

Growing an Organic Indoor Location System

Jun-geun Park[†], Ben Charrow[†], Dorothy Curtis[†], Jonathan Battat[†],
Einat Minkov[§], Jamey Hicks[§], Seth Teller[†], Jonathan Ledlie[§]

[†]MIT CSAIL, Cambridge, MA, USA

[§]Nokia Research, Cambridge, MA, USA

[†]{jgpark, dcurtis, teller}@csail.mit.edu, [†]{bcharrow, battat}@alum.mit.edu
[§]{einat.minkov, jamey.hicks, jonathan.ledlie}@nokia.com

ABSTRACT

Most current methods for 802.11-based indoor localization depend on surveys conducted by experts or skilled technicians. Some recent systems have incorporated surveying by users. Structuring localization systems “organically,” however, introduces its own set of challenges: conveying uncertainty, determining when user input is actually required, and discounting erroneous and stale data. Through deployment of an organic location system in our nine-story building, which contains nearly 1,400 distinct spaces, we evaluate new algorithms for addressing these challenges. We describe the use of Voronoi regions for conveying uncertainty and reasoning about gaps in coverage, and a clustering method for identifying potentially erroneous user data. Our algorithms facilitate rapid coverage while maintaining positioning accuracy comparable to that achievable with survey-driven indoor deployments.

Categories and Subject Descriptors

C.2.4 [Computer Communication Networks]: Distributed Systems—*Distributed Applications*; H.5.3 [Information Interfaces and Presentation]: Groups and Organization Interfaces—*Collaborative Computing*

General Terms

Algorithms, Experimentation, Human Factors, Measurement

Keywords

Localization, Crowd-Sourcing, Location-Based Services

1. INTRODUCTION

Incorporation of information about a user’s location can enhance a variety of applications, including calendars, reminders, navigation assistants, and communication tools. For example, the Locale application automatically adjusts mobile phone behavior based on location [18]. However, most current location-aware applications are restricted to

outdoor operation; they depend upon GPS [15], which requires clear sky visibility and may take minutes to provide a location estimate.

Much of the research into alternatives to GPS has converged on methods that rely on existing wireless and cellular infrastructure (*e.g.*, [2, 13, 21]). These methods share underlying elements: first, create a database that associates ambient wireless or cellular signals, or *fingerprints*, with physical locations; next, to localize, find the most similar fingerprint in the database to what one’s device currently observes, and return the associated location as the result. While these methods can localize indoors to within a few meters in regions with high infrastructure coverage [13], they have a high deployment burden. Surveyors must methodically walk from room to room, gaining access to all areas of a building to create the required fingerprint database [11]. For a moderately-sized office building, this process can take several days and cost tens of thousands of dollars, and must be repeated when the wireless infrastructure changes.

Because this deployment cost is prohibitive for all but the most managed environments (*e.g.*, airports), researchers have developed systems in which users perform the required surveying activity [3–5, 35]. While these *organic* location systems reduce deployment and management burden significantly, they also introduce a new set of challenges. For example, if the fingerprint database is initially empty and grows in a piecemeal fashion, thus providing location estimates of spatially-varying quality, how can the system meaningfully convey to users both the need for more data, and the relative accuracy of its current location estimate? How can the system determine when to prompt user-surveyors for input? Insufficient prompting will not produce enough fingerprint data for a useful system, while too much prompting will annoy users. Additionally, user-surveyors will provide data of varying quality; how can the system sift through users’ contributions to retain accurate contributions while discarding stale, erroneous or even malicious data? This paper addresses these questions through the following contributions:

- A Voronoi diagram-based method for conveying localizer uncertainty and increasing coverage;
- A clustering-based method that automatically discards erroneous user input through outlier detection in the signal space; and
- An evaluation of these methods during a nine-day trial with nineteen users.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobiSys'10, June 15–18, 2010, San Francisco, California, USA.
Copyright 2010 ACM 978-1-60558-985-5/10/06 ...\$10.00.

The next section provides the necessary background required to understand the algorithmic and systems contributions of the paper. Section 3 describes our Voronoi-based method for characterizing localization uncertainty. Section 4 describes how RF scan data can be clustered and how these signal-space clusters can be used to detect outliers arising from erroneous user input. Section 5 discusses our system implementation and user interface. Section 6 details our evaluation of these new algorithms and our system as a whole through simulation and a live nine-day deployment. Section 7 reviews related work. Section 8 concludes and discusses future work.

2. BACKGROUND

GPS receivers [15] estimate position by trilateration from a set of government-managed satellites. Because GPS functions well only in outdoor regions with sufficient sky visibility, researchers have explored alternative means for indoor localization. Most proposed approaches require dedicated infrastructure, such as fixed beacons, to support localization [28, 38].

One non-GPS localization approach relies on ambient wireless and cellular network signals [2, 14]. Because this “infrastructure” comes as a side-effect of providing network coverage and incurs no additional cost, the research and commercial communities have become interested in this type of method for a variety of applications. Several positioning systems, including the system described in this paper, have adopted a fingerprint-based approach to associate ambient network signals with particular spaces.

2.1 Fingerprint-based Localization

Fingerprint-based localization methods exploit the spatial variation in available radio frequency (RF) signals, such as 802.11 and cellular broadcasts, compiling this information into a map [2, 21]. The location of a mobile device can then be estimated by identifying the space within the map whose fingerprint best matches the fingerprint recently observed by the device. Researchers have previously reported room-level location accuracy of 95% within a building-sized testbed region using this approach [13].

The observed per-MAC signal strengths constitute the *fingerprint* for a particular space. Figure 1 depicts four such fingerprints. Each of the four access points (APs) may be received in each space with varying strengths. Due to walls, distance, and other factors, the signals observed within a particular space differ substantially from those observed in other spaces, even those that are directly adjacent.

Together, the RF signals observed in a given space form that space’s fingerprint; many spaces collectively make up a *signal strength map*. Because most indoor RF sources are geographically fixed, fingerprints are fairly consistent over time, unless of course APs are repositioned, added, or removed.

2.1.1 Bayesian Localization

Given a database of fingerprinted locations L and a set of observations o , the goal of localization is to infer the most likely location \hat{l} of the mobile device. Specifically, in 802.11 localization systems, an observation o typically consists of a set of per-AP signal strengths, $\{s_i | i \in \text{AP}\}$.

