Noname manuscript No.
(will be inserted by the editor)

Real-Time WiFi Localization of Heterogeneous Robot Teams Using an Online Random Forest

Balaguer Benjamin · Gorkem Erinc · Stefano Carpin

the date of receipt and acceptance should be inserted later

Abstract In this paper we present a WiFi-based solution to the localization and mapping problem for teams of heterogeneous robots operating in unknown environments. By exploiting wireless signal strengths broadcast from access points, a robot with a large sensor payload creates a WiFi signal map that can then be shared and utilized for localization by sensor-deprived robots. In our approach, WiFi localization is cast as a classification problem. An online clustering algorithm processes incoming WiFi signals that are then incorporated into an online random forest. The algorithm's robustness is increased by a Monte Carlo Localization algorithm whose sensor model exploits the results of the online random forest classification. The proposed algorithm is shown to run in real-time, allowing the robots to operate in completely unknown environments, where a priori information such as a blue-print or the access points' location is unavailable. A comprehensive set of experiments not only compares our approach with other algorithms, but also validates the results across different scenarios covering both indoor and outdoor environments.

1 Introduction

Due to its importance, localization continues to be at the forefront of robotics research and to generate a considerable number of algorithms, each assuming different sensors, robotic platforms, and scenarios. Numerous solutions rely on having accurate but expensive sensors such as a Laser Range Finder (LRF) mounted on one robot. A

Balaguer Benjamin · Gorkem Erinc
Intelligent Computer Programming Labs Inc.
413 Saddle Rock Lane
Rio Vista, CA 94571
E-mail: bbalaguer,gerinc@icplabs.com

Stefano Carpin
School of Engineering
University of California, Merced
5200 North Lake Rd.
Merced, CA 95343
E-mail: scarpin@ucmerced.edu

paradigm shift continuing to gain momentum relies instead on teams of heterogeneous robots including one or more sophisticated platforms cooperating with many minimalistic robots [19]. The simpler robots have the distinct advantage of being inexpensive, allowing to utilize more of them for a given scenario. The limited number of sensors they have onboard however complicates their operation in general and localization in particular.

Following the increasingly popular trend of multi-robot teams consisting in large part of inexpensive robots, we present an accurate localization algorithm that exploits WiFi Access Points (APs). Regardless of how few sensors robots get, thanks to the ever increasing availability of pervasive connectivity, it is safe to assume that they will each have a WiFi card to enable communication with each other or with an external base station. Working with WiFi signal strengths broadcast from APs has its advantages and disadvantages. On one hand, APs are ubiquitously embedded in today's urban environments and have unique identifiers (i.e., MAC addresses), any WiFi cards can measure the APs' signal strengths without connecting to them, and, unlike GPS, they can be exploited indoor or outdoor. On the other hand, WiFi signals suffer from obstructions, non-linear fading, and multipath effects, making them notoriously difficult to model. The WiFi localization problem has seen some limited success over the last decade, but the lack of comparisons between solutions and strict constraints imposed by previously-published algorithms render its application to some scenarios inadequate. In particular, most approaches rely on an offline stage requiring earlier access to the environment for the WiFi signals to be sampled [1–3, 20, 25]. Moreover, the sampling process is often time consuming and requires that robots (or humans) stop along the way to collect multiple signal samples at the same location. Although these constraints can be acceptable in some applications (e.g., when the environment is available prior to the robots accessing it), they become an unacceptable limitation in scenarios where robots do not have prior information about the environment in which they will operate. In this paper we solve these outstanding issues and propose a localization algorithm exploiting WiFi signals that is faster and more accurate than previously-published algorithms, as demonstrated by our comparative study in Section 5.1. Our contributions are the following:

- we propose a clustering algorithm that allows the acquisition of the WiFi map to be performed online, without having to stop the robot to collect WiFi readings or having a priori access to the environment;
- we solve the localization problem using WiFi by casting localization as a classification problem that is trained and solved in real-time using the formerly proposed Online Random Forest algorithm [22];
- we contrast our method with six published algorithms, accounting for both real-time and offline performances;
- we develop a Monte Carlo Localization (MCL) algorithm built with on top of the Online Random Forest, in order to exploit the information provided by the robots' odometry;
- we establish the power of our WiFi localizer by assessing its strengths across different datasets that cover kilometers of indoor and outdoor environments.

The paper is organized as follows. In Section 2 we highlight previous work on WiFi localization. Our problem statement is described in Section 3. The WiFi localizer algorithm follows, consisting of three potentially independent components, whose results are consequently presented within their respective sections. We present the Online

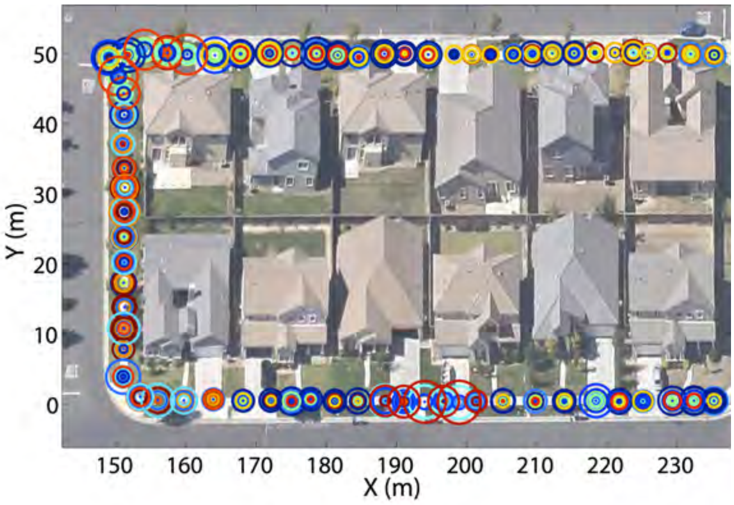

Fig. 1 Illustrative representation of the WiFi readings in an outdoor environment, superimposed on Google Maps. The size of each ring is proportional to the AP's signal strength and each color represents a unique AP. The Cartesian coordinates are derived from a GPS sensor.

Clustering, in Section 4 and the online Random Forest is described in Section 5. Section 6 introduces the MCL in the context of the Online Random Forest, and the paper is brought to a close in Section 7 with final remarks and potential future work.

2 Related Work

The problem of WiFi localization, where signals received from APs are exploited to localize robots, has been solved with different techniques that be categorized into two classes. *Signal modeling* attempts to predict how a wireless signal behaves given a set of parameters describing a particular condition or environment. The model provides an estimation that can be compared to current signal readings to infer a robot's position. *Signal mapping*, an example of which is shown in Figure 1, records WiFi readings that are spatially related to a map. The resulting WiFi map can then be utilized by robots to localize based on current wireless readings. Theoretically, signal modeling often requires a priori information such as the environment's topology, the location of APs, or approximations thereof [16, 26] - information that is not required by signal mapping. Experimentally, Howard et al. have demonstrated that signal modeling yields worse localization results than signal mapping [17]. Electromagnetic signals are distorted due to "typical wave phenomena like diffraction, scattering, reflection, and absorption" [4]. These challenges account for the limited availability of practical implementations of algorithms based on this paradigm. We consequently focus on works related to signal mapping. Readers looking for more information on signal modeling are referred to [16].

