Molé: a Scalable, User-Generated WiFi Positioning Engine

Jonathan Ledlie,
Nokia Research Center,
Cambridge, USA
Email: jonathan.ledlie@nokia.com

Jun-geun Park, Dorothy Curtis,
Massachusetts Institute of Technology,
Cambridge, USA
Email: {jgpark,dcurtis}@mit.edu

André Cavalcante, Leonardo Camara,
Afonso Costa and Robson Vieira
Nokia Institute of Technology, Manaus, Brazil
Email: {andre.cavalcante,leonardo.camara,
afonso.costa,robson.domingos}@indt.org.br

***Abstract*—We describe the design, implementation, and evaluation of Molé, a mobile organic localization engine. Unlike previous work on crowd-sourced WiFi positioning, Molé uses a hierarchical name space. By not relying on a map and by being more strict than uninterpreted names for places, Molé aims for a more flexible and scalable point in the design space of localization systems. Molé employs several new techniques, including a new statistical positioning algorithm to differentiate between neighboring places, a motion detector to reduce update lag, and a scalable "cloud"-based fingerprint distribution system. Molé's localization algorithm, called Maximum Overlap (MAO), accounts for temporal variations in a place's fingerprint in a principled manner. It also allows for aggregation of fingerprints from many users and is compact enough for on-device storage. We show through end-to-end experiments in two deployments that MAO is significantly more accurate than state-of-the-art Bayesian-based localizers. We also show that non-experts can use Molé to quickly survey a building, enabling room-grained location-based services for themselves and others.**

I. Introduction

The ability for a mobile device to perceive a user's location has many applications, from social networking "check-ins" to location-appropriate content, such as automatically presenting people with a relevant train schedule.

While the global positioning system (GPS) enables devices to sense their location in most outdoor environments, bad weather and "urban canyons" can restrict its operation. In addition, there are many indoor positioning applications where GPS can provide only limited assistance, as it typically provides a position fix only near windows and doors.

To enable room-grain indoor and outdoor positioning in GPS-less environments, researchers have used physically-fixed wireless beacons to associate a unique "fingerprint" with each place or grid point [1]–[4]. While the types of wireless beacons have varied over time, most techniques now use 802.11 WiFi beacons because of their near ubiquity, particularly in urban and suburban environments. Because of the difficulty in translating between distance and received signal strength [5], more compact alternatives to fingerprinting – e.g., triangulating among the beacons – are generally eschewed.

One of the key problems with fingerprinting, however, is learning the fingerprint for each place – however "places" are designated. We call the process where a person links a fingerprint to a place "binding." Several commercial vendors offer positioning services, which include a fingerprint-generation survey [6]. However, these come at a steep price: a large office building can cost $10,000 USD with no maintenance included. Because this is prohibitively expensive for many applications – such as contextualizing a device's behavior based on which room of a house it is in – several research groups have begun to crowd-source fingerprints from end-users [7]–[10]. In the model for these Wikipedia-style approaches, a single locally-knowledgeable user performs the bind for a place and many visitors can then rely on the database of fingerprints.

Molé focuses on a new point in the design space in crowd-sourced, or "organic," positioning systems. Some systems, such as OIL [8], present a map to the user: users bind places by clicking on the map. Others, like Redpin [7], allow the association of any text string with a place's fingerprint. In contrast, Molé arranges the world hierarchically; this imposes a clean, intuitive namespace (country, region, ...), and allows for data prefetching at a building scale if not larger. It also isolates problems in the fingerprint database to small portions of the tree. Molé relies on compact data structures that allow many fingerprints to be stored on the user's device. In turn, this allows the user's device – not a server – to differentiate among potential places with similar fingerprints, improving privacy.

Here we describe how Molé's hierarchical namespace leads to a scalable design, where its servers can be easily replicated in the "cloud." We show how its new statistical positioning algorithm uses response rate as additional fingerprint information. In our experiments, this leads to an improvement in accuracy of 10% over the current state-of-the-art. We also show how Molé uses accelerometer-based motion detection both to reduce the latency in showing the correct place after a user has moved and to collect clean fingerprints from end-users. Through a crowd-sourcing experiment, we show that a multi-story building can be quickly and accurately covered by non-experts: in one hour, four people completely surveyed a mid-sized research lab. After this surveying period, which can be concurrent with use, any person visiting the lab can benefit from room-customized behavior, including location-aware assistance, scheduling, notifications, and device behavior.

This paper's contributions are:

- A new organic positioning system, called Molé, that models the world as a tree of hierarchical places.

978-1-4577-1804-5/11/$26.00

- A new positioning algorithm that explicitly accounts for temporal variations in the signal space.
- A simulation and experimental analysis of Molé, including crowd-sourcing a multi-floor building with untrained users.
- An open source implementation of Molé.

The paper proceeds as follows. We describe and show the user interface for the place hierarchy in Section II. We outline our positioning algorithm in Section III. In Section IV, we show how Molé is implemented in the "cloud," allowing for fingerprints to be combined efficiently and for clients to receive the contributions of others quickly. In Section V, we describe our evaluation of Molé, examining its positioning algorithm versus the current state-of-the-art, its use of movement detection, and how end-users can use it to build up a working deployment in a multi-story building. We describe related work in Section VI and conclude in Section VII.

II. Model of Places

Molé arranges the discrete, human-designated places of the world in a hierarchy. While the hierarchy could be of variable depth, our current implementation contains five levels, as the estimate in Figure 1 illustrates. From coarse to fine, the levels typically refer to country, region, city, area, and unique place (e.g. room). Areas are the unit of fingerprint aggregation, transfer, and, therefore, privacy; the server knows at most what areas you visit. Areas typically refer to street addresses (e.g., "4 Cambridge Center" in Figure 1), although they could refer to larger outdoor areas such as parks. The design also allows aggregation at higher levels.

We believe that arranging places in a hierarchy is useful in many organic positioning settings. Earlier approaches have used visual maps [8], [9] or uninterpreted strings [7] to identify individual places. Visual maps require that a fairly accurate map of the area – typically a building – exists. While well-managed places, such as universities and airports, may be able to generate maps, this approach may not scale to individual homes or businesses, where people may not have the time, knowledge, or interest to create a map of their sets of places. In addition, many users find it non-trivial to locate themselves on indoor maps, particularly in complex buildings. Assigning uninterpreted strings to places during a bind has its own challenges: for example, the namespace may rapidly become crowded with similar names. While Barry et al. do allow for spaces within buildings [10], their hierarchy is not intended to cover the world.

Figure 1 shows the current estimate of a device's position within this hierarchy. Users click on the "Incorrect Estimate?" button to edit the current estimate and make a new bind, improving future estimates for themselves and other users. The statistics are explained in Section IV.

III. Algorithm

In this section, we describe our new, statistical localization algorithm (§ III-A), briefly review naïve Bayesian localization (§ III-B), and describe a kernelized RSSI histogram variation that can be applied to both localizers (§ III-C).