The Bayesian localization method addresses this problem using Bayes’ rule, computing the degree of belief in the hy-

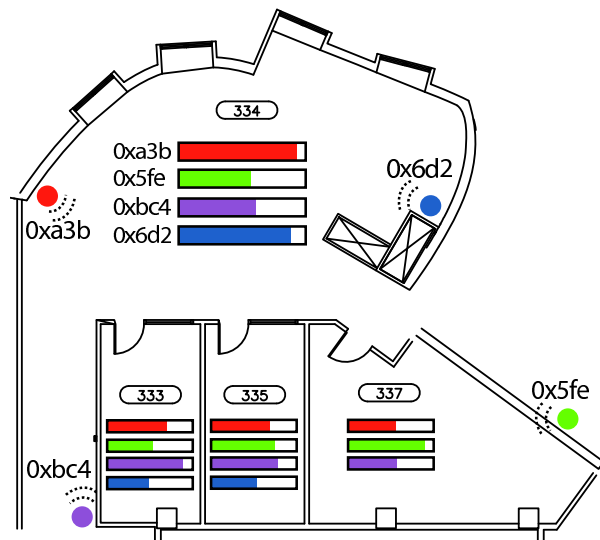


Figure 1: RF fingerprints. The bars in each space illustrate the RSSI from each in-range AP. Even though room 337 is physically close to access point 0x6d2 (in blue), RF signals from that AP are dampened by structures in the environment.

pothesis that the mobile device is located at location $l \in L$ given the available evidence. Given an observation o , the degree of belief, or *posterior probability*, of being in location l is given by:

$$P(l|o) = \frac{P(o|l)P(l)}{P(o)}. \quad (1)$$

According to this model, \hat{l} is the location with the maximum posterior probability. Since the *observation likelihood* $P(o)$ is fixed across all candidate locations, the decision rule becomes:

$$\hat{l} = \operatorname{argmax}_{l \in L} [P(o|l)P(l)]. \quad (2)$$

Therefore, it remains to estimate the *class-conditional probability*, $P(o|l)$, and the *prior probability*, $P(l)$, for each candidate location $l \in L$. For a location l , we assume that each signal strength s_i is conditionally independent of every other strength s_j for $j \neq i$, yielding the naïve Bayes model:

$$\hat{l} = \operatorname{argmax}_{l \in L} \left[\prod_i P(s_i|l)P(l) \right]. \quad (3)$$

Finally, the conditional distribution $P(s_i|l)$, which models signal strength per AP i for a given location l , can be estimated from labeled observations and stored in a database (§ 5.3). Similarly, the prior probability of each location $P(l)$ can be evaluated from labeled data.

2.2 Expert vs. Organic Surveying

To collect the fingerprints necessary for indoor localization, most previous approaches have utilized expert surveying. Methods based on expert surveying have practical, cultural and technical downsides preventing them from achieving widespread use. On the practical side, such methods have a high fixed cost, as they require an initial site survey to build and populate the signal strength map. This deployment burden typically requires a few person-days of careful

and spatially-comprehensive survey effort by skilled technicians. This approach faces a cultural barrier as well, as members of a community may feel reluctant to allow strangers into certain areas such as private offices. A technical challenge is that site survey data may become outdated over time, *e.g.* through access point reconfiguration, repositioning or replacement, each of which may degrade or invalidate subsequent location estimates.

These factors led to the development of user-generated, or *organic* localization systems, where the initial, comprehensive site survey is replaced with *ad hoc*, incremental collection of data by individual users [3–5, 35]. Organic localization merges the “survey” and “use” phases that were distinct in earlier work [2, 13] into a single state, where the users of the system are also prompted to construct the signal strength map. After a handful of early users populate the map for a building environment, a typical user will enjoy high-quality location discovery with minimal individual effort.

2.3 System Overview

We have developed the Organic Indoor Location (OIL) system for user-supported localization system. In OIL, client software running on each user’s mobile device periodically gathers a fingerprint of nearby wireless sources. This fingerprint is checked against a client-maintained signal strength map cache, populated asynchronously from a shared server. The software attempts to determine the user’s location. If it cannot, it prompts the user to indicate his/her current location on a displayed map. This on-the-fly surveying *binds* the fingerprint observed by the user to the relevant space. Each such addition to the signal strength map is soon reflected globally, resulting in improved localization for other users.

OIL has several distinctive characteristics compared with other user-generated localization systems. First, OIL is designed to run primarily as a daemon process on the client, estimating the device location and sharing this information with location-based applications and services running on the device. Managing localization as a background process means that we cannot rely on the user to proactively contribute or notice positioning mistakes. Instead, OIL must determine on its own when explicit human input is necessary. Second, we assume that human users, who are occasionally prompted to provide explicit location information, are supplied with a map. The map need not be precise, but it must be sufficient to convey a user’s position. This requirement does not appear to be overly burdensome because manual and automatic map-making tools are available [31, 39]. Lastly, we assume the existence of an accessible shared server that receives bind information from clients, and sends updated signal strength maps to clients. In a real-world setting, this server must include redundancy and be highly available. However, because storage and processing can be partitioned based on physical topology or RF sources, scaling the server is manageable. In addition, because there is client-side caching, devices can continue to localize even if they cannot contact the server, so long as the needed fingerprints are in the cache.

Given this outline of our organic localization system, a main challenge stems from the fact that the signal strength map, constructed from user-provided information, may be incomplete and may include erroneous data. In the next section we propose a mechanism for evaluating *confidence*

when inferring a specific location, which takes into account possible gaps in coverage. In addition, we suggest a policy, based on this confidence measure, in which users are prompted to provide additional binds in order to improve coverage. Section 4 discusses methods for identifying erroneous binds in order to improve localization accuracy.

3. VORONOI-BASED USER PROMPTING

In survey-based positioning, the survey provides a snapshot of ambient RF for all spaces where client positioning is desired. Based on the obtained signal strength map, a standard localizer can then find the space that matches the fingerprint observed by a client device with the highest probability. Organic positioning, however, begins with an empty database which is gradually populated with user-provided fingerprints. If the fingerprint database is empty or if the database does not include any of the RF sources that a client sees, then the organic localizer outputs an “unknown location” response. However, when the database is only partially populated, the localizer’s use of incomplete information may bias its predictions. For example, consider the extreme case where only a single bind has been made; if a database of known fingerprints contains a single location, the localizer will output that space as its prediction, even if the wireless scan observed by the device only slightly overlaps the fingerprint associated with that location. We therefore require a method for conveying the localizer’s spatial *confidence* in its output prediction. This confidence measure can be displayed along with the location estimate in a way that is intuitive to contributors and non-contributors alike. We can also use the confidence measure as the basis of a policy for requesting user input. For organic systems, in order to increase coverage, it is useful to occasionally prompt users for their location. However, there is a trade-off between providing imprecise estimates due to lack of coverage, and irritating users with too many bind requests, especially when the fingerprint database is only partially populated. When should a user be prompted with an explicit location request? Our system prompts whenever localizer confidence falls below a threshold.