Signal mapping can be divided into a training and a localization phase. During the training phase, the signal strengths of all APs in range are recorded and linked to a robot's spatial coordinate until the entire environment is covered. The training phase is generally offline and can be acquired by a human instead of a robot [13, 20]. In fact, every cited publication in this section performs the training phase offline, where every data point is collected prior to generating a signal map. Our proposed method is instead online since new training data points are incrementally integrated into the signal map as they are being collected, allowing the map to be immediately available for localization. Given training data, the WiFi localization problem consists in finding a robot's location only utilizing the WiFi signal strengths it receives. Early solutions to WiFi localization consisted in nearest neighbor searches [1] and bag-of-words models for each AP [25]. An analysis of the distribution of signal strength readings suggested that signal strengths are normally distributed [17, 15], paving the way to an abundance of Gaussian-based techniques that attempt to model the inevitable variance in signal strength readings. The first Gaussian-based solution to WiFi localization [17], which we subsequently call the *Gaussian Model*, consists in fitting a Gaussian distribution for each location and each AP, using data acquired during the training phase. Once the Gaussian distributions have been computed, an unknown location described by a new set of observed signal strengths can be determined using the Gaussians' probability density function (PDF). Its simplicity, speed, and good accuracy have made it, even to this day, one of the most popular WiFi localization techniques [17, 20, 3]. Other popular Gaussian-based algorithms for WiFi localization involve Gaussian Processes (GP), whether they are used directly [14, 9], within a Latent Variable Model framework (GP-LVM) [13],

WiFi GraphSLAM [18] was introduced to alleviate weaknesses exhibited by GP-LVM such as its slow speed and the requirement of signature uniqueness. We note that, although promising, GP, GP-LVM, and WiFi GraphSLAM require a long parameter optimization step that make them unsuited to be used in our scenario where real-time operation is required and computational power limited. Other techniques cast WiFi localization as a machine learning problem solved using Support Vector Machines (SVMs) [24] and Random Forests [2]. The accuracy of these methods can significantly increase by taking into account a robot's inherent temporal and spatial coherence over time using Hidden Markov Models (HMMs) [20] or MCL [3, 17]. In our former work [2], focused exclusively on an offline setting, we compared all of these methods, cross-validated using multiple datasets, and have found that Random Forests supplemented by MCL yields the best results, both in terms of speed and accuracy. As explained in the remainder of the paper, our method also includes Gaussian components, but they are used as part of a two step process, the first of which relies on the classification provided by the online random forest. Gaussian kernels are introduced as a second step to implement the measurement model of a particle filter that improves the localization performance by enforcing temporal coherence – an aspect missing in the classification step.

3 System Setup and Problem Definition

We consider a scenario where the goal of an heterogeneous robot team is to fully explore an unknown environment. The team is comprised of R robots, with R_l "sensor-deprived" robots and R_m "sensor-full" robots under the conditions that $R_l + R_m = R$ and $R_l >> R_m > 0$. For convenience, we will refer to the "sensor-deprived" and "sensor-full" robots as *localizers* and *mappers*, respectively. Every robot can compute odometry and possesses a WiFi card. The *mappers* are the only robots capable of localizing by traditional means (e.g., SLAM, GPS), whereas the *localizers* have no additional sensors. As the robots independently explore the environment, the *mappers* incrementally build their own WiFi maps, which are transferred to the *localizers* whenever they encounter each other. Once they have received a WiFi map, *localizers* can evidently transfer it to other *localizers*. In essence, the *mappers* are responsible for mapping the environment in real-time, the information of which is relayed to the *localizers* who are then able to localize within the region explored by the *mappers*. Evidently, the viability of this approach rests on the assumption that *mappers* and *localizers* operate in the same environment and are then in the range of the same set (or at least subset) of APs. In terms of computational requirements, the *mappers* need to both localize themselves using SLAM algorithms and update the WiFi map. *Localizers* just need to localize themselves using the WiFi map produced by the *mappers*. As described in the following, this WiFi localization relies on a particle filter that exploits results of a random forest. Hence, coherently with our setup, localizer robots are tasked with less demanding computational tasks.

It is worthwhile to note that the environment is completely unknown to the team of robots before they start exploring. At some point in time during the mission, however, the environment will be discovered (partially, at least) by the *mappers*. As long as the *mappers* are capable of deducing spatial coordinates, we do not impose restrictions on how they discover the environments - a fact highlighted by our experiments that use GPS and SLAM to acquire the coordinates. The *localizers*, that do not have adequate sensors to discover the environment by themselves or use the *mappers*' data directly, will exploit WiFi to localize within the *mappers*' knowledge space. In this paper we focus our discussion on the WiFi map building and localization when a single WiFi map has been shared. More complex scenarios involving multiple *mappers* sharing and merging their individual signal maps are left for future research.

The data for the WiFi map is acquired and represented as follows. The ith observation $Z_i = [z_i^1, z_i^2, \ldots, z_i^a]$ is acquired using a WiFi card, where z_i^k stands for the signal strength received from the k-th AP and a denotes the total number of APs seen throughout the environment. Although we measure the signal strengths z_i^k as the number of decibels relative to one milliwatt (dBm), other measures such as the signal-to-noise ratio have been shown to work well for WiFi localization [2,20]. Since we do not know in advance a, the total number of APs in the environment, the size of the observation vector Z_i is dynamically adjusted as the robot explores new regions of the environment and discovers previously unseen APs. For those APs that cannot be seen from certain locations, we set $z_i^k = -100$ for any AP k that was not perceived within observation Z_i. A *mapper* links each observation Z_i to a Cartesian location $\hat{C}_i = [X_i, Y_i]$ that is inferred using a traditional localization process (e.g., SLAM, GPS), resulting in the pair

$$\hat{A}_i = \left\{\hat{C}_i, Z_i\right\}. \tag{1}$$

As the *mapper* explores the environment over time, more $\hat{A}_i$ pairs are collected, the collection of which is written

$$\hat{T} = \bigcup_{i=1}^{m} \hat{A}_i$$

where m is the total number of Cartesian-observation pairs. Figure 1 provides a visualization of $\hat{T}$ acquired in an outdoor environment. It additionally provides a qualitative indication that it is possible to differentiate between different locations by only considering APs' signal strengths.

Remark: although temporal variability, signal distortion, multi-path effects, and other noise sources are known problems associated with WiFi signals, these challenges are implicitely mitigated when the robots operate in environments covered by a large number of APs. More specifically, and as will be shown in the experiments, localization accuracy is directly proportional to AP density and at least 6 APs must be seen to achieve GPS-accuracy (i.e., 3 meters or less). In our real-world experiments (see Section 5.2), the inherent noise encompassed in WiFi signal measurements did not negatively impact the performance of the algorithms presented in the following. A comprehensive sensitivity analysis of this aspect is however beyond the scope of this paper.