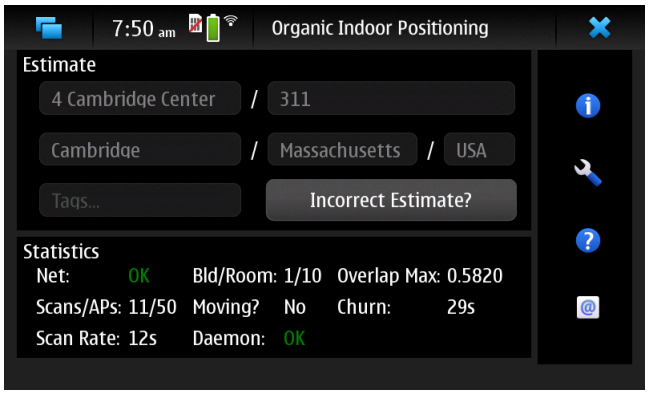


Fig. 1. **Molé's User Interface.** It shows the country, region, city, area, room hierarchy in street address format. The statistics shown are described in Section IV-A.

A. Maximum Overlap Localization

Maximum Overlap, or MAO, selects its estimated place as the one whose fingerprint is most similar to the user's fingerprint, using a similarity function we describe below. The two key advantages to MAO are (1) that it is efficient to compute and (2) that it provides a general scan distance function, which can be used to estimate physical distances between sets of fingerprinted objects. Because we anticipate localization algorithms running continuously in the background on mobile devices, this simple computation should translate into longer battery life. Scan distance functions are also useful for clustering scans, outlier detection, and cleaning scan databases [8]. By themselves, distance functions are also useful for estimating the physical distance between the positions where the scans were made, which we show in Section V-D.

To create a MAO fingerprint, we begin with a standard set of place-to-APs histograms containing raw RSSI readings [3]. As in Haeberlen et al. [3], we summarize each per-place per-AP histogram with a single Gaussian with mean μ and standard deviation σ (we describe a kernelized histogram variant in § III-C).

Every place is assigned a fingerprint, which is a set of mappings from access points to data triples:

$$AP_i \Rightarrow \langle w_i, \mu_i, \sigma_i \rangle \quad (1)$$

where w_i is the weight of AP_i, the number of observable APs is τ, and the total weight for each fingerprint $\sum_{i=1}^{\tau} w_i$ is 1. Note that the most recent k scans of the user also form a fingerprint using the same method.

Determining the weight w to apply to each visible AP is an important component of our algorithm. A straw man method would be to simply weigh each visible AP equally: $1/\tau$. Instead, we base the weight on the probability that the given AP will actually be observed in the place. Specifically, we set the probability to the response rate, the fraction of a fingerprint's scans in which a given AP was observed. When a place is scanned many times, some APs will be seen in every

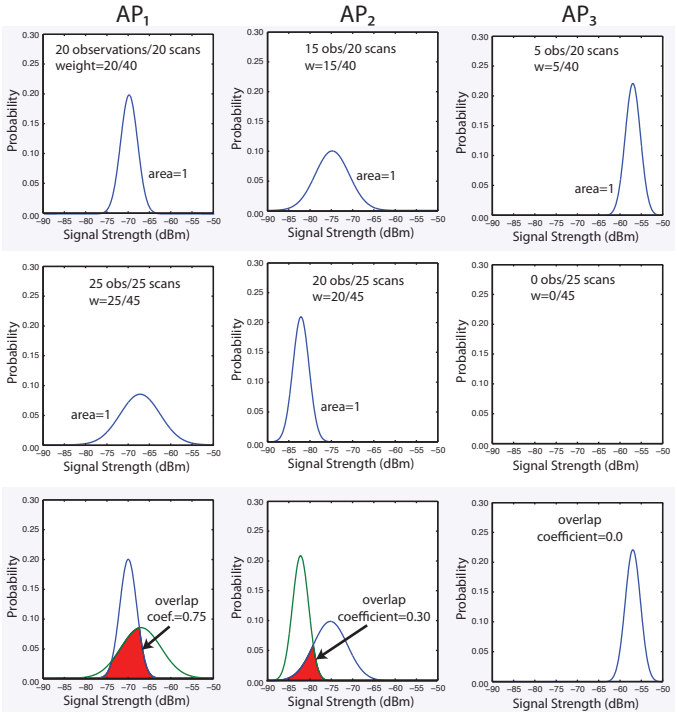


Fig. 2. Example of MAO: $place_1$ (top) $\cup$ $place_2$ (middle) = Overlap (bottom). The 20 scans in $place_1$ have observed three different access points: AP_1, AP_2, AP_3. The 25 scans in $place_2$ have only observed two access points, both of which are the same as those seen in $place_1$: AP_1, AP_2. AP_3 was not observed in $place_2$. To compute the weights for $place_1$, we divide the observations for each AP by the total number of observations: $20 + 15 + 5 = 40$. The same procedure is done for $place_2$. This completes the creation of fingerprints for these two places. The bottom row shows how the similarity between the two fingerprints for places 1 and 2 is computed: $s = 0.75 \times \frac{20/40+25/45}{2} + 0.30 \times \frac{15/40+20/45}{2} - 5/40$ (where $p = 1$). "$Place_2$" could equivalently be a set of scans as seen by a user's device: the algorithm to compute their similarity would be the same.

scan, and some seen only rarely. This captures the intuition that a user's device will see the same APs with the same signal strength distribution *and* the same observation frequency when it is in the same place (these two quantities are only weakly correlated as we show in Section V-A). If the user's fingerprint does not contain an AP that is almost always observed when in a particular place, it is highly unlikely that the user is in this place. Weighting according to response rate reflects this intuition. Specifically, the weight for AP_i is:

$$w_i = r_i / \sum_{j=1}^{\tau} r_j \tag{2}$$

where r_k is the number of readings of AP_k.

To find the similarity between two fingerprints, we determine the similarity in signal strengths of APs that exist in both fingerprints, and penalize for missing APs, weighting both quantities by the response rate. The comparison of any two fingerprints returns a similarity $-2 \leq S \leq 1$, where a comparison of identical fingerprints returns 1 and of disjoint fingerprints returns -2 (Disjoint fingerprints are those that share no access points). For fingerprints A and B, let $S(A,B) = \sum_{i \in A \cup B} \delta_i$, where the effect each AP_i is:

$$\delta_i = \begin{cases} \frac{\omega_a + \omega_b}{2} \times O\left(\mu_a, \sigma_a, \mu_b, \sigma_b\right) & \text{if } i \in A, i \in B, \\ -\omega_a \times p & \text{if } i \in A, i \notin B, \\ -\omega_b \times p & \text{if } i \notin A, i \in B, \end{cases} \tag{3}$$

where $O(\cdot)$ is the overlap coefficient between the two Gaussian distributions [11] and $0 \leq p \leq 1$ is the penalty to apply for missing APs. Figure 2 provides an example of computing the distance between a pair of fingerprints. Because the same comparison applies whether place B is a user's fingerprint or any collection of scans – such as a location tag – it can be used to estimate a physical distance between two real or virtual objects (e.g. virtual graffiti).