During the development of OIL, we considered several prompting policies and their implications for coverage. The simplest policy is to prompt all users at regular intervals, regardless of their location or estimate confidence. However, this method was intrusive and conflicted with our goal of having knowledgeable “locals” be the primary data generators. An alternative policy was to prompt with frequency inversely proportional to coverage: as more spaces in a building are associated with fingerprints, user prompting decreases. However, we found via simulation that this approach resulted in a high false prompting rate, such that users were prompted in spaces that did not require it. A third policy – inserting interpolated, artificial fingerprints for unbound spaces – requires good coverage of nearby spaces to obtain meaningful results. We therefore arrived at a user prompting policy based on spatial uncertainty.

3.1 Spatial Uncertainty

In an organic localization system, if a user is in an unbound space, the most likely space to be selected by the localizer is the nearest bound space. This is because the RF fingerprint of the unbound space is likely to be similar to those of physically nearby spaces (Figure 2). If the localizer

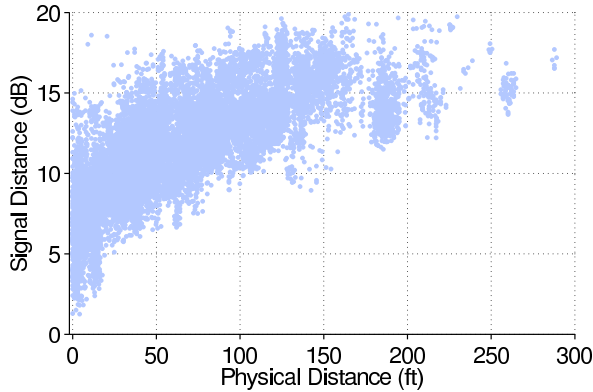


Figure 2: Signal distance vs. physical distance. As users add binds from more physically distant spaces, signal distance (Eq. 5) between binds increases.

predicts the user to be in a space with unbound neighbors, the user’s true location may be one of the surrounding spaces and *not* the bound space. To convey this spatial uncertainty, we can display to the user the set of possible spaces, including unbound ones. We can also prompt the user to bind in one of the unbound neighbors, as he or she is likely to be nearby.

In order to estimate uncertainty, we employ discrete Voronoi diagrams. In a standard, continuous two-dimensional Voronoi diagram [9], a set of point *sites* exists on a plane. Each site is associated with a *Voronoi cell* containing all points closer to that site than to any other site.

In our solution, bound *spaces* are sites, and unbound spaces become members of the cell associated with the nearest bound space (Figure 3). As a space shifts from being unbound to bound, it becomes a site and adds nearby unbound spaces to its newly-formed cell. The underlying intuition is that if a user is in an unbound space, the space most likely to be selected by the localizer will be the nearest bound space – the Voronoi site associated with the user’s true location. Therefore, the size of the bound space’s Voronoi cell naturally captures the spatial uncertainty associated with prediction of the bound space.

More formally, let L denote the set of all locations in a given floor, and B be the set of bound locations. Let L_c and B_c be sets of centroid coordinates of L and B , respectively. The *Voronoi diagram* for B_c is a planar subdivision of \mathbb{R}^2 in which every point x in the plane is assigned to $p \in B_c$ if $d(x, p) \leq d(x, p') \forall p' \in B_c, p' \neq p$. The set of points that are assigned to p is denoted as $V(p)$, the Voronoi cell of p .

For every bound location $b \in B$ with centroid p , we define a *spatial uncertainty region* $\mathcal{U}(b)$ to be a subset of L , as follows: every location $l \in L$ is assigned to one of the uncertainty regions, $\mathcal{U}(b)$, if the Euclidean distance from its centroid l_c is smaller to p than to any other $p' \in B_c$; equivalently, l_c belongs to the Voronoi region of p , $l_c \in V(p)$. In essence, we maintain a generalized Voronoi diagram as a collection of mutually disjoint spatial uncertainty regions.

For each spatial uncertainty region $\mathcal{U}(b)$ for a bound space b and its centroid p , we define two spatial uncertainty metrics: the *number of unbound locations*, $n(b)$, and the *maxi-*

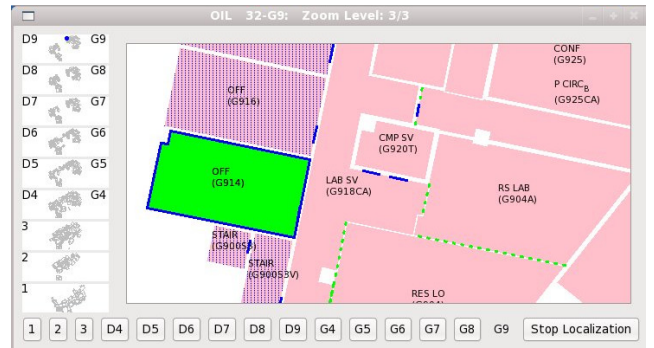


Figure 3: Spatial uncertainty. The user sees the bound space (green) as the Voronoi site, along with its Voronoi cell (stippled). This helps non-contributing users understand localization precision, and helps contributing users know which binds would improve coverage and/or accuracy.

mum uncertainty radius $r(b)$ defined as:

$$r(b) = \max_{l_c \in V(p)} d(l_c, p) \quad (4)$$

which is the maximum distance from the Voronoi site to the farthest unbound location in $\mathcal{U}(b)$. The number of unbound locations is used for the user prompting algorithm (Algorithm 1), giving the spatial uncertainty metric. The maximum uncertainty radius $r(b)$, used when drawing a circle centered on the corresponding bound space b , conveys uncertainty to the user.

As noted above, when a space changes from being unbound to bound, the floor’s Voronoi diagram must be updated. The update operation is efficient; it is linear in the number of spaces held by any adjacent cells. This update is performed on the server, then propagated to the clients.

3.2 User Prompting Algorithm

OIL requests user input in order to improve either coverage or accuracy. To this end, each client monitors a pair of hypotheses in determining whether further user input will improve the fingerprint database. Specifically, each time a location estimate is produced, the device evaluates the following questions:

1. If the user binds a nearby location, will the system’s *coverage* increase?
2. If the user binds his/her current location, will the system’s *accuracy* increase for this location?