4 Online Clustering

In an attempt to model the noise of the WiFi readings and, consequently, improve classification accuracy, we group WiFi readings that are spatially close together. Specifically, we partition the m observations using their Cartesian coordinates $\hat{\mathbf{C}} = \{\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_m\}$ into p clusters $\mathbf{C} = \{C_1, C_2, \ldots, C_p\}$, where $p < m$ and C_i represents the i-th cluster's center. Striving for an online algorithm, every time a cluster is created its label i and centroid C_i are utilized to update the Online Random Forest described in the next section. The clustering needs to be performed online since new Cartesian-observation pairs $\hat{A}_i$ are collected over time as a *mapper* explores the environment. These new observations need to be clustered without altering the cluster assignments of existing observations that have already been used by the Online Random Forest. Hence, we need an online clustering algorithm that makes hard decisions with respect to the clusters' boundaries. In other words, once a boundary is set between two clusters, it cannot be modified and the cluster is *fixed.* Two parameters are crucial for successful localization: p, the total number of clusters, and s_i, the number of observations within each cluster. Due to the fact that the number of clusters is proportional to the size of the explored area and the WiFi readings' density, it is not possible to determine the value of p until the whole environment is explored (since it would contradict with our goal of creating an online clustering algorithm). The number of clusters implicitly defines the average distance between the clusters' centers, which has an impact on the localization accuracy that we analyze in Section 5.2. Hence, instead of employing a sequential k-means clustering algorithm that requires the number of clusters to be set before the clustering process starts, our online clustering algorithm focuses on the maximum radius g_{max} of any cluster, which can be set based on the desired localization resolution. The number of observations per cluster s_i encompasses an important tradeoff between accurate noise modeling and mapping time. The value $s_i = 3$ has been chosen based on the experience we gained in [2].

The design of our Online Clustering algorithm takes into account these parameters and is built around the idea of growing multiple clusters until they encompass enough observations, in which case the clusters are *fixed.* Consequently, each cluster can be in one of two states describing whether the cluster's boundary is *fixed* or *growing.* For each new Cartesian-observation pair $\hat{A}_i$, we identify the centroid C_c of its closest cluster A_c and compute the Euclidean distance g between $\hat{C}_i$ and C_c. There are two cases to take into account, depending on the cluster's state. If cluster A_c is *fixed* and the distance g is less than a pre-defined maximum radius g_{max}, then the observation Z_i is directly added to cluster A_c. Otherwise (if $g \geq g_{max}$), a new *growing* cluster is started with observation Z_i. If cluster A_c is *growing* the observation Z_i is added to A_c. The *growing* clusters are converted to *fixed* clusters by performing a binary split on them. *Growing* clusters are split using the k-means algorithm with $k = 2$, resulting into two clusters A'

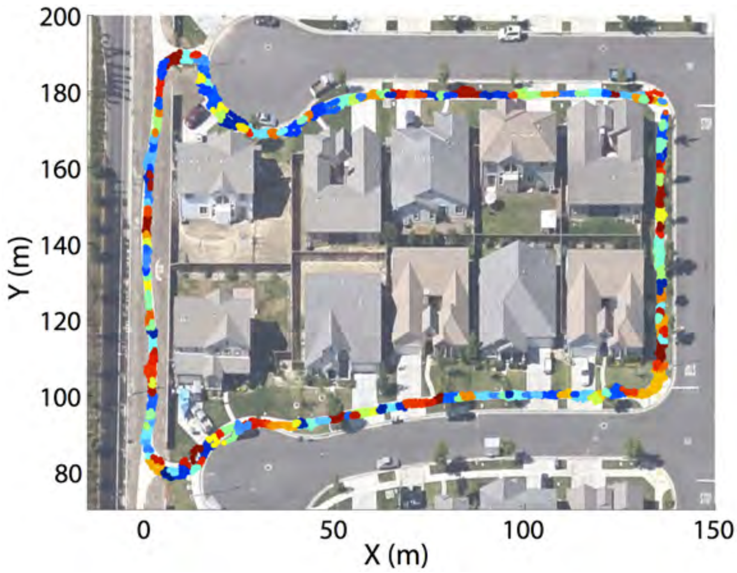

Fig. 2 Illustrative representation of the clustering algorithm in an outdoor environment, superimposed on Google Map. The different clusters are represented by different colors and each encompass a varying number of WiFi observations.

and A''. Both new clusters are *fixed* if they each encompass at least s_{min} observations within a radius of g_{max} from the cluster's center. When either of those conditions fail, the new clusters A' and A'' are ignored and the original cluster A_c keeps *growing*. This strategy guarantees that each *fixed* cluster will have at least a minimum number of observations per cluster s_{min}, which is set to 3 based on the results of [2]. Algorithm 1 summarizes the clustering strategy we propose. The input parameter $\hat{A}_i$ is the new sensor reading to integrate as defined in Eq. 1. $\mathbf{A}$ is the current collection of clusters where each cluster A_i is defined as in Eq. 2.

Algorithm 1 Clustering($\hat{A}_i = \{\hat{C}_i, Z_i\}, \mathbf{A} = \{A_1, A_2, \ldots, A_p\}$)

Let C_c be the centroid closest to $\hat{C}_i$ and A_c the corresponding cluster
$g \leftarrow ||C_c - \hat{C}_i||_2$
if FixedCluster(A_c) **then**
  if $g \leq g_{max}$ **then**
    Add Z_i to cluster A_c
  else
    Create new growing cluster $A_{p+i} = \hat{A}_i$
    Add A_{p+i} to $\mathbf{A}$
if GrowingCluster(A_c) **then**
  Add Z_i to cluster A_c
  Split cluster A_c into fixed clusters A' and A''
  if $|A'| > s_{min}$ **and** $|A''| > s_{min}$ **and** $radius(A') < g_{max}$ **and** $radius(A'') < g_{max}$ **then**
    Remove A_c from $\mathbf{A}$
    Add A' and A'' to $\mathbf{A}$
  else
    Discard A' and A''

For each cluster, we store the Cartesian coordinate of the cluster's center C_i along with all of the observations located inside the cluster into a set A_i

$$A_i = \left\{ C_i, Z_i^1, Z_i^2, \ldots, Z_i^{s_i} \right\} \tag{2}$$

where s_i is the total number of observations inside cluster i, i.e., s_i observations are made for each cluster i, whose center is C_i. It is worthwhile to note that the number of observations in one cluster can be different than the number of observations in another cluster, i.e., s_i is not necessarily equal to s_j $\forall_{i,j} : i \neq j$. Finally, the WiFi map can be represented by a set T of clusters and their observations,

$$T = \bigcup_{i=1}^{p} A_i.$$

Figure 2 shows a visual representation of the resulting clusters for a path followed in an outdoor environment.

By analyzing our experimental online clustering data, we have made the following two observations. First, signal strength readings are consistent across robots with different WiFi cards, a fact corroborated by [17] that implies sharing WiFi data among robots equipped with different hardware is possible. Second, signal strength measurements taken at different days, times, or conditions were not significantly different, indicating that WiFi data acquired a long time ago (i.e., months apart) can still be exploited by robots operating in the same environment.