One particularly nice aspect of this overlap computation is that it exists as a closed-form function when Gaussians are used to represent the RSSI readings [11]. Alternatively, the results from the function can be stored in a look-up table [12]; we found a table with only hundreds of values gave almost the same results as a function. This simple computation is in contrast to graphical models [13], which can require thousands of iterations to converge.

A special case exists where we have only a few RSSI measurements for an AP. In particular, the sample variance, which is a second-order statistical property, is not well-defined with only one sample. Because this situation typically exists for rarely observed APs, taking more scans is not advisable as we may need to take many more in order to obtain a stable estimate for σ. To estimate σ for these APs, we use a weighted average of this AP's sample standard deviation σ_s (if it exists) together with a common prior σ_c:

$$\sigma_i = \frac{(r_i - 1)\sigma_s + \sigma_c}{r_i} \tag{4}$$

With this, the overlap coefficient can be computed even with very few RSSI values, or even a single value, from a given AP. We found $\sigma_c = 1$ worked well in our experiments.

B. Naïve Bayes Localization

We compare MAO to state-of-the-art naïve Bayes localization in Section V-B. For completeness, we briefly review Bayes localization here (for more detail, see Haeberlen et al. [3] and Madigan et al. [13]). Bayesian localization estimates the most likely location using Bayes' rule. Naïve Bayes localization further assumes that the signal strengths from different access points are independent from each other given a location. Therefore, given a signal strength vector $s = [s_1, s_2, ..., s_k]$ from k access points, the posterior probability of being in location l is given by

$$P(l|s) = \frac{\prod_i^k P(s_i|l)\ P(l)}{P(s)}. \tag{5}$$

With a uniform prior assumption on $p(l)$, the final location estimate $\hat{l}$ is given as follows:

$$\hat{l} = \underset{l}{\operatorname{argmax}} \left[\prod_i^k P(s_i|l) \right]. \tag{6}$$

C. Kernelizing RSSI Histograms

A final algorithmic technique that we have tested in Molé is kernelizing RSSI values – essentially spreading a given reading over adjacent bins – a technique Park et al. used for sharing fingerprints across heterogenous devices [14]. While we showed MAC using a single Gaussian, MAC and Bayesian localizers can use this technique instead: it is independent of the localizer itself. The key observation is that summarizing a set of RSSI values with a Gaussian can often lead to Gaussians that are similar across nearby rooms. The alternative to a Gaussian summarization has typically been to simply leave the RSSI values in raw histogram form. For example, a fingerprint based on histograms might include five -78 dBm readings, three -80 dBm, and one -83 dBm reading from a particular access point. By leaving these values in raw form, a reading of e.g. -79 dBm will be discounted, instead of contributing to a match as it should. A Gaussian summarization, however, will also not yield an accurate picture for the distribution's shape: it is skewed toward -78. Instead, we can apply a kernel to each RSSI value, effectively spreading it out into adjacent bins without affecting the overall shape. This shape can capture differences between neighboring spaces that would be blurred by a Gaussian summary.

One complaint with using histograms as compared to Gaussians is the requirement that – in theory – they consume an order-of-magnitude more space. This increase in space consumption is due to the range of RSSI values (typically -30 to -100 dBm) versus a simple mean and standard deviation. Multiplied by many places and many access points, this space consumption arguably could be significant for on-device positioning. In practice, however, because so few bins of the histogram are used – even when kernelizing – that the actual space consumption is only a few floating point numbers more per access point in our experience.

IV. Implementation

Molé's implementation is divided into client and server components. The client portion periodically scans WiFi signals and makes an estimate of the current place available to other applications on the same mobile device. Because all position estimates are calculated on the client using a cache of fingerprints, the client's exact position remains private and new estimates can be made in the absence of network connectivity.

A. Client Components

The client itself consists of two parts: a daemon, which runs continuously in the background, and a user interface, which is displayed when the user wants to make a bind, modify the daemon's behavior, or view statistics. Figure 1 shows the user interface. Its statistics include: the number of scans being used to form the estimate; the count of distinct APs that were observed within these scans; the current time between scans (i.e. scan period); the number of areas and individual places within those areas under MAC consideration; whether the user is deemed to be moving; the score of the current estimate ("overlap max"); and churn, the time since the estimate was last changed. The Molé daemon exports the current location estimate to all applications on the device, assuming that the user has set "sharing" to be on.

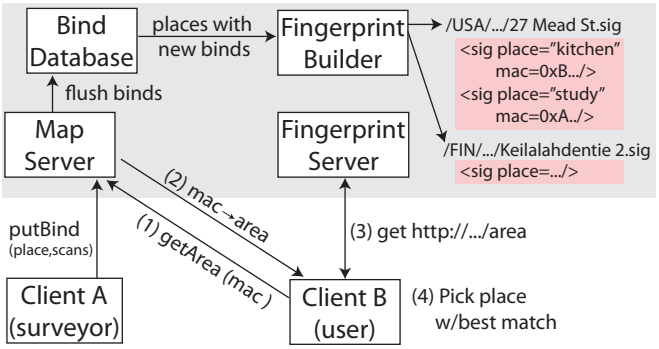


Fig. 3. Interaction between Molé's client and server components. Two paths are shown: (a) a bind coming from a surveyor (client A), being added to the bind database, and being processed into an area's fingerprint file (e.g. Keilalahdentie 2.sig) and (b) a user's device (client B) updating its local cache of fingerprints for the areas that it is potentially in. First it queries to see which areas match a random "loud" MAC with `getArea()`, then it fetches the fingerprint files for those areas. After its cache is up-to-date, it can form a position estimate locally.

Using Motion Detection As Haeberlen et al. showed [3], comparing more user scans against each fingerprint improves spot-on accuracy, with diminishing returns after about eight scans with their data. But frequent scanning reduces battery life, and having a fixed, large number of user scans introduces a lag when the user is moving between places. If a device has an accelerometer, Molé uses it to find a happy medium between battery consumption and update lag. If the device is estimated to be stationary, it slows down the scan rate and other functions. When walking is detected, the current set of user scans is discarded and the scan rate is increased (up to once per 10s in our current implementation). By truncating the user scans (11 in Figure 1), Molé returns a less accurate, but more timely estimate. When the user stops moving, the user scans accumulate and the estimate improves. Because we simply truncate the positioning and bind queues in response to movement, our method is independent of the choice of the particular motion detection algorithm; we use Shafer and Chang's detector [15]. An alternative method would treat the detector as less of a black box and could dynamically adjust the length of the queues based on the magnitude or confidence with which movement was detected. To further reduce battery usage, we run the motion detector every 10 seconds with a duty cycle of 5%; at this rate motion detection has little effect on the overall battery consumption of a typical smartphone. We evaluate the effect of using motion detection on update delay and fingerprint creation in Section V-C.