The first question is answered by considering the spatial uncertainty of the current location estimate $n(b)$. High spatial uncertainty means that many nearby locations remain unbound; thus adding user input for nearby spaces will enhance the overall coverage of the fingerprint database. If the spatial uncertainty metric exceeds a threshold, the user is prompted for input. The second question is answered by checking whether recent location estimates for the user’s current location have been stable. Because the duration of each user contribution can be short and wireless signal strength can vary rapidly, a user in a space with a sparse fingerprint might experience unstable and inaccurate localization results. The user is also prompted in this case. Algorithm 1

Algorithm 1 User prompting algorithm. C_s^{max} and C_i^{max} are thresholds to determine (in)stability, and n^* is a pre-defined spatial uncertainty threshold. Note that prompting based on high spatial uncertainty occurs only when the location estimate is stable.

```

1: Input: location estimate  $l$ , uncertainty region  $\mathcal{U}(l)$ 
2: Output: prompt = { true, false }
3: States: stability counter  $C_s$ , instability counter  $C_i$ , previous location estimate  $l_p$ 
4: Initialization:  $C_s \leftarrow 0, C_i \leftarrow 0, l_p \leftarrow \text{Nil}$ 
5:
6: if  $l_p = \text{Nil}$  then
7:    $l_p \leftarrow l, \text{prompt} \leftarrow \text{false}, \text{return}$ 
8: else
9:   if  $l_p = l$  then
10:     $C_s \leftarrow C_s + 1, C_i \leftarrow \max\{C_i - 1, 0\}$ 
11:   else
12:     $C_i \leftarrow C_i + 1, C_s \leftarrow \max\{C_s - 1, 0\}$ 
13:   end if
14:   if  $C_s > C_s^{max}$  and  $n(l) > n^*$  then
15:     prompt  $\leftarrow$  true,  $C_s \leftarrow 0, C_i \leftarrow 0$ 
16:   else if  $C_i > C_i^{max}$  then
17:     prompt  $\leftarrow$  true,  $C_s \leftarrow 0, C_i \leftarrow 0$ 
18:   else
19:     prompt  $\leftarrow$  false,  $l_p \leftarrow l$ 
20:   end if
21: end if
22: return

```

shows the method used in OIL to answer these two questions. If the decision is to prompt the user, the user can see local coverage rates on the UI (Figure 3), and (a) decide whether to bind in the current space, (b) bind in an adjacent space, or (c) request not to be bothered again for a short or long duration (5 minutes and 4 hours respectively in our current implementation).

4. FILTERING ERRONEOUS USER INPUT

An organic localization system is expected to encounter some level of noisy user contributions. In particular, users will not always indicate the right room when they are prompted to make a “bind.” Early tests of our system showed that both ordinary and skilled users did indeed make mistakes.

Across organic location systems, mistaken contributions fall roughly into three categories: (1) when selecting the location from a map – as in OIL – the user may select the wrong room or floor; (2) when entering a user-defined space name – as in RedPin [5], for example – the user may type an incorrect or atypical name; and (3) while making a long interval bind, a user may move, polluting a bind with scans acquired in distinct spaces (see § 5.2.1). Identifying erroneous contributions is a key problem in organic localization because, without high-quality binds, database and positioning accuracy will suffer.

While we focus on the first type of error in our algorithm description and evaluation, variations on our method would also identify the other two types of errors. While we believe our method would also filter out uncoordinated malicious input, combating a pervasive attack – with many spoofed APs [36], for example – is beyond the scope of this paper.

Since location fingerprints are generated organically, there is no *a priori* model available for identifying correct binds.

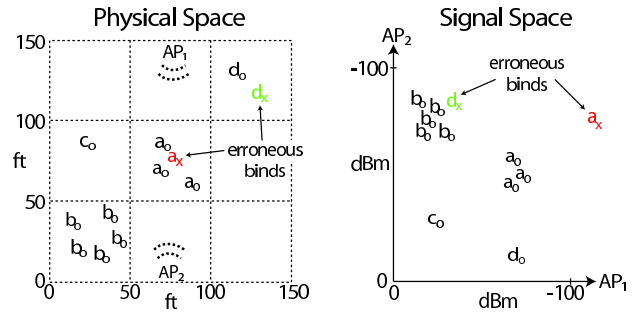


Figure 4: Correct binds made in the same physical space tend to cluster in signal space. By observing outliers in the signal space, we can detect and eliminate erroneous binds. Correct binds are denoted as *e.g.* a_o ; incorrect binds are denoted as *e.g.* a_x .

Error detection should therefore be managed in an unsupervised fashion. Our approach for handling erroneous user inputs hinges on outlier detection in *signal* space – the observed RF signal strengths for each access point (Figure 4). We rely on the fact that independent correct binds made at the same location are similar, and tend to cluster. We apply a clustering algorithm to detect outlier binds, which are suspected to be erroneous.

When a new user bind is received, it is processed in two steps. First, a clustering algorithm identifies, for the annotated location, a group of binds that are similar in signal space. Then, our proposed erroneous bind detection method tags the new bind as correct or erroneous. Later, localization incorporates only those binds tagged as correct. The fingerprint for each location, however, maintains all binds assigned to it (regardless of correctness), so that all data can be used to periodically reclassify clusters and outliers for that location. Sections 4.1 and 4.2 describe our approach in detail.

While we focus on erroneous bind detection in this section, we anticipate using a similar approach to address other problems in organic localization, such as detecting AP addition, deletion, and movement. Currently, we consider a collection of binds from a limited time window, to detect outliers within that window. The same approach can be employed *across* time windows of different granularity to detect true changes in the RF environment, such as AP movement. If clusters of correct binds formed in consecutive windows for the same location vary substantially, this is indicative of a change in the environment. In this case, localization accuracy may be increased by discarding old binds. We leave exploration of this issue to future work.

4.1 Erroneous Bind Detection

We represent a *bind* as a signal-strength vector in a k -dimensional signal space, with k the number of observed APs. Given multiple scans per bind, the bind’s i -th dimension is populated with the mean RSSI per AP_i . APs for which no signal is observed in the input scans (due to range or channel collision) are assigned a fixed value of -100 dBm.