5 Online Random Forest

5.1 Description

We cast WiFi localization as a classification problem. Our objective is to determine a function $f : Z \rightarrow C$ starting from the training data T acquired by the *mapper*. The function f takes a new observation $Z = [z^1, \ldots, z^a]$ from a *localizer* and returns the Cartesian coordinate C corresponding to observation Z. Many different machine learning techniques have been introduced to compute f from T, but the Random Forest (RF) [5] has been shown to provide the best results for WiFi localization [2]. The leading limitation of the RF algorithm is due to the fact that the entire training data set T must be available a priori, an assumption that does not hold for real-time WiFi localization. Similarly to [22], we modify the RF algorithm, creating an Online Random Forest (ORF) that learns from incremental updates to the training data T, in an online fashion. To the best of our knowledge, this is the first successful use of ORF to learn WiFi maps and solve the robot localization problem. We subsequently describe our ORF algorithm with respect to its offline counter-part. Specifically, an RF is an ensemble of F decision trees built to reduce over-fitting behaviors often observed in single decision trees. Each tree f is created from a subset T' of the full training data T (selected randomly with replacement). Every time a new node in a tree is created, a random set D of decision functions $d(Z) > \theta$ are created, the best of which is used to split the node into left and right children. The definition of "best" varies from one implementation to another, with the most popular partitioning criteria being the Gini index, the twoing rule, and the information gain. We empirically found no distinction between the criteria in terms of localization accuracy and arbitrarily use the Gini index.

The process of generating decisions is the same for the RF and ORF. Since we do not have access to the full training data to create our trees, we update them, as shown in Algorithm 2, every time we receive a new observation-cluster pair (Z, i), with i being a cluster number that is in a *fixed* state.

Algorithm 2 ORF-Update(Z, i)

1: **for** $f \leftarrow 1$ to F **do**
2:     **for** $k \leftarrow 1$ to $P = Poisson(1)$ **do**
3:         l = navigateToLeaf(Z)
4:         updateLeaf(l, Z, i)
5:         **if** shouldSplit(l) **then**
6:             $\arg\max_{d \in D}$ $\triangle$Gini(l, d)
7:             createChildren(l, d)
8:     **if** $P \leftarrow 0$ **then**
9:         $OOBE_t \leftarrow$ updateOOBE(t, Z, i)
10:     **if** x drawn from Bern($OOBE_t$) **then**
11:         rebuildTree(t)

Algorithm 3 shouldSplit(l)

1: **if** diffClusters(l) < 2 **then**
2:     **return** *false*
3: **if** $\triangle$Gini(l, d) $< \alpha \quad \forall d \in D$ **then**
4:     **return** *false*
5: **return** *true*

Each of the F trees is individually updated (line 1) and a Poisson distribution with parameter $\lambda = 1$ models the incremental arrival of new data (line 2). The Poisson distribution essentially mimics the random selection with replacement of the RF's training data. The integer P is drawn from the Poisson distribution and dictates the number of times that the tree will take into account the observation-cluster pair (Z, i). P can be 0 or greater than 1, mimicking an observation that was not sampled or sampled multiple times, respectively. The choice of a Poisson distribution comes from Oza et al. [21] who proved convergence with respect to the offline RF. Since the nodes do not have access to all of the training data, we use observation Z to navigate through the tree, eventually reaching a leaf l (line 3). We add the new observation-cluster pair (Z, i) to leaf l (line 4). So far, the algorithm simply updates the amount of data encompassed by each node of each tree in the ORF. We decide when the leaves should be expanded using the *shouldSplit* function (line 5), whose pseudo-code is given in Algorithm 3. When choosing whether or not to split a node, we take into account two parameters (Algorithm 3). First, the leaf needs to possess observations representing at least two different clusters. Since we know that the clusters will come in sequentially from the Online Clustering algorithm, we want to make sure that we do not split a node whose data only account for one cluster. Second, we only want to split the node if the splitting criterion gain with respect to a decision d, $\triangle$Gini(l, d), has increased significantly. The extent to which the gain should increase is dictated by parameter α. If the choice of splitting the node is made (line 5), the best decision $d \in D$ is determined (line 6) and exploited to split the leaf, creating left and right children (line 7). The newly-created children are imparted with their parent's data.

In order to make the ORF more robust, we add the capability of removing and reconstructing bad trees in the forest. This feature adds the capability of unlearning old information that might no longer reflect the overall data distribution. To do so, we exploit the observation-cluster pairs that were not trained as part of the tree, called Out-Of-Bag (OOB) samples. From these samples, which only occur when $P = 0$, we can compute the OOB Error (OOBE) for each tree (line 10). In some sense, the OOBE, which is analogous to computing cross validation errors, supplies a quantitative measure of a tree's performance. By evaluating this measure, we can infer when a tree becomes less useful, in which case we destroy it and rebuild it from scratch using the generic RF tree building algorithm. To decide whether or not a tree should be discarded (line 11), we sample x from a Bernoulli distribution whose parameter p is set to the OOBE. The sample x is either 1 or 0 and indicates if the tree should be discarded. The Bernoulli distribution essentially models the probability of discarding a tree, which is directly proportional to the OOBE. It is important to note that reconstructing one tree in the entire forest does not significantly damage the algorithm's performance, yet allows the forest to adapt over time.

5.2 Results

Since our MCL algorithm will depend on the Online Random Forest, we evaluate the Online Random Forest before describing the MCL algorithm in the next section. More specifically, we compare the Online Random Forest against six previously-published algorithms, namely: Nearest Neighbor Search [1]; Gaussian Model [17,20,3]; Gaussian Processes [14,9]; Support Vector Machine [24]; Decision Tree [6]; and the (offline) Random Forest [5]. Interested readers are directed to the referenced publications for additional details. The reader should also notice that the experimental comparisons presented in our former paper [2], although apparently similar, are different, because they refer to offline situation, whereas we are here considering an online scenario. Since we are focusing on online WiFi map building, we briefly describe each method with respect to its online capabilities. We "convert" the Random Forest to an online version by re-training it from scratch every time a new cluster is created. This is evidently time consuming, as will be shown in this section, but provides a very accurate baseline to compare against. The multi-class SVM can be trained in a one-against-one or one-against-all methodology. Since the one-against-all technique does not allow for online learning (unless training is performed from scratch for every new cluster), we use the one-against-one technique. Assuming we have discovered c clusters that were already trained one-against-one, a new cluster will require training c new SVMs. Evidently, the number of observations grows over time and so does the SVM training time. The Gaussian Model and Nearest Neighbor Search are already capable of running online. Since the Gaussian Model computes a set of Gaussian distributions for each cluster independently of any other clusters, we can simply wait to get a new cluster before computing its distribution. The Nearest Neighbor Search does not have a training phase and is consequently online by default.