Client-side Filtering and Positioning Client localization involves fetching the correct area's fingerprint file (if it is not cached on-device), filtering down to a few fingerprints to be tested more precisely, and producing a top estimate with MAC. As shown in Figure 3, the client periodically asks the server for the list of areas associated with one of its visible MACs (step 1), and receives the fully-qualified (hierarchical)

area name in response (step 2). It then requests the area's fingerprint file (step 3) and localizes using the current user scans (step 4). To reduce the number of fingerprints that MAO must compare, we apply Charrow's fingerprint filtering to our local cache to identify a set of "nearby" locations [16]; in Figure 1, ten places have passed this filter. We use the filter twice: first on the cached areas, then on the cached places within the unfiltered areas. Because areas can contain many places, this greatly reduces the number of places that must be compared when many areas are cached on the device. In addition, because the filter uses only MAC presence/absence, it is far less CPU intensive than a room-level localizer. The more CPU intensive MAO then runs on the smaller subset of places that have successfully passed the area and place filtering steps. Like other room-level localizers, MAO's CPU usage is linear in the number of potential places under consideration, so reducing the set under its consideration can reduce battery consumption considerably when many places are cached.

B. Server Components

Figure 3 shows Molé's four main server components and the key methods clients use to make binds and access fingerprints. Molé's server side is designed to run elastically "in the cloud:" its client-facing components, the Map Server and Fingerprint Server are easy to replicate. The figure shows the two paths of client actions: (a) binding and (b) localizing. A client bind is sent to the Map Server, which acts as a write-back cache. The Fingerprint Builder periodically monitors the database for places with new binds (or entirely new places). For each of these places, it aggregates all recent binds and determines a new fingerprint. Fingerprints for other places in the same area are cached in the database. The builder then writes out each changed area's collection of fingerprints in a single area fingerprint file. Because these files change infrequently and are named by the fully-qualified area, they can be trivially cached, versioned, and diffed.

Molé's server components are currently hosted on Amazon Web Services. While we show only one server instance in the figure, it is fairly trivial to replicate and scale the server components because they can be divided geographically; that is, the bind database, in particular, can be partitioned down to the level of individual areas if need be. Because area fingerprint files change slowly over time after their initial creation period, we serve these files with an efficient static web server. Replicas could be further pushed toward the client with a content delivery network. To receive fingerprints created by other nearby users, clients poll for changes in their current area's fingerprint at one minute intervals.

The source code for Molé has been released under an open source license and we invite contributions. The client components are $\approx 7k$ lines of Qt/C++; the server is written in Java and Perl and relies on several open source libraries.

V. Evaluation

We have successfully tested Molé in preliminary trials at several labs, using Nokia N900 tablets. Here, we examine Molé in detail, both at the algorithmic and end-user level:

- We find that MAO's weight (based on response rate) and RSSI readings contribute independently to uniquely identifying a place (§ V-A).
- Through a set of controlled experiments, we find that MAO has favorable accuracy results as compared to a state-of-the-art Bayes localizer, achieving better performance when places are physically adjacent (§ V-B).
- We show that use of a motion detector can result in a dramatic improvement in update delay and in unpolluted fingerprint creation in organic settings (§ V-C).
- We show how MAO can be used to estimate the physical distance between two objects (§ V-D).
- Using the results from a deployment in a two story building, we show that Molé can rapidly crowd-source an accurate location system (§ V-E).

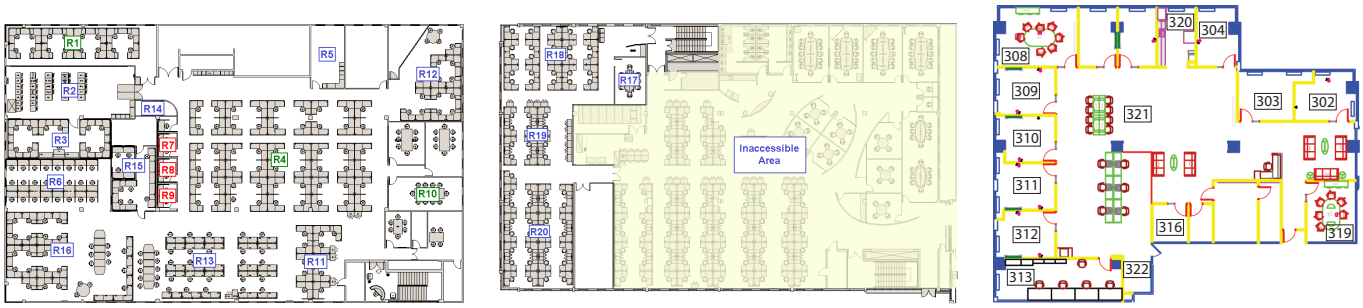
A. Using Response Rate

Before we examine Molé's performance, it is reasonable to ask whether it is valid to use response rate as the basis for MAO's weighting factor at all. That is, is the response rate supplying distinct and consistent information as compared with RSSI values, or could one be substituted for the other? Specifically, we ask: (a) Are they redundant quantities? (b) Is response rate consistent over visits to the same space? (c) Do they increase differentiability between different spaces? and (d) Does the weight improve end-to-end accuracy? We also examine the effect of weight experimentally in Section V-B.

First, using bind data from different alpha users of Molé, we find that they are not redundant quantities. Using data from 81 binds from different indoor environments (e.g. labs, houses), we compared the average RSSI value versus the response rate for the same MAC. In this data set, the two are only moderately correlated ($\rho^2 = 0.62$), suggesting that response rate, and therefore Molé's normalized weighting measure, provide additional information beyond received signal strength (Scatterplots elided due to space constraints).

Second, to examine consistency over time, we compared an older set of binds to a newer one for eight places in one of our labs. The older set were all generated at least six months earlier than the newer set. Fingerprints in each place contained 31 distinct MAC addresses on average. We found a strong correlation for both response rate ($\rho^2 = 0.73$) and RSSI ($\rho^2 = 0.87$), suggesting that these mainly independent values stay consistent over time, and, therefore, can separately assist in identifying individual places.

Third, we examine whether MAO's weight is correlated with RSSI overlap; that is, when comparing across places, is it providing additional, differentiable information. Using a professionally-collected scan data set from a nine-story building which contains more than 1,400 distinct places, we compared the weight $\frac{\omega_a+\omega_b}{2}$ and the overlap $O(\cdot)$ for all MACs in common across places (≈ 150 million entries). When overlap is computed with Gaussians, we find a correlation of $\rho^2 = 0.11$ and, when it is computed using kernelized histograms, that of $\rho^2 = 0.08$. We also divided the rooms that



(a) Lab A Third Floor Map (b) Lab A Second Floor Map (c) Lab B Map

Fig. 4. Stationary tablets for "nearby" experiment placed in R7, R8, and R9 (in red); for "distant" in R1, R4, and R10 (in green) on Lab A Third Floor.