Given multiple binds made at location l , our goal is to arrange these binds into meaningful clusters. We apply an agglomerative hierarchical clustering approach [17] to group binds by similarity. In this approach, clusters are succes-

Algorithm 2 Erroneous bind detection method.

```

1: Input: new bind  $b^N$  to location  $l$ , set of all binds  $\mathcal{B}_l$  for
   location  $l$ , neighbor locations  $\mathcal{N}_l$ 
2: Output: set of correct binds  $C_l^*$ 
3:
4: Add  $b^N$  to  $\mathcal{B}_l$ .
5: if  $|\mathcal{B}_l| = 1$  or  $\mathcal{N}_l = \emptyset$  then
6:    $C_l^* \leftarrow \mathcal{B}_l$  {No information for detection is available.}
7:   return
8: else
9:    $C_l \leftarrow$  Hierarchical clustering of  $\mathcal{B}_l$  (Eq. 5 and Eq. 6).
10:  if  $|\mathcal{C}_l| > 1$  then
11:     $C_l^* \leftarrow$  Identification of the correct cluster (Eq. 7).
12:  else
13:     $C_l^* \leftarrow C_l$ 
14:  end if
15: end if

```

sively merged in a bottom-up fashion, based on a similarity metric, until no clusters are similar enough to be merged. We define the distance (dissimilarity) metric between two bind vectors $b^s = (b_1^s, \dots, b_k^s)$ and $b^t = (b_1^t, \dots, b_k^t)$ as the normalized signal-space Euclidean distance:

$$d_s(b^s, b^t) = \left[\frac{1}{M} \sum_{i=1}^k (b_i^s - b_i^t)^2 \right]^{1/2} \quad (5)$$

where $M \leq k$ is the number of APs for which a signal has been detected in either bind b^s or b^t . The normalization term yields proper “per-AP” distances because, in any real setting, each bind will involve only a small subset of the observed APs.

Further, the distance between two clusters C_s and C_t , referred to as the *linkage function*, is defined as the average distance between inter-cluster bind pairs as follows:

$$D_S(C_s, C_t) = \frac{1}{|C_s||C_t|} \sum_{\substack{(b^s, b^t) \in \\ (C_s, C_t)}} d_s(b^s, b^t) \quad (6)$$

Clustering continues until the linkage function between all pairs of clusters falls below a pre-defined cut-off distance d^* . We use independent labeled data to obtain *a priori* knowledge about intra- and inter-location signal distance to set d^* . This procedure is described in detail in Section 4.2.

Once binds are grouped into clusters, the system must identify which cluster includes the correct binds (the rest of the clusters are assumed to contain erroneous binds). If we assume that most users make correct binds, it is natural to take the largest cluster as the correct one. However, in a previous deployment we found that more than 75% of the spaces had three or fewer associated binds [35]. When the organic system has not yet obtained good coverage, majority voting is not feasible. Instead, we use the observation that signal distance between two locations is positively correlated with physical distance between them (cf. Figure 2). Therefore, we identify the correct cluster of binds C_l^* given a set of bind clusters at location l , \mathcal{C}_l , according to the following criterion:

$$C_l^* = \underset{C \in \mathcal{C}_l}{\operatorname{argmin}} \sum_{m \in \mathcal{N}(l)} D_s(C, C_m^*) \quad (7)$$

where $\mathcal{N}(l)$ is the set of locations neighboring location l , and

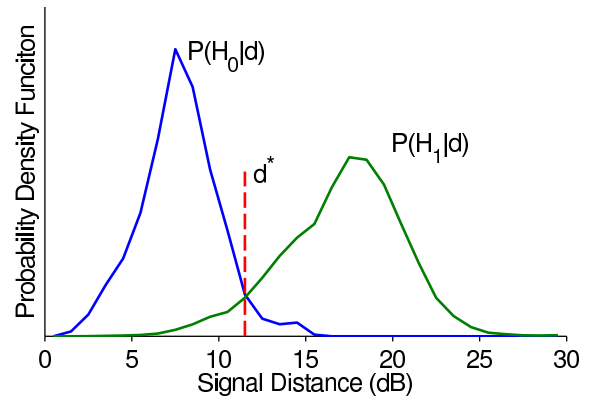


Figure 5: Determination of the clustering cut-off threshold. To achieve minimum probability of error, we choose \mathcal{H}_0 if $d < d^*$ and \mathcal{H}_1 otherwise.

C_m^* is the cluster of correct binds at the neighboring location m at the time of computation. Algorithm 2 outlines our approach.

4.2 Clustering Threshold Tuning

The quality of the clusters formed dictates the performance of erroneous bind detection. This section describes how the linkage function threshold d^* is determined.

During clustering, cluster pairs for which the value of the linkage function is larger than d^* are kept distinct. Thus, an optimal threshold is one that is effective at separating binds associated with different locations. If comprehensive labeled data is available, then the cut-off threshold can be set empirically, *e.g.* by cross-validation, to maximize the localizer performance. In practice, it is hard to obtain such large labeled datasets (indeed, this is the primary reason for pursuing organic localization). Instead, we assume that a handful of scans are available from a variety of locations. This data can be collected by the system designers or extracted from early user input. Using this data, we can cast the threshold tuning problem as a Bayesian decision problem, testing whether or not two binds originate from the same location.

Formally, suppose there are two binds b^s and b^t . We wish to evaluate the hypotheses:

$$\begin{aligned} \mathcal{H}_0 &: b^s \text{ and } b^t \text{ are from the same location; and} \\ \mathcal{H}_1 &: b^s \text{ and } b^t \text{ originate at different locations.} \end{aligned} \quad (8)$$

Let d denote the distance defined in Eq. 5. In clustering, $d < d^*$ implies that \mathcal{H}_0 holds; otherwise it is estimated that \mathcal{H}_0 is false (i.e., that \mathcal{H}_1 is true). We assume the cost of false positives is the same as false negatives. Then, according to the Bayesian decision rule, \mathcal{H}_0 is accepted if

$$P(\mathcal{H}_0|d) > P(\mathcal{H}_1|d) \iff \frac{P(d|\mathcal{H}_0)P(\mathcal{H}_0)}{P(d|\mathcal{H}_1)P(\mathcal{H}_1)} > 1. \quad (9)$$

That is, we will select \mathcal{H}_0 and judge that b^s and b^t are from the same location if $P(\mathcal{H}_0|d)/P(\mathcal{H}_1|d) > 1$ (Figure 5). A system designer can use the small amount of bind data to estimate the distributions $P(d|\mathcal{H}_0)$ and $P(d|\mathcal{H}_1)$. The optimal threshold, d^* , is the point at which the two *posterior* probability distributions cross. To estimate the posterior distribu-