We run the experiments on a set of six different datasets, each representing distinctive environments and operating conditions. Table 1 provides detailed information for each of the six datasets. As can be seen from the table, we acquired two datasets, UCM and Merced, representing indoor and outdoor environments. The remaining four datasets, RICE, CMU, USC, and HKUST, were made publicly available by the authors of [20], [8], [17], and [10], respectively. The datasets differ significantly in terms of environment's type (e.g., indoor, outdoor), WiFi map acquisition process (e.g., offline, online), mapping agent (e.g., robot, human), localization technique used to infer the Cartesian coordinates C_i (e.g., SLAM, GPS, manual labeling, laser-based MCL), wireless signal metric (e.g., dBm, SNR), approximate size of the environment covered, total number of clusters (c), average distance between clusters, minimum number of observations per cluster, and total number of APs discovered (a). As a group, these datasets exemplify a wide range of operating conditions and provide the most extensive set of WiFi localization experiments to date. Since this set of experimental results is purely meant to evaluate the Online Random Forest against other algorithms and across different datasets, all of the data presented in this section was acquired by a single robot.

	UCM	**M**erced	**R**ICE	**C**MU	**U**SC	**H**KUST
Source	In-House	In-House	Public	Public	Public	Public
Environment	Indoor	Outdoor	Indoor	Indoor	Indoor	Indoor
Map Acquisition	Offline	Online	Offline	Offline	Online	Offline
Mapping Agent	Robot	Robot	Human	Human	Robot	Human
Localization	SLAM	GPS	Manual	Manual	MCL	Manual
Signal Metric	dBm	dBm	SNR	SNR	dBm	dBm
Length Covered	150m	1.5km	1.5km	300m	100m	100m
No. of Clusters, p	156	720	506	209	120	106
Clusters' Distance	0.89m	1.96m	3.33m	1.58m	0.69m	1.00m
$\min s_i \in A_i$	20	7	100	32	20	25
No. of APs, a	48	231	57	152	4	144

Table 1 Detailed information about the datasets used for the classification experiments. Each column corresponds to a dataset, with each row representing a specific characteristic of the dataset. Note that although the USC dataset was acquired offline, we processed it with our online algorithm.

We start by analyzing each algorithm's accuracy as a function of the number of readings taken at each location, s, a plot of which is shown in Figure 3. The training and classification data are randomly sampled 50 different times from UCM dataset for each experiment in order to remove any potential bias from a single sample. Presented results are averages of those 50 samples and we omit error bars in our graphs since the results' standard deviation were all similar and insignificant. Since our goal is to ultimately be able to map unknown environments in real-time, the data acquisition needs to be efficient. It takes on average 411ms to get signal strength readings from all the APs in range so we want to minimize s. Figures 3 and 4 show the average and the standard deviation of the classification error, respectively. As expected, there is a tradeoff between the algorithms' localization accuracy and the number of readings used during the training phase. Since we want to limit the time it takes to gather the training data, setting the number of readings per location to 3 ($s = 3$) provides a good compromise between speed and accuracy, especially since the graph shows an horizontal asymptote starting at or around that point for the best algorithms. As such, the rest of the results presented in the paper will be performed using 3 readings per location. This value is in accordance with the findings we determined in [2]. All the algorithms, except the Decision Tree, reached an average classification error below 1m with 3 readings per location. Furthermore, the Gaussian based techniques, as expected, performed badly when 1 reading is used per location since in that case the training data does not contain enough variance to model, i.e., $\sigma = 0$.

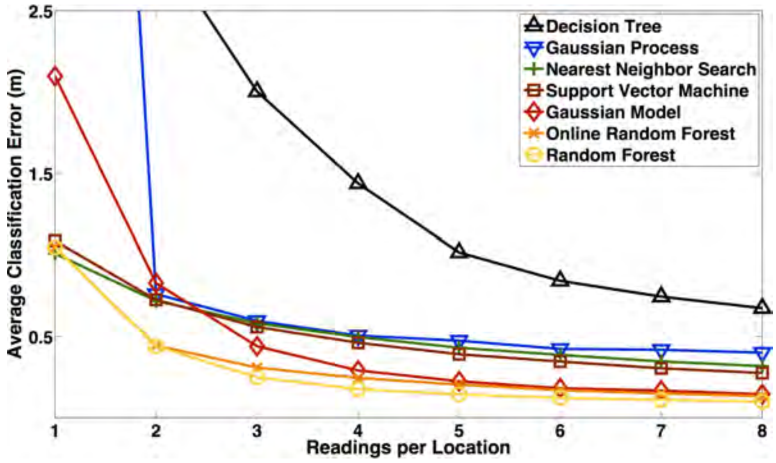

Fig. 3 Average classification error, in meters, for each classification algorithm as a function of increasing number of readings per location.

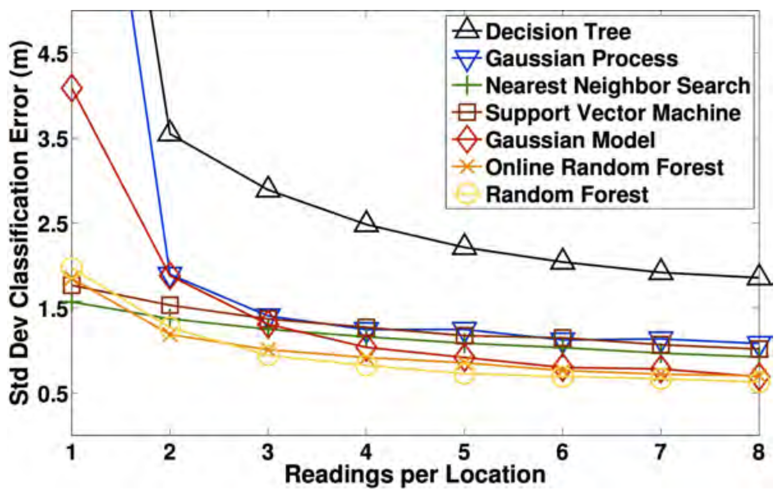

Fig. 4 Standard deviation of the classification error, in meters, for each classification algorithm as a function of increasing number of readings per location.

Figure 5 shows the cumulative probability of classifying a location within the error margin indicated on the x axis. This figure corroborates the findings of Figure 3, showing that the best algorithm is the Random Forest, which had never been exploited in the context of WiFi localization. Moreover, the Random Forest can localize a new observation, Z, to the exact location (i.e., zero margin of error) 88.58% of the time.

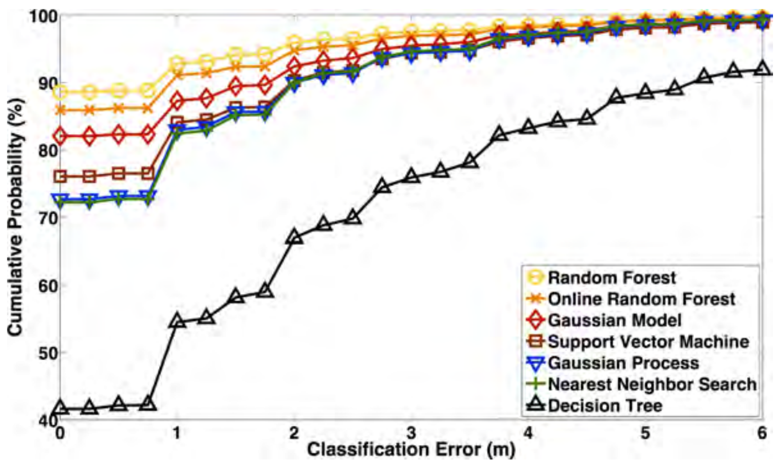

Fig. 5 Cumulative probability of classification with respect to the error margin.