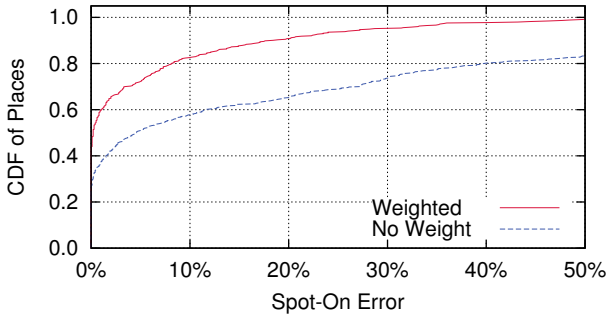


Fig. 5. When the weighting functionality is switched off and all RF sources are counted equally, we found greatly reduced accuracy. For example, only 9% of places had an error rate of less than 20% using the weighting function, but 34% had this same level of accuracy without it.

were close to each other and far away (less than the median distance of 12m or greater than that, respectively), and found the correlation essentially unchanged. Collectively, this set of results suggests that not only are response rate and weight consistent over time, but also that they provide independent information for comparison among places.

Lastly, we examined the effect of eliminating the weighting factor on positioning accuracy. In simulation, we prepared a scan data set from the same nine-story building as above. We first excluded places with fewer than three visible APs or fewer than ten scans, removing 6% of places. Next we assigned a fingerprint to each place, assuming knowledge of all scans of the place. For each place, we took eight scan samples to build a "user" fingerprint, and then observed which place had the maximum matching fingerprint. If the place the localizer estimated was the same as the user's, this was deemed a spot-on estimate. We repeated this test 1000 times for each place: for example, an accuracy of 80% means that we correctly localized $\frac{800}{1000}$ times. Using the scan trace, Figure 5 shows the effect of weighting according to response rate as compared to weighting each AP equally, i.e. setting $w = \frac{1}{\tau}$. While it is possible other refinements exist, such as weighting according to the maximum RSSI value seen for the given AP, it is clear that a reasonable weighting is more accurate than simply valuing all APs equally. In addition to comparing MAO to a Bayesian localizer, we confirm the positive effect of basing a weight on response rate experimentally in the following section.

B. Positioning Algorithms

To examine Molé's positioning performance in a controlled setting, we conducted three experiments in two different labs, shown in Figure 4. The goal of these experiments is to compare MAO with a state-of-the-art naïve Bayes algorithm, using both Gaussian summaries and kernelized histograms.

In Lab A, we conducted two end-to-end comparisons: a "nearby" experiment and a "distant" experiment. In the "nearby" experiment, the target rooms were adjacent to each other, separated by less than three meters, and had glass walls on one side. In the "distant" experiment, the targets are spread uniformly over the floor. In both experiments, we placed a stationary *spot-check* device in each target room. The spot-check devices did not move throughout the experiment; they polled the server for new fingerprints at one minute intervals, performed scans at ten second intervals, and computed four estimates – one for each algorithm – using the same set of scans and fingerprints at the same time. Another device acted as the roving, crowd-sourcing surveyor. With it in hand, a member of our team walked to 6-8 rooms on the floor, including those being tested, and bound each room with 2-3 minutes worth of scans. The bind data was sent to the server, processed into a fingerprint, and made available for download by the spot-check tablets. The bind database was cleared before each experiment.

Results from the "distant" experiment showed that all of the algorithms were able to perfectly distinguish rooms when the rooms under test were tens of meters apart. Here, the target rooms were R1, R4, and R10, as highlighted in Figure 4a. The mean distance between each of these target rooms and all of the other rooms where binds were performed (R2, R3, R5, R11, R13) was 21.59m, 15.90m and 22.59m, respectively. Data from the spot-check tablet in Room R4 in shown in Figure 6. Initially, all devices select Room R1 because that was the only entry in the database. The roving survey tablet binds Room R4 at minute 10. After the spot-check tablet fetches the area's new fingerprint, which now includes R10, all of the algorithms change their prediction to R10. Even with binds in three rooms that are less than 14m away, the algorithms

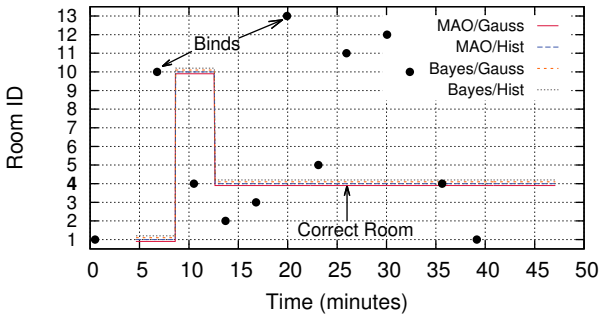


Fig. 6. When the locations with fingerprints are tens of meters apart, all of the localization algorithms were able to successfully select the correct space. Soon after the roving tablet bound R4, all of the algorithms running on the stationary device in R4 selected it.

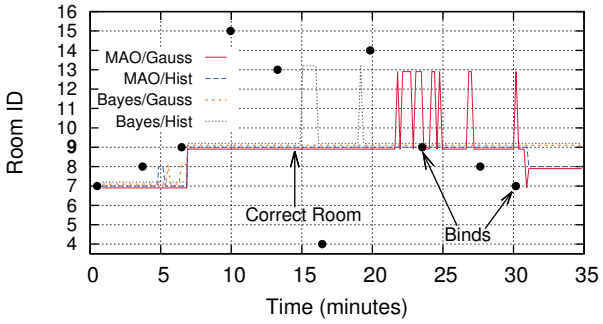


Fig. 7. The time series from the "nearby" experiment illustrates how both MAO and Bayes can exhibit instability.

continue to estimate the correct room.

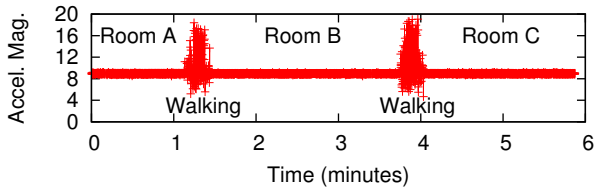
In contrast to the "distant" experiment, the "nearby" one exemplifies the challenges in fine-grain WiFi localization. The three rooms (R7, R8, R9) are small meeting rooms, each 5.8m^2. They lie in a line with the two end rooms 2.7m from the center one, R8. The entry walls and doors are glass, and join a common hallway. We examined the average spot-on hit rate for each room's stationary tablet, in addition to the mean error and its deviation. The results show that, while Bayes with Gaussians had the lowest mean error for the two end rooms, MAO with kernelized histograms had the best overall performance: its average hit rate was 90.7% while Bayes with Gaussians, the next best, was 68.4%. Two interesting behaviors, in particular, are apparent from examining time-series plots for the stationary tablets. First, MAO is more willing to shift between rooms – sometimes to its advantage, sometimes to its detriment. It does this because it has no hysteresis, or prior, like naïve Bayes does. MAO with Gaussians was particularly unstable in this experiment (see, for example, minutes 22-25 in Figure 7). Second, MAO with kernelized histograms was the only algorithm able to consistently differentiate between the two immediately adjacent spaces; this is what leads it to having the highest average hit rate. From this we conclude that any of the algorithms are acceptable in medium to coarse grained scenarios, as in the "distant" experiment, but that MAO with kernelized histograms may supply the best average-case performance if room-grain accuracy is required.