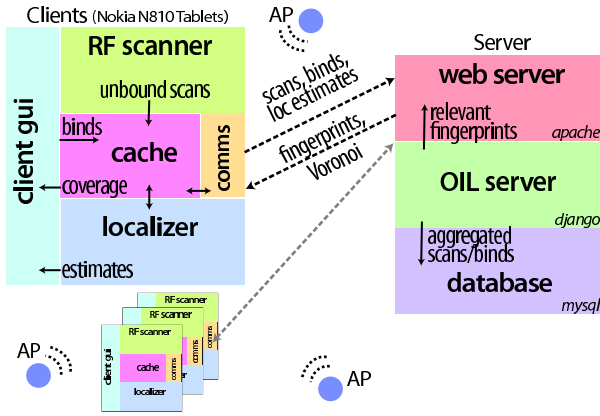


Figure 6: OIL System Architecture. OIL client software on a mobile device collects signal strengths from nearby APs, time-stamping and periodically transmitting them to an OIL server. There, they are aggregated, checked for accuracy, and sent out to other clients. The server updates Voronoi regions as part of the aggregation process.

tion, one must estimate the prior probabilities $P(\mathcal{H}_0)$ and $P(\mathcal{H}_1)$ (Eq. 9). We tested two different ways of modeling the hypothesis prior. One possibility is to consider the hypotheses equally likely; this leads to a Neyman-Pearson-type likelihood-ratio test, $P(d|\mathcal{H}_0)/P(d|\mathcal{H}_1) > 1$. Alternatively, one can model the assumption that most binds associated with any location l are correct; in other words, hypothesis \mathcal{H}_0 is more likely in an operative system. For example, if we assume that 90% of binds are correct and that the erroneous binds are not mutually correlated, then $P(\mathcal{H}_0) \approx 0.9^2$ and $P(\mathcal{H}_1) \approx 1 - 0.9^2$. When we tested our data with each of these two assumptions, we obtained $d^* = 12$ dB and $d^* = 15$ dB, respectively. Each of these values was used as a stopping criterion in the hierarchical clustering algorithm (§ 6.4). The first value closely matches Bhasker *et al.*'s empirical closeness threshold [4].

5. IMPLEMENTATION

This section presents our implementation of OIL, describing in detail its client and server architecture (Sections 5.1 and 5.2). Because the success of OIL relies on facilitating user input, building a responsive interface was important, so that users could quickly view the effects of data that they had contributed. This led us to performing local cache updates and performing localization computations on the user's device (Figure 6). User updates are subsequently migrated to the server and then to other users. In our experiments, the observed latency of data updates across users was in the range of thirty seconds to five minutes. While not the focus of our work, another benefit of on-device localization is improved user privacy, compared with commercial system implementations, *e.g.* Skyhook [32], which perform localization computations on the server and thus can track users. In order to perform on-device operations efficiently, our implementation aims at optimizing the storage and computation resources required at the client. We cover the details of our localization algorithm in Section 5.3.

5.1 Server

The cooperative nature of organic localization necessitates a repository for bind aggregation and exchange. We currently implement the repository as a stand-alone server with which the OIL client exchanges data in order to share contributions and facilitate location discovery. The server's main roles are: (1) to store scans and binds reported by OIL clients and (2) to provide OIL clients with fingerprints for nearby locations, and corresponding Voronoi region information. Our current implementation also records clients' localizer estimates. By comparing client estimates with user binds, we can produce a real-time stream of ground truth measurements for evaluation purposes.

5.1.1 Building Fingerprints and Voronoi Diagrams

The primary role of the server is to aggregate scans and binds into fingerprints and Voronoi diagrams. Using a lightweight protocol, the server receives scans and associated binds from clients. The server updates the appropriate fingerprint with the newly-received scan. In addition, the floor's Voronoi diagram is updated with bind information when appropriate. Changes to Voronoi diagrams are piggy-backed onto normal communications with clients so that they can update any invalid cached Voronoi diagram information. In order to differentiate data from multiple users, each message to the server is stamped with the client's MAC address. (Client privacy could be increased by using temporary cookies in place of client MAC addresses.)

5.1.2 Fingerprint Pre-Fetching

To enable client localization and caching, the server's API provides access to location fingerprints and their associated Voronoi regions. Our system uses a mechanism that limits the data sent to a subset of relevant locations. This ensures that the localization service does not overwhelm the client devices, which have limited computation and storage resources.

We determine location relevancy using a similarity metric, where our goal is to identify a set of "nearby" locations. Considering the collection of scans sent by the user as an input fingerprint, we evaluate its similarity to fingerprints in the dataset. Formally, the server calculates a similarity score $c(l, q) \in [0, 1]$ for each candidate location $l \in L$ and the client-provided fingerprint q . Let A and B represent the set of MAC addresses in the fingerprints of l and q , respectively. We compute the following similarity score:

$$c(l, q) \equiv \frac{1}{2} \left(\frac{|A \cap B|}{|A \cup B|} + \frac{|A \cap B|}{|B|} \right)$$

The first term in the formula is the Jaccard index [16]. Since B is typically much smaller than A , we added the second term to model asymmetric similarity. A location l is passed to the user if $c(l, q) \geq \theta$, where θ is a predefined threshold. In our experiments we found a threshold of $\theta = \frac{1}{3}$ to be effective. Overall, this mechanism allowed us to reduce the amount of data cached at the client without excluding relevant fingerprints.

5.2 Client

Our design goals dictate that our client infrastructure facilitate the contribution of user data, communicate this data to the server promptly, and maintain a cache of location fingerprints for localization.

We implemented our client in Python on Nokia N810 Internet Tablets. Users are presented with a clickable map, where polygonal space contours and space names have been automatically extracted from AutoCAD files [39].

5.2.1 Data collection and sharing

The client collects three main types of data. They are grouped by type and sent to the server periodically.

- A **scan** is a set of access point MAC addresses and time-stamped signal strengths as observed by the device. Scans are collected at about $\frac{1}{3}$ Hertz.
- A **location estimate** is a time-stamped estimate generated by the localization algorithm. Our prototype estimates its position once every fifteen seconds.
- An **interval bind** is a user-selected location name together with start, end, and time-of-bind time stamps, as described below.

In earlier versions of our system, binds were momentary: they merely associated the most recent scan sent by the device with the user-indicated location name. However, because at least a few minutes of scanning is required to produce stable fingerprints, instantaneous binds required significant user effort to cover a single space. Since contributors were often in the same space for minutes at a time or longer, we altered the binding user interface to allow the user to specify a *time interval* in addition to a location. Our contributing users all had the same type of device. However, we do not anticipate device heterogeneity to be a major issue, since scan data can be calibrated to a common scale [13].