The average classification error of each algorithm and each dataset is shown in Figure 6. As can be seen from the figure, both Random Forests are consistently more accurate than the other algorithms, regardless of the dataset and, consequently, environment used. The Online Random Forest is only marginally worse than its offline counterpart; 7.39% on average. Compared with the other algorithms, the Online Random Forest is, on average, 15.96%, 103.24%, 301.26%, 475.99%, and 1400.71% better than the previously published WiFi localization algorithms (Gaussian Model, Nearest Neighbor Search, Support Vector Machine, Gaussian Process, and Decision Tree, respectively). It is also worth mentioning that the Rice dataset uses signal to interference ratios for their observations as opposed to signal strengths. Although both measures are loosely related, this indicates that the classification algorithms are both general and robust. These important findings indicate that the proposed Online Random Forest is data- and environment-independent and that we are not biased or over-fitting our datasets. The reader should note that for the indoor environments (i.e., UC Merced and Rice), the average classification accuracy is proportional to the average distance between clusters. As seen in Figure 6, the outdoor environment is the most challenging, due to the large distances between APs, prominent multi-path effects, and size of the environment.

The time that each algorithm takes to incrementally add a new cluster into the WiFi map is computed and the resulting training times are shown in Figure 7, where the number of clusters are progressively increased from 20 to 100 as the robot explores the map. It is important to note that 100 clusters accounts for a small map covering an area ranging from 100 to 300 square meters and that is the reason why the training process takes this little time in this example. Figure 7 does not include the Nearest Neighbor Search since it does not require a training phase. Similarly, training time for the Gaussian Process is omitted since it takes too long to train for online applications (i.e., an average of 4.75 minutes for 100 clusters). The figure clearly shows that both the

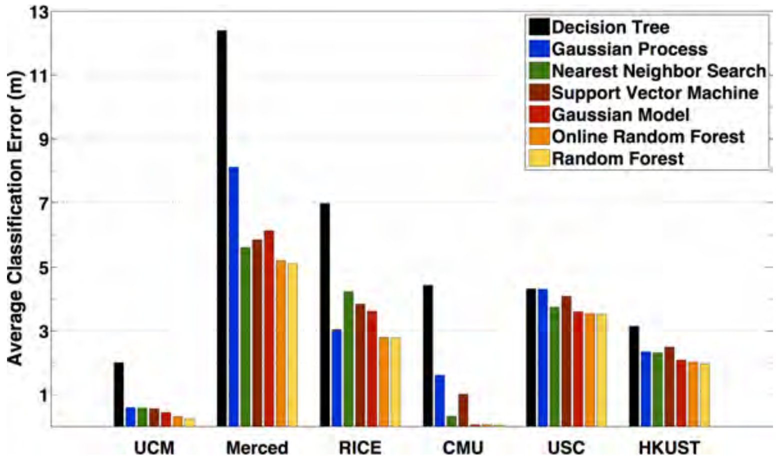

Fig. 6 Average classification error, in meters, for each classification algorithm executed on the UCM, Merced, RICE, CMU, USC, and HKUST datasets.

Random Forest and Support Vector Machine take too much time to train, especially since they grow linearly and exponentially, respectively, as the number of clusters increases. On the other hand, the Gaussian Model and Online Random Forest add new clusters to a trained map in constant time, with speeds of 27.4ms and 2.5ms, respectively.

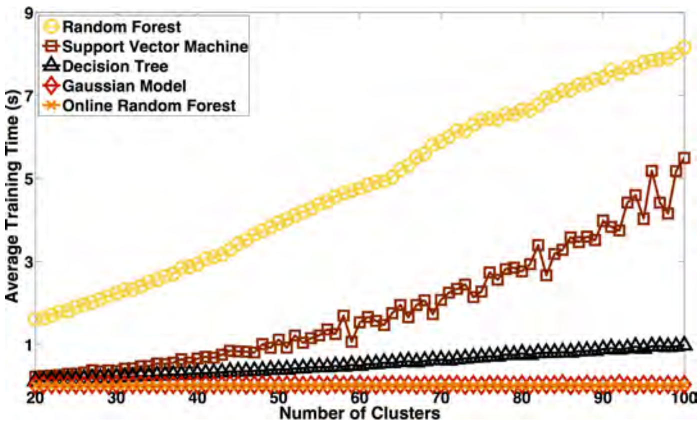

Fig. 7 Average training time, in seconds, as the clusters increase during exploration. The Nearest Neighbor Search is omitted since it does not need training. The Gaussian Process is also omitted since it takes too long: 4.75 minutes on average for 100 clusters.

Not only is the training time important for real-world scenarios, but so is the classification time. Indeed, it does not matter how fast WiFi maps can be created, if they take minutes to be exploited. Figure 8 shows, for each algorithm, the average time to classify 10 observations. Both the Gaussian Process and Support Vector Machine algorithms are omitted due to their slow classification speeds of 3.05s and 9.07s, respectively, for maps containing 100 clusters. The trends for the remaining algorithms are approximately constant, except for the Gaussian Model which is linear. Once again, the Online Random Forest provides the fastest option, along with the Nearest Neighbor Search.

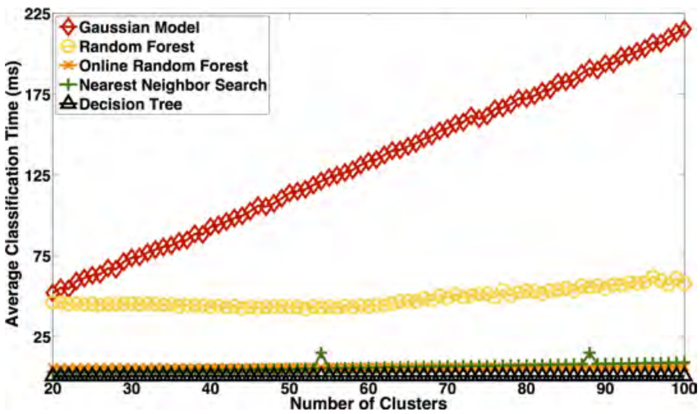

Fig. 8 Average time, in milliseconds, it takes to classify 10 observations, as the number of clusters increases during robot exploration. The Gaussian Process and Support Vector Machine algorithms are omitted because they take too much time relative to the other algorithms (i.e., 3.05 and 9.07 seconds, on average, for a map size of 100 clusters).