TABLE I
ALGORITHM COMPARISON (LAB B) - SPOT-ON HIT RATE

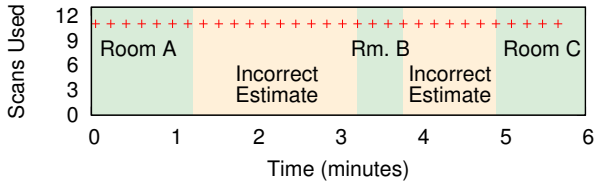
Room	MAO/ Gauss	MAO/ Hist./ No Wt.	MAO/ Hist./ No Pen.	MAO/ Hist./ Penalty	Bayes/ Gauss	Bayes/ Hist
302	92.87	100.00	100.00	100.00	100.00	100.00
303	90.59	100.00	100.00	100.00	74.64	88.52
304	92.09	98.73	100.00	100.00	91.46	79.75
308	99.84	99.84	100.00	100.00	90.49	100.00
309	32.86	42.70	59.37	58.89	31.75	43.33
310	1.59	10.95	41.75	81.59	24.76	11.43
311	90.59	41.15	87.08	94.90	87.56	95.37
312	42.88	93.67	90.03	70.89	70.09	72.47
313	100.00	100.00	100.00	100.00	99.16	100.00
316	100.00	100.00	100.00	100.00	98.73	100.00
319	99.84	100.00	100.00	100.00	98.57	97.14
320	100.00	98.74	100.00	100.00	99.37	100.00
321	99.22	100.00	100.00	100.00	100.00	100.00
322	88.38	100.00	100.00	99.53	92.15	97.33
All	80.46	84.49	91.17	93.16	82.52	84.43

In Lab B, we had more spot-check tablets available and conducted a larger-scale, longer duration experiment. We first instrumented the tablets to run six localizers in parallel, each producing an estimate using the same set of scans at the same time. In particular, we wanted to see the effect of different parameters for MAO with kernelized histograms in a live setting. Referring to Equation 3, we ran MAO: (a) with *no weight*, treating all APs equally (as in Figure 5) and with no penalty for missing APs, (b) with weight but *no penalty*, weighting APs by response rate, but still without the $-\omega_a \times p$ and $-\omega_b \times p$ factors, and (c) with weight and a *penalty*, where $p = \frac{1}{4}$. We placed fourteen spot-check devices in different rooms in the lab, including three in public spaces without doors between them (320, 321, and 322 in Figure 4). Using two roving devices, we bound the spaces starting in room 312 and moving clockwise around the lab; each bind contributed approximately five minutes worth of scans (about 30 scans). All data collection was done during an active workday with people moving around the lab during the experiment.

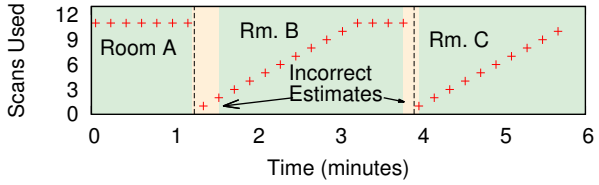
We examined the spot-on accuracy of the stationary tablets during two periods: immediately after all of the rooms had been bound and 24 hours later. We report on the later data although the results were similar. We highlight three aspects of the results, shown in Table I. First, in contrast to previous simulation results which found that Bayes with Gaussians exhibits accuracies above 95% [3], it had a hit rate of 82.5% in this more challenging live setting. Second, as in the "nearby" experiment above, no algorithm was able to consistently differentiate between small, adjacent offices that had doors open to a common hallway (309, 310, 311, 312); the centroids of the rooms are about 2.5m apart. MAO with kernelized histograms and with a weak penalty for missing APs performed best, on average, but even it was not consistent. Second, overall this same algorithm and parameter choice performed best overall. In particular, it was significantly more stable throughout the lab and was 10% more accurate than Bayes with kernelized histograms, the current state-of-the-art.



(a) Accelerometer Magnitude (at 10ms intervals)



(b) Place estimates without motion detector



(c) Place estimates with motion detector

Fig. 8. Using a motion detector to vary the number of scans used by the localizer significantly reduces update lag, presenting more up-to-date results to the user. Each + shows when the localizer received a new scan and produced a new estimate. The dashed line shows when the periodic motion detector determined that the user was walking.

C. Motion Detection

We wanted to examine the effect of using a motion detector to improve update lag and fingerprint clarity. Update lag occurs when a person moves from one room to another but the localizer does not reflect the new room immediately. This lag occurs because the localizer uses stale data: scans collected in a previous room or while walking are still being used to form the estimate of the current location. We performed an experiment where we examined the use of a simple motion detector to expire old scans; the user's fingerprint repopulated with scans when the user stopped moving. Intuitively we would like to use as many scans as possible, but only if those scans come from the location the user is actually in.

Figure 8 illustrates how using a walk detector can significantly improve the user experience. In this experiment, we walked from room A to room B to room C, staying in room B for about two minutes. We logged the raw accelerometer readings at 10ms intervals (Figure 8a) and ran two instances of Molé, both running MAO with Gaussians. One instance did not use the motion detector and the other used the periodic motion detector described in Section IV-A, sampling for 0.5 seconds every 10 seconds. All fingerprints were cached and did not change during the experiment.

When Molé did not use a motion detector (Figure 8b), the estimate lagged behind the ground truth for one to two minutes because it used stale scans. When the instance running the motion detector detects motion (the dashed lines in Figure 8c), the localizer's scan queue is immediately truncated and there is far less delay before the correct space is chosen.

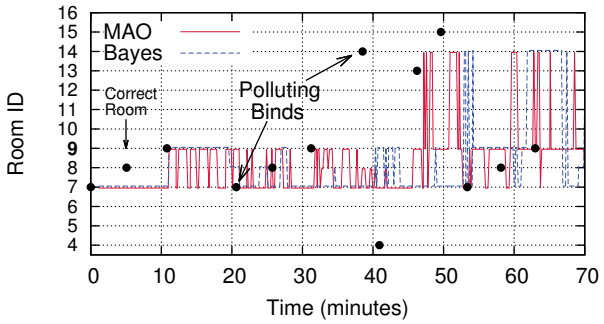


Fig. 9. In a crowd-sourced positioning system without motion detection, fingerprints can easily become polluted with scans from old rooms. The data show that the two fingerprints of Rooms R7 and R14 erroneously acquired scans collected in Room R9, causing the estimate for R9 to vacillate with both positioning algorithms.