A user creates an interval bind by indicating a space on the map, how long s/he has been in that location, and how long s/he intends to remain in that location. All scans made within the imputed time interval are then associated with the selected location. The interval bind also includes the actual time of the user action. This allows the fingerprint aggregation process to reconstruct a complete ordering for each user’s set of binds and handle contradictions. We introduced interval binds in earlier work [35]; Bolliger *et al.* independently developed a similar mechanism called “interval labeling” [6].

Our interface also allows users to cancel previously-made binds and mark spaces as inaccessible, suppressing further prompts for user input in these spaces.

5.3 Localization Algorithm

Our implementation included two significant changes to the standard Bayesian localization method presented in Section 2.1.1. First, because the locations of organic binds can be highly skewed – *e.g.*, near users’ offices – we set the prior distribution $P(l)$ to a uniform distribution over all candidate locations. Second, we approximate the conditional probability of signal strength, $P(s_i|l)$, using a histogram of signal strengths that have been observed in location l for access point AP_i . Given a histogram of the observed signal strengths at location l , a common approach to computing the conditional probability of the signal strength is to use the maximum likelihood estimate (MLE), *i.e.*, relative counts. However, because the histograms in an organic system can be sparse, MLE can be inaccurate. Instead, we use a smoothed *m-estimate*, which provides a regularization effect [10]. Specifically, let k denote the number of readings

for signal strength s_i for access point AP_i at location l , and n denote the number of total readings. Then the *m-estimate* is given by:

$$\hat{P}(s_i|l) = \frac{k + mp_k(s_i)}{n + m} \quad (10)$$

where $p_k(s_i)$ encodes the prior probability for the probability we wish to evaluate, and m determines how much weight we attribute to the prior $p_k(s_i)$. Observations (scans) transform the prior histogram into a histogram specific to that location; the effect of the prior histogram will vanish as observations are added [8]. Our implementation set $p_k(s_i)$ to a uniform distribution for simplicity, and m to the number of effective histogram bins (we used $m = 70$ since signal strength typically ranges from -94 to -25 dBm).

6. EVALUATION

We plan to deploy OIL broadly on our campus and eventually to larger areas. For this reason, we examined the algorithmic foci of the paper both in detailed simulation, where we could explore parameter changes easily, and in a live deployment, where we could gather real user input and feedback.

We first describe how the live deployment was conducted and provide overall results from it, including changes in coverage and accuracy over time. Section 6.3 examines our claims that Voronoi diagram-based prompting improves coverage rates and helps explain localization precision. Section 6.4 evaluates our erroneous bind detector with organic user input. Lastly, Section 6.5 examines how users participated in the deployment and how their behavior matches that of users in other collaborative settings.

6.1 Test Deployment

We launched a test deployment of OIL, inviting building residents to participate. Nineteen people participated, including two administrators, three people from a non-technical department, and four members of our group. We gave each participant a mobile tablet with the OIL client and building map installed and showed them how to make interval binds and operate the client in general. We asked users to respond to the tablet’s prompts when they were able to do so, but not to go out of their way to provide coverage. Users were encouraged to take the tablets with them, including out of the building if they wished.

At the start of the deployment, we also installed fourteen stationary “spot check” tablets in different rooms in the building. We did not make any binds on these tablets, but left them to run and report their location estimates back to the server.

Table 1 summarizes the users’ contributions as of the end of the study.

6.2 System Utility

We studied the interval binds and logged location estimates to characterize the utility of the system according to several metrics.

The **Coverage** metric characterizes the fraction of map spaces to which readings had been bound by the user community (Figure 7). One day into the deployment, our user group had covered 57, or about 4.1%, of the 1,373 spaces in the corpus. Four days later, that figure had grown to 95 (about 6.9%). By the end of the deployment, almost