6 Monte Carlo Localization

6.1 Description

The Online Random Forest and its experimental results solve a classification problem, where each new observation is classified independently of any other observations. For robot exploration, it is advantageous to exploit the fact that a robot's position and observation should be coherent within small time steps. The WiFi localizer should take into account the robot's previous location or, more precisely, the probability distribution of the robot's previous location. We introduce the robot's temporal and spatial information into our WiFi localizer with a particle filter, whose algorithm comes from the "standard" implementation presented by Thrun et al. [23]. Since we have already shown the Online Random Forest to be very accurate for WiFi localization cast as a classification problem, it will become an integral part of the MCL's measurement model. The robot's motion model, which takes into account the robot's translational and rotational velocities, is exactly the same as in [23] (chapter 4). We implement a new measurement model that takes into account results from the Online Random Forest. Given a particle with Cartesian coordinate C and a new observation Z, the measurement model computes the probability $P(C|Z)$. We generate a probability distribution P by observing that each tree in the Online Random Forest "votes" for one cluster center C_i and the one with the most votes is chosen as the robot's location. The votes can be converted to the robot's probability of being at a particular cluster's center, $P(C_i|Z) = V_i/F$, where V_i is the total number of votes received for cluster i and F is the number of trees in the Online Random Forest. We exploit this information to compute our Probability Density Function (PDF) as a Gaussian Mixture Model (GMM) described in Algorithm 4.

The GMM in Algorithm 4 non-linearly diffuses, in two-dimensional Cartesian space, results acquired from the Online Random Forest. Specifically, we build a GMM composed of one mixture component for each cluster i (line 2). Each mixture component represents a bivariate Gaussian distribution with a mean μ_i corresponding to the ith cluster's center C_i (line 3), a covariance matrix Σ_i (line 4), and a mixture weight ϕ_i (line 5) proportional to the votes V deduced by classifying observation Z with the Online Random Forest (line 1). Line 5 highlights a key difference between the classification and MCL methodologies. By taking the mode of the Online Random Forest's results, the classification discards valuable information that is instead exploited by the MCL. Effectively, given an observation Z, the GMM represents the Online Random Forest's classification confidence across the set of every cluster's center (in two-dimensional space). It is important to note that a GMM is built for each new observation Z. Additionally, the parameter σ^2, which dictates how much the mixture components affect each other, needs to be defined. We have empirically determined that setting σ^2 to the average distance between clusters' centers provided good results for the GMM.

Algorithm 4 Construct-GMM(Z)

1: $V \leftarrow$ Online-Random-Forest-Prediction(Z)
2: **for** $i \leftarrow 1$ to p **do**
3: $\quad \mu_i \leftarrow C_i$
4: $\quad \Sigma_i \leftarrow \sigma^2 I$
5: $\quad \phi_i \leftarrow P(C_i|Z) \leftarrow V_i/F$
6: **return** gmm ←Build-GMM(μ, Σ, ϕ)

With the aforementioned PDF, the measurement model becomes a simple algorithm, whose pseudo-code is presented in Algorithm 5. It highlights the effectiveness of our GMM implementation (line 1), which seamlessly integrates into the MCL's measurement model. Indeed, each particle i (line 2) is assigned a weight proportional to the probability of being at the particle's state given a WiFi observation Z. The particle's weight is retrieved by using the PDF of the GMM (line 3).

Algorithm 5 Measurement-Model(Z)

1: $gmm \leftarrow$ Construct-GMM(Z)
2: **for** $i \leftarrow 1$ to n **do**
3: $\quad i$.weight ← PDF(gmm, i.state(X,Y))

We conclude this subsection noting that although we opted for a particle filter, there are alternative choices that one could explore. For example, one could incorporate the outcome of the WiFi-based Online Random Forest as a measurement model for a different Bayesian estimation method, such as the Extended Kalman Filter (EKF) or the unscented Kalman filter (UKF) [23]. These possible extensions are left for future work.

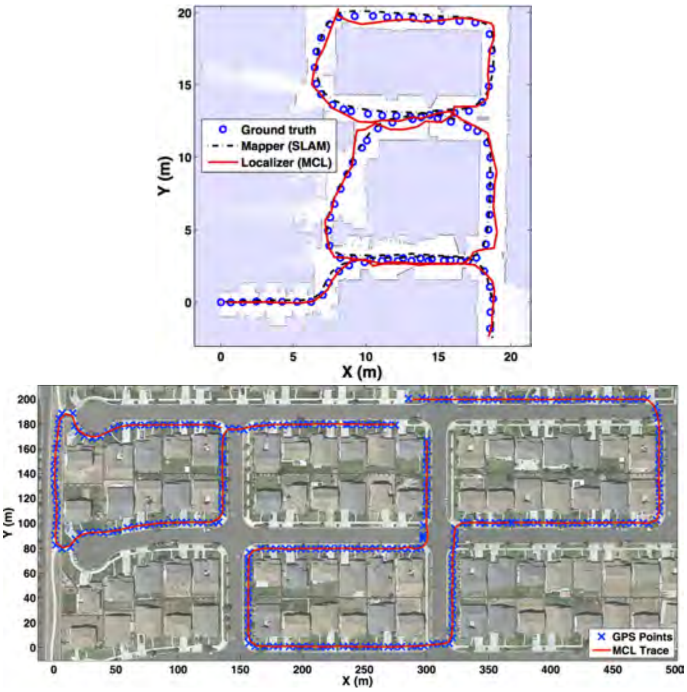

Fig. 9 Experimental results for indoor (top) and outdoor (bottom) scenarios, showing the results of our WiFi localization (red) against ground truth (blue) and SLAM (black, indoor only). For the outdoor scenario, GPS is used as "ground truth".

6.2 Results

In this section, we evaluate our MCL algorithm, running with 5000 particles, on the same indoor UCM and outdoor Merced datasets that were used for the classification experiments (see Section 5.2). Since we do not have physical access to the environments where the other datasets were collected, we cannot use the MCL on the other datasets. Consequently, the Online Random Forest is trained by *mappers* exploring the UCM and Merced datasets. For the indoor UCM environment (first column of Table 1), the *mappers* covered approximately 150 meters, discovered 48 unique APs, and created 156 clusters, each encompassing at least 20 observations and separated, on average, by 0.89 meters. For the outdoor Merced environment (second column of Table 1), the *mappers* covered approximately 1.5 kilometers, discovered 231 unique APs, and created 720 clusters, each encompassing at least 7 observations and separated, on average, by 1.96 meters. The *localizers* explore the same environments and localize strictly exploiting WiFi signal strengths readings within our algorithmic framework. In order to quantitatively analyze the MCL, an LRF and GPS receiver are mounted on the *localizers* for the indoor and outdoor scenario, respectively. Please note that these two sensors are only meant to generate pseudo-ground-truth and are not used as part of the WiFi localization. The indoor and outdoor environments are each explored 10 different times. For each exploration, the MCL is performed 50 times to account for the random nature of the MCL. After each measurement model update, we localize the robot by computing the weighted mean of the particles and compute the error between the MCL and pseudo-ground-truth (LRF SLAM or GPS, depending on the environment). The average error for the 50 trials of the 10 indoor and 10 outdoor explorations are 0.61 meters and 1.02 meters, respectively. These results show the power of our MCL algorithm, since it increases the localization accuracy by 60% and 500%, respectively, when compared to using the Online Random Forest on its own. Figure 9 provides a visualization for 1 indoor and 3 outdoor runs (superimposed on one map). For the outdoor run, the *localizers* were always perfectly estimated on the sidewalk. This outcome is logical, however, since the *mapper* was driven on the sidewalks and the *localizers* use that data to localize.