A second significant benefit to using a motion detector in a crowd-sourced positioning system is improved fingerprint clarity. In a crowd-sourced environment, a user can walk into a room, notice the estimate is incorrect, and immediately send a correcting bind. Unfortunately, this can lead to scans collected prior to the user entering the room becoming part of the fingerprint: a polluted fingerprint. To mitigate this problem, we truncate the on-device "bind" queue whenever walking is detected. This queue constitutes the scans that will be bound to the place if the user makes a correction.

To illustrate fingerprint pollution, we conducted the same "nearby" experiment as described in Section V-B only with the motion detector switched off. Figure 9 illustrates two instances where the fingerprint for Room R9, where the spot-check tablet is located, is polluted by binds in other rooms. When the roving tablet binds Room R7 for the second time at minute 20, we observe a shift in the estimates from R9 to R7. This occurred because many of the scans that were collected in Room R9 were not dropped when the user walked into Room R7, causing R7's fingerprint to become similar to R9's. The second instance is when Room R14 is bound at minute 39, soon after the user leaves the target room, R9. Again, this causes R14 and R9 to erroneously have similar fingerprints, resulting in the localizers vacillating between several locations. As Figure 7 shows, *with* the motion detector switched on, fingerprints do not become polluted when new binds occur.

D. Using Fingerprint Similarity

One potential advantage of MAO is that it provides an abstract similarity function between any two fingerprints, either by using Equation 3 directly or by replacing $O(\cdot)$ with the overlap of the two histograms. Inferring physical distance from fingerprint distance has many uses, from the canonical "finding the nearest printer" to proximity-based notifications and device-pairing.

By processing the scans from the 1,400 room building discussed above, we found that a discernible and useful cor-

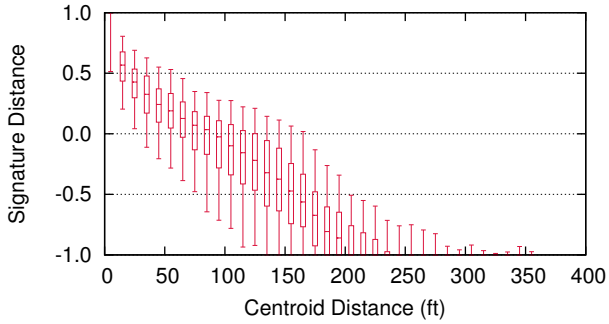


Fig. 10. Fingerprint-to-Physical Distance Correlation. The data show that there is a strong correlation between fingerprint similarity and physical distance, particularly when spaces are nearby to one another. Physical distance is measured as centroid-to-centroid in a nine-story building which contains more than 1,400 distinct spaces.

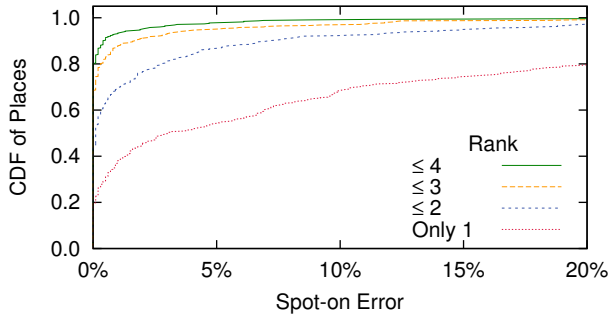


Fig. 11. Many of the places in our data set are topologically close to one another. Topologically close places often have similar RF fingerprints, which occasionally confounds RF-based localization. If we allow for non-exact matches, the data show that the true correct place is often only a few steps away from our best guess.

relation existed between the physical distance and fingerprint similarity across pairs of rooms. Using MAO with Gaussians, we show the correlation for this data set in Figure 10. Given objects or spaces tagged with fingerprints, this suggests that MAO can be used to estimate physical distances between them at a finer grain than simply observing that they can see the same MAC, for example. In this data set, spaces which had a fingerprint similarity > 0.5 were always less than 100 feet apart. Because the fingerprint similarity computation is fairly trivial, it would also be possible to see if any k devices were likely to be within some physical distance of one another.

A second use for fingerprint similarity is that, even when the correct place is not the *most* similar to the user's fingerprint, it is almost always *one of* of the most similar. Because MAO returns a similarity score for each potential place, it is possible to look down the list of returned places beyond the top ranked place. Figure 11 shows that the correct place is almost always in the top four ranked places. In a visual map application, all of the highly ranked places could be highlighted if one did not stand out, assisting the user in making a correction.

E. Crowd-sourcing Behavior

For our last experiment, we wanted to understand whether Molé could be used by untrained participants. We first modified the user interface to include a positive feedback button ("Estimate OK"), signifying that the displayed estimate was, in fact, correct. We recruited four volunteers from Lab A; they had seen us testing Molé previously, but were otherwise untrained. Before giving the tablets to the volunteers, we performed one bind in Room R1 on an empty database. In effect, this initialized the hierarchy shown on all of the tablets so the volunteers would only need to edit the room label. They were given instructions to walk from room to room, fixing the estimate when it was wrong and clicking "Estimate OK" when it was correct. They were allowed to wait up to 30 seconds (three scans) for the estimate to become correct before marking "OK" and up to 60 seconds (six scans) before binding a correction.

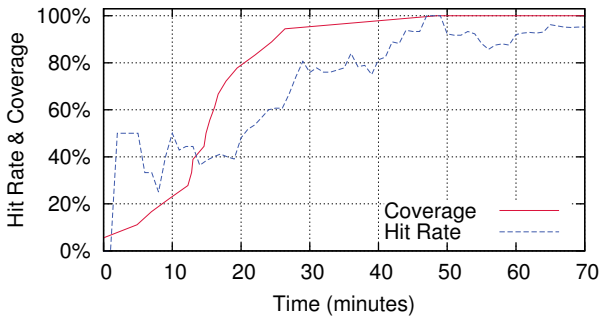


Fig. 12. Four untrained volunteers were able to quickly survey a two story lab, with resulting accuracies above 85%. No floor errors were encountered.

In the middle of the workday, the volunteers then surveyed two floors for seventy minutes, splitting their time about 75/25 across the third and second floors, respectively (see Figure 4). Two of the meeting rooms, R14 and R17, were occupied during the experiment, and so were left unbound. Figure 12 shows how coverage and spot-on hit rate changed during the experiment. We calculated hit rate as a ten minute moving average of spot-on accuracy (i.e. when the volunteer clicked "Estimate OK"); this included the first bind for each room which is, by definition, an incorrect estimate. Once the rooms were surveyed after minute 50, the hit rate remained above 85% as it had done in our controlled experiments. No incorrect floor estimates were encountered.

The volunteers provided us with feedback on Molé's usability and utility. They found the motion detector was occasionally not sensitive enough and that the device needed to be artificially shaken when it did not detect their walking (we had instructed them to do this). Because binds are sent to the server and not immediately applied to the binding client's area fingerprint cache (see Figure 3), there can be up to a twenty second delay in reflecting binds back to the user; several volunteers found this confusing. We plan on fixing this and the motion detector in upcoming versions. In general, the volunteers found Molé highly accurate, although they noticed the estimates were most often wrong in the same small, adjacent rooms we used for the "nearby" case (§ V-B). They enthusiastically described several potential use cases such as navigating shopping malls, airports, and museums,

finding promotions when shopping, and locating friends.