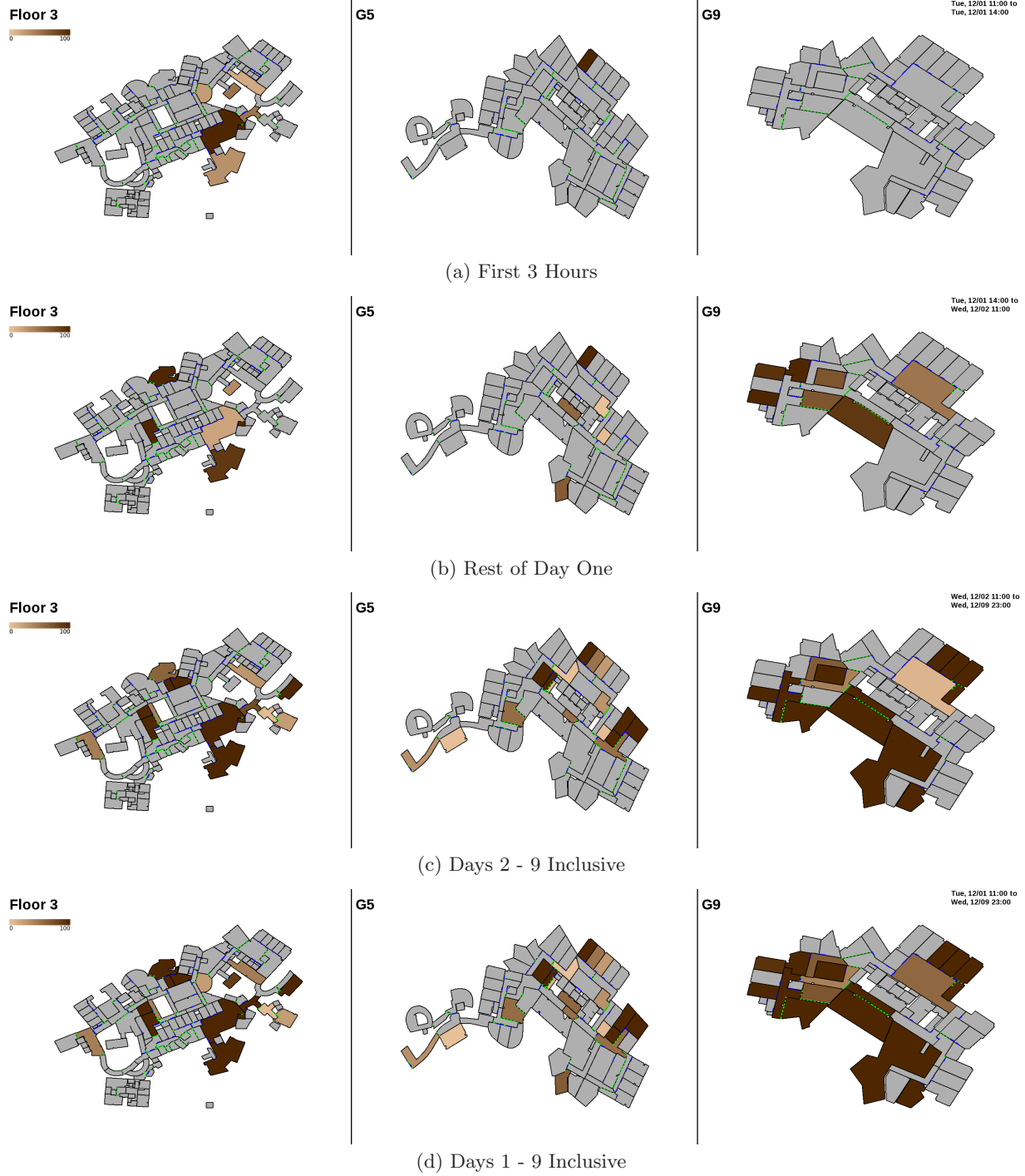


Figure 7: Organic contributions to Floors 3, G5 and G9 during the first three hours (a); hours 4-24 of the first day (b); days 2-9 (c); and for the entire nine-day deployment (d). Color indicates number of bound scans per space; uncolored spaces accumulated no bound scans during the displayed interval.

Map Spaces	1,373
Contributing Users	19
Bind Intervals (from users)	604
Scans (from devices)	1,142,812
Bound Scans	108,418 (9.4%)
Spaces with Bound Scans	116 (8.4%)

Table 1: Statistics for our 9-day test deployment.

all covered spaces had sufficiently many readings to support accurate localization. More broadly, in a building with approximately a thousand daily occupants, some nineteen users – fewer than two percent of the occupants – covered almost ten percent of the building in just over a week.

The **User Accuracy** metric characterizes the quality of the location estimate computed by participants’ tablets. The ground truth for these estimates was established by looking at location estimates directly before a user made a bind. At the start of the deployment, the user accuracy was zero, as there were no locations known to the system. The user accuracy increased thereafter, to the point that on the final day of the deployment the mean error between the centroid of the estimated room and the centroid of the true room was less than 4.5m. (The average distance from one space to its closest adjacent space is 5.3m.) This error is comparable to error rates seen in survey-driven indoor deployments [13].

The **Spot Check Accuracy** metric characterizes the quality of the location estimates computed by the spot check tablets. Of these 14 tablets, 11 of their rooms were bound at some point during the deployment, yielding an accuracy comparable to that seen by mobile users (Figure 8(a)).

Figure 8(b) shows the distribution of bind-minutes per space per day. The data show that as bind-minutes increase, mean localization error decreases.

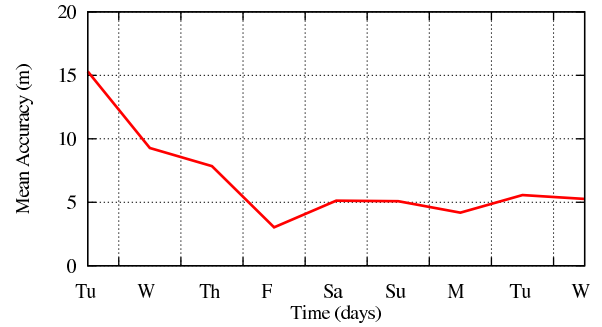
6.3 Voronoi-based User Prompting

During initial planning for our deployment, we had several ideas for improving coverage rates. We evaluated them via simulation (§ 6.3.1), selected the most promising for deployment, then interviewed participants for feedback (§ 6.3.2).

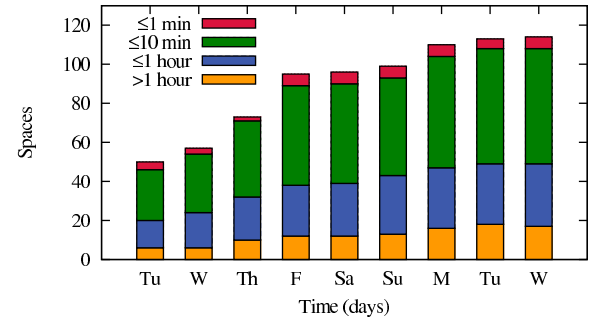
6.3.1 Improving Coverage Rates

Our basic proposals for requesting user input included *periodic prompting*, where users are prompted at regular intervals, and *inverse coverage*, where prompting rates decline as more spaces on a floor are bound. These methods are simple, but do not direct users to where user input is actually needed. Instead, the dynamic Voronoi diagram approach captures the fact that a user is likely to be located in, or near, a space that requires more coverage. Our opinions were split on how we should expect contributors to act: were users active (willing to move to an adjacent space), or passive (willing to provide input only where they were when prompted)?

We compared four proposals via a simple simulation. We moved users randomly across an artificial 100×100 grid floor. To test the performance of prompting methods independently of a localization algorithm, we assume perfect accuracy. We also assume that users always bind when prompted. We let our “active” Voronoi users be willing to move to adjacent locations with probability $\frac{1}{2}$ per request.



(a) Spot check Accuracy



(b) Cumulative Per-Space Bind-Minutes

Figure 8: Organic growth: overall localization accuracy grew directly with user contributions.

We examined coverage per number of user prompts, because the purpose of efficient user prompting is to improve coverage while minimizing user effort. Figure 9 shows the results of running each method 300 times. Voronoi-based prompting outperforms the other methods, especially if the active user model is assumed. Periodic prompting suffers from prompting too much – irritating users – because it does not consider the current coverage or location estimate. While the inverse coverage method adapts to increasing global coverage, it does not do as well as the Voronoi-based methods. Voronoi-based prompting considers both coverage and local spatial uncertainty, reducing unnecessary user prompts. When users provide binds in nearby locations, as assumed with the active user model, the effectiveness of Voronoi prompting further increases. Our OIL client asked users to bind in adjacent rooms, in effect suggesting to users that they adopt the active user model.

6.3.2 Conveying Spatial Uncertainty

After completing our test deployment, we interviewed participants about the Voronoi prompting mechanism. Overall, the responses were mixed. Of the top two contributors (see Figure 12) one said the prompts were the main reason that she made so many binds. She also found the Voronoi regions, as in Figure 3, were useful for quickly locating the room that she was in as well as assessing how well the tablet knew her current location. The other top contributor said that the prompting mechanism had no effect on his behavior. One less active user found the prompting irritating as he rarely left his office, had little interest in making binds, and continued to be prompted. Although he could have marked the unbound spaces surrounding his office as inaccessible, he did not do so – but he did turn off prompting.