The contribution of the proposed algorithm comes not only from its accuracy but also from its efficiency, both in terms of training by the *mappers* and localization by the *localizers*. Each time a *mapper* discovers a new Cartesian-observation pair $\hat{A}$, the clustering algorithm runs with complexity $O(\log(p))$, where p is the current number of clusters. When the Cartesian-observation pair is added to a fixed cluster, the *mapper* incorporates it into the Online Random Forest, a process with time complexity $O(Fa\log(m))$, where F is the number of trees in the forest, a is the number of APs discovered (i.e., features), and m is the total number of observations. Comparatively, and since the Random Forest has a time complexity of $O(Fam\log(m))$, the Online Random Forest has strictly smaller order of magnitude than its offline counterpart. To localize, the *localizers* must make a prediction using the Online Random Forest in $O(F\log(m))$ and run the MCL, whose biggest time complexity comes from the measurement model being computed for all n particles in $O(np)$. These asymptotic time complexities translate to real-world efficiency and allow the algorithms to run in real-time on a consumer laptop. More specifically, a dual-core 2.5GHz laptop is, on average, capable of incrementally clustering, adding an observation to the online random forest, making a prediction, and running the MCL in 1.1ms, 2.5ms, 1.2ms, and 195.19ms, respectively. The space complexity of the algorithm is insignificant and consequently not discussed in great details, requiring less than 3MB on average and 22MB in the worst-case.

As demonstrated by the numerous experiments performed in environments as large as 1.5km, the theoretical observations, and the practical results, the proposed algorithm is accurate, efficient, and scalable. Data and code used to present the results presented in this paper are available to the public upon request to the authors.

7 Conclusions and Future work

We have presented a state-of-the-art WiFi localization algorithm that is capable of localizing accurately and running in real-time. The algorithm solves major problems found in other WiFi localizers, since it does not require robots to stop when acquiring WiFi signal strength readings and can build WiFi maps incrementally as part of an online process. The claims made throughout the paper are substantiated by a comprehensive set of experiments analyzing 5 different algorithms across 3 datasets accounting for 4 kilometers of indoor and outdoor exploration. Together, all of the experiments provide compelling evidence regarding the robustness of our approach. The algorithm's accuracy, speed, and low sensor requirements make it particularly suited for exploration scenarios of heterogeneous robot teams consisting of many sensor-deprived (and inexpensive) robots. Autonomous robot navigation and control using the WiFi localizer is an interesting future direction and is also being used to enable heterogeneous map

merging [11].
Numerous directions for future research are possible. First, and similarly to what we did with other map types [7,12], it would be interesting to develop methods that merge, in a principled way, multiple WiFi maps independently developed by multiple *mappers* operating in a shared environment and occasionally exchanging their data. The clustering strategy could also be improved, with backtracking being an attractive modification (i.e., reconsider clusters that have previously been fixed in order to optimize new data points and new clusters). Additionally, our clustering algorithm could be substituted or combined with a parallel implementation capable of processing batches of data with the objective of maximizing localization accuracy. Finally, and as formerly mentioned, one could also explore the use of alternative Bayesian estimation techniques, such as EKF/UKF.

References

1. P. Bahl and V. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *IEEE International Conference on Computer Communications*, pages 775–784, 2000.
2. B. Balaguer, G. Erinc, and S. Carpin. Combining classification and regression for WiFi localization of heterogeneous robot teams in unknown environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3496–3503, 2012.
3. J. Biswas and M. Veloso. WiFi localization and navigation for autonomous indoor mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 4379–4384, 2010.
4. W. Braun and U. Dersch. A physical mobile radio channel model. *IEEE Transactions on Vehicular Technology*, 40(2):472–482, 1991.
5. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
6. L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* CRC Press, Boca Raton, FL, 1984.
7. S. Carpin. Fast and accurate map merging for multi-robot systems. *Autonomous Robots*, 25(3):305–316, 2008.
8. K. Chen and C. Guestrin. Wifi cmu. `http://select.cs.cmu.edu/data/index.html`, 2009.
9. F. Duvallet and A. Tews. WiFi position estimation in industrial environments using Gaussian processes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2216–2221, 2008.
10. L. Eddy and S. Wai. Lego robot guided by wi-fi devices. Technical Report 271-QYA2, The Hong Kong University of Science and Technology, 2010.
11. G. Erinc, B. Balaguer, and S. Carpin. Heterogeneous map merging using wifi signals". In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5258–5264, 2013.
12. G. Erinc and S. Carpin. Anytime merging of appearance-based maps. *Autonomous Robots*, 36(3):241–256, 2014.
13. B. Ferris, D. Fox, and N. Lawrence. WiFi-SLAM using Gaussian process latent variable models. In *International Joint Conferences on Artificial Intelligence*, pages 2480–2485, 2007.
14. B. Ferris, D. Hahnel, and D. Fox. Gaussian processes for signal strength-based location estimation. In *Robotics: Science and Systems*, pages 303–310, 2006.
15. J. Fink, N. Michael, A. Kushleyev, and V. Kumar. Experimental characterization of radio signal propagation in indoor environments with application to estimation and control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2834–2839, 2009.
16. A. Goldsmith. *Wireless Communications.* Cambridge Press, 2005.
17. A. Howard, S. Siddiqi, and G. Sukhatme. An experimental study of localization using wireless ethernet. In *International Conference on Field and Service Robotics*, 2003.
18. J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun, and A. Aggarwal. Efficient, generalized indoor WiFi GraphSLAM. In *IEEE International Conference on Robotics and Automation*, pages 1038–1043, 2011.

19. D. Koutsonikolas, S. Das, Y. Hu, Y. Lu, and C. Lee. CoCoA: Coordinated cooperative localization for mobile multi-robot ad hoc networks. *Ad Hoc & Sensor Wireless Networks*, 3(4):331–352, 2007.
20. A. Ladd, K. Bekris, A. Rudys, D. Wallach, and L. Kavraki. On the feasibility of using wireless ethernet for indoor localization. *IEEE Transactions on Robotics and Automation*, 20(3):555–559, 2004.
21. N. Oza and S. Russell. Online bagging and boosting. In *Artificial Intelligence and Statistics*, pages 105–112, 2001.
22. A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *Workshop on "On-line Learning for Computer Vision" at IEEE ICCV*, 2009.
23. S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*, chapter 5,8. The MIT Press, Cambridge, Massachusetts, 2005.
24. D. Tran and T. Nguyen. Localization in wireless sensor networks based on support vector machines. *IEEE Transactions on Parallel and Distributed Systems*, 19(7):981–994, 2008.
25. M. Youssef, A. Agrawala, A. Shankar, and S. Noh. A probabilistic clustering-based indoor location determination system. Technical Report CS-TR-4350, University of Maryland, College Park, 2002.
26. S. Zickler and M. Veloso. RSS-based relative localization and tethering for moving robots in unknown environments. In *IEEE International Conference on Robotics and Automation*, pages 5466–5471, 2010.