VI. Related Work

While there is much previous research on indoor positioning in general [1]–[4], [7]–[10], [17]–[20], here we focus on work similar to MAO and prior uses of motion detection to augment WiFi-based localization algorithms.

We use the overlap of pairs of Gaussian summaries, weighted by response rate, as one of MAO's localizers (we found that using the overlap of kernelized histograms provided superior accuracy, however). Lemelson et al. use unweighted Gaussian overlap – not to localize – but to anticipate the likely estimate localization error for a given point [21]. They show that points with very similar fingerprints (as determined by the overlap function) tend to have poor localization accuracy, because they are often confused with adjacent points. Our results show that Gaussian overlap with weighting is clearly superior to a weighting based on response rate, which performs the worst of the algorithms under test in Table I.

We showed how Molé varies the length of the localizer's scan queue to use many recent scans, but only if those scans are likely to be from the current place. Truncating the scan queue on movement detection also prevented bind pollution (§ V-C). Several pieces of prior work have used accelerometer-based motion detection in location systems in different ways. Kim et al. [20] use motion detection to save energy: after a user has arrived at a place and enough scans have been collected, and the variance in the motion detected is low enough, the WiFi radios are turned off. Once the motion variance exceeds its threshold, WiFi scanning is resumed. Shafer and Chang [15] detect movement and, if walking is detected, perform what they refer to as a "full localization," which presumably entails taking many scans over a short period after truncating the scan queue. If a user's walk is longer than the movement detection period, this can result in significantly more battery drain than our method because long series of scans will be repeatedly discarded. They also propose using low variance to detect idleness, choosing to scan slowly rather than switch off WiFi entirely. Bolliger et al. [22] describe *asynchronous interval labeling* which allows sets of scans collected during the same stationary period to be retroactively bound at a more convenient time. In the future, we plan to combine slow background scans, an idleness detector, and a distinguishing MAO score to generate automatic binds that maintain an area's fingerprints as access points change over time due to maintenance events.

VII. Conclusion

This paper presented Molé, a mobile organic localization engine, specifically designed for large-scale positioning. We focused on three key components of Molé: its hierarchical arrangement of places – which allows for unambiguous interpretation of users' location input – its efficient and accurate localization algorithm, and its "cloud"-based server design. Together, these components contribute to a positioning system that can run compactly on a broad range of mobile devices and scale worldwide. In particular, through controlled experiments and simulations, we showed that our localization algorithm was 10% more accurate than the current state-of-the-art. This boost in accuracy occurred because we used discriminating information – response rate – that prior work had discarded. We also showed how the use of a motion detector can significantly reduce user-perceived estimation latency and eliminate bind pollution, where scans collected outside of a room spuriously become part of that room's fingerprint. Finally, we gave Molé to untrained users and found that they could quickly survey a medium-sized building, resulting in an accurate, shared location database that could be used for many applications. We plan on extending the hierarchical and scalable structure of Molé to a visual map-based UI and on constructing a browser plug-in version of Molé.

Acknowledgment

We thank the RVSN research group, especially Prof. Seth Teller, at MIT for their on-going collaboration and insights.

References

[1] P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," in *INFOCOM*, Mar. 2000.
[2] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," in *MOBICOM*, Boston, MA, Aug. 2000.
[3] A. Haeberlen *et al.*, "Practical Robust Localization over Large-Scale 802.11 Wireless Networks," in *MOBICOM*, Philadelphia, PA, Sep. 2004.
[4] J. Krumm and J. Platt, "Minimizing Calibration Effort for an Indoor 802.11 Device Location Measurement System," Microsoft Research, Tech. Rep. MSR-TR-03-82, Nov. 2003.
[5] K. Pahlavan *et al.*, "Wideband Radio Propagation Modeling for Indoor Geolocation Applications," *IEEE Comm. Mag.*, April 1998.
[6] "Ekahau positioning engine," ekahau.com.
[7] P. Bolliger, "RedPin: Adaptive, Zero-Configuration Indoor Localization," in *LoCA*, Oberpfaffenhofen, Germany, 2008.
[8] J. Park *et al.*, "Growing an Organic Indoor Location System," in *MobiSys*, San Francisco, CA, Jun. 2010.
[9] E. S. Bhasker, S. W. Brown *et al.*, "Employing user feedback for fast, accurate, low-maintenance geolocationing," in *PerCom*, Mar. 2004.
[10] A. Barry, B. Fischer, and M. Chang, "A long-duration study of user-trained 802.11 localization," in *MELT*, Orlando, FL, Sep. 2009.
[11] H. F. Inman and E. L. Bradley, "The overlapping coefficient as a measure of agreement between probability distributions and point estimation of the overlap of two normal densities," *Communications in Statistics-Theory and Methods*, vol. 18, no. 10, pp. 3851–3874, 1989.
[12] J. M. Linacre, "Overlapping Normal Distributions," *Rasch Measurement Transactions*, vol. 10, no. 1, 1996.
[13] D. Madigan, E. Einahrawy *et al.*, "Bayesian Indoor Positioning Systems," in *INFOCOM*, Miami, FL, March 2005.
[14] J. Park, D. Curtis, S. Teller, and J. Ledlie, "Implications of Device Diversity for Organic Localization," in *INFOCOM*, Apr. 2011.
[15] I. Shafer and M. L. Chang, "Movement Detection for Power-Efficient Smartphone WLAN Localization," in *WSWiM*, Miami Beach, FL, 2010.
[16] B. Charrow, "Organic Indoor Location: Infrastructure and Applications," Master's thesis, Massachusetts Institute of Technology, Feb. 2010.
[17] T. Gallagher, B. Li, A. G. Dempster, and C. Rizos, "A Sector-based Campus-wide Indoor Positioning System," in *IPIN*, 2010.
[18] A. LaMarca *et al.*, "Place Lab: Device Positioning Using Radio Beacons in the Wild," in *Perv Comp*, Germany, May 2005.
[19] R. Want, V. Falcao, and J. Gibbons, "The Active Badge Location System," *ACM Trans on Information Systems*, vol. 10, pp. 91–102, 1992.
[20] D. H. Kim, Y. Kim, D. Estrin, and M. B. Srivastava, "SensLoc: Sensing Everyday Places and Paths using Less Energy," in *SenSys*, 2010.
[21] H. Lemelson, M. Kjaergaard, R. Hansen, and T. King, "Error Estimation for Indoor 802.11 Location Fingerprinting," in *LoCa*, May 2009.
[22] P. Bolliger *et al.*, "Improving Location Fingerprinting through Motion Detection and Asynchronous Interval Labeling," in *LoCA*, May 2009.