be worse. We pursue an orthogonal approach, whose goal is to make the positioning algorithm faster and more accurate given any fingerprint training set already obtained. Our contribution is essentially modeling of the localization problem as a spatially hierarchical classification problem, where classes represent regions that are spatially hierarchically formed. Our evaluation study has shown that the proposed approach outperforms the conventional classification-based fingerprint approaches in terms of both localization accuracy and computational efficiency. Although this comparison is based on specific datasets, it is strong enough to suggest that using a spatially hierarchical classification approach for fingerprint localization will likely lead to the best result in practice. In the future work, we would like to extend this promising framework to target tracking, specifically how we can apply Hidden Markov Models in a spatially hierarchical way to determine the best trajectory for a moving target in real time.

Acknowledgements

We thank Mauro Brunato [15] and Kevin Bauer [18] for help with usage of their dataset. Our work was supported in part by the NSF award CNS-1116430. Any opinions, findings and conclusions or recommendations expressed in this material are ours and do not necessarily reflect those of the NSF.

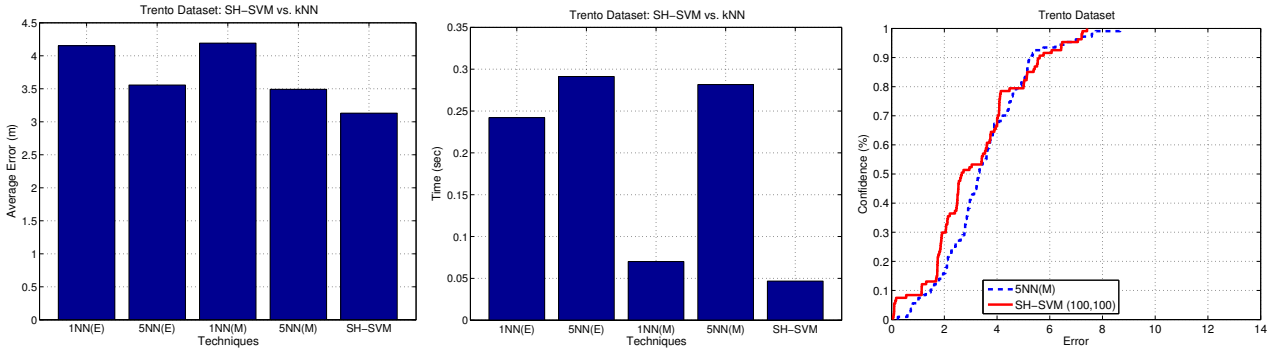
References

[1] K. Whitehouse, C. Karlof, and D. Culler, "A practical evaluation of radio signal strength for ranging-based localization," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 11, no. 1, pp. 41–52, Jan. 2007. [Online]. Available: http://doi.acm.org/10.1145/1234822.1234829

[2] P. Bahl and V. N. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *INFOCOM*, 2000, pp. 775–784.

[3] Y. Chen, D. Lymberopoulos, J. Liu, and B. Priyantha, "Fm-based indoor localization," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, ser. MobiSys '12. New York, NY, USA: ACM, 2012, pp. 169–182. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307653

Authorized licensed use limited to: INSTITUTO MILITAR DE ENGENHARIA. Downloaded on March 01,2021 at 16:19:31 UTC from IEEE Xplore. Restrictions apply.



(a) Error (b) Time (c) Error Distribution

Fig. 5. SH-SVM vs. kNN: Trento dataset

[4] A. Varshavsky, E. de Lara, J. Hightower, A. LaMarca, and V. Otsason, "Gsm indoor localization," *Pervasive Mob. Comput.*, vol. 3, no. 6, pp. 698–720, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.pmcj.2007.07.004

[5] A. M. Hossain, Y. Jin, W.-S. Soh, and H. N. Van, "Ssd: A robust rf location fingerprint addressing mobile devices' heterogeneity," *IEEE Transactions on Mobile Computing*, vol. 12, no. 1, pp. 65–77, 2013.

[6] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik, "Indoor localization without infrastructure using the acoustic background spectrum," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 155–168. [Online]. Available: http://doi.acm.org/10.1145/1999995.2000011

[7] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman, "Indoor location sensing using geo-magnetism," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 141–154. [Online]. Available: http://doi.acm.org/10.1145/1999995.2000010

[8] C. Feng, W. Au, S. Valaee, and Z. Tan, "Received-signal-strength-based indoor positioning using compressive sensing," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 12, pp. 1983–1993, Dec 2012.

[9] D. A. Tran and P. Truong, "Total variation regularization for training of indoor location fingerprints," in *ACM MOBICOM Workshop on Mission-Oriented Wireless Sensor Networking (ACM MiseNet 2013)*, Miami, Sep 2013.

[10] K. Lorincz and M. Welsh, "Motetrack: a robust, decentralized approach to rf-based location tracking," *Personal Ubiquitous Comput.*, vol. 11, no. 6, pp. 489–503, Aug. 2007. [Online]. Available: http://dx.doi.org/10.1007/s00779-006-0095-2

[11] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 269–280. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348578

[12] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen, "A probabilistic approach to WLAN user location estimation," *International Journal of Wireless Information Networks*, vol. 9, no. 3, pp. 155–164, July 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1016003126882

[13] M. Youssef and A. Agrawala, "The horus location determination system," *Wirel. Netw.*, vol. 14, no. 3, pp. 357–374, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1007/s11276-006-0725-7

[14] C. Laoudias, D. G. Eliades, P. Kemppi, C. G. Panayiotou, and M. M. Polycarpou, "Indoor localization using neural networks with location fingerprints," in *Proceedings of the 19th International Conference on Artificial Neural Networks: Part II*, ser. ICANN '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 954–963.

[15] M. Brunato and R. Battiti, "Statistical learning theory for location fingerprinting in wireless lans," *Comput. Netw.*, vol. 47, no. 6, pp. 825–845, Apr. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2004.09.004

[16] C.-L. Wu, L.-C. Fu, and F.-L. Lian, "WLAN location determination in e-home via support vector classification," in *Networking, Sensing and Control, 2004 IEEE International Conference on*, vol. 2, 2004, pp. 1026–1031 Vol.2. [Online]. Available: http://dx.doi.org/10.1109/ICNSC.2004.1297088

[17] C. Figuera, J. L. Rojo-Álvarez, M. Wilby, I. Mora-Jiménez, and A. J. Caamaño, "Advanced support vector machines for 802.11 indoor location," *Signal Process.*, vol. 92, no. 9, pp. 2126–2136, Sep. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.sigpro.2012.01.026

[18] K. Bauer, D. McCoy, E. Anderson, M. Breitenbach, G. Grudic, D. Grunwald, and D. Sicker, "The directional attack on wireless localization: how to spoof your location with a tin can," in *Proceedings of the 28th IEEE conference on Global telecommunications*, ser. GLOBECOM'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 4125–4130. [Online]. Available: http://dl.acm.org/citation.cfm?id=1811982.1812067

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

[20] D. A. Tran and T. Nguyen, "Localization in wireless sensor networks based on support vector machines," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 7, pp. 981–994, July 2008.

[21] A. Boukerche, H. A. B. F. Oliveira, E. F. Nakamura, and A. A. F. Loureiro, "Vehicular ad hoc networks: A new challenge for localization-based systems," *Comput. Commun.*, vol. 31, no. 12, pp. 2838–2849, Jul. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2007.12.004

[22] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The active badge location system," *ACM Trans. Inf. Syst.*, vol. 10, no. 1, pp. 91–102, Jan. 1992. [Online]. Available: http://doi.acm.org/10.1145/128756.128759

[23] W. Zhang and M. Kavehrad, "A 2-d indoor localization system based on visible light led," in *Photonics Society Summer Topical Meeting Series, 2012 IEEE*, july 2012.

[24] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad hoc networks of sensors," in *ACM International Conference on Mobile Computing and Networking (MOBICOM 2001)*, Rome, Italy, July 2001, pp. 166–179.

[25] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-free localization schemes in large scale sensor networks," in *ACM Conference on Mobile Computing and Networking*, 2003.

[26] Y. Liu and Z. Yang, *Location, Localization, and Localizability - Location-awareness Technology for Wireless Networks*. Springer, 2011.

[27] M. Youssef, A. Agrawala, and A. U. Shankar, "Wlan location determination via clustering and probability distributions," in *In IEEE PerCom 2003*, 2003.

[28] X. Nguyen, M. Jordan, and B. Sinopoli, "A kernel-based learning approach to ad hoc sensor network localization," *IEEE Transactions on Sensor Networks*, vol. 1, pp. 134–152, 2005.

Authorized licensed use limited to: INSTITUTO MILITAR DE ENGENHARIA. Downloaded on March 01,2021 at 16:19:31 UTC from IEEE Xplore. Restrictions apply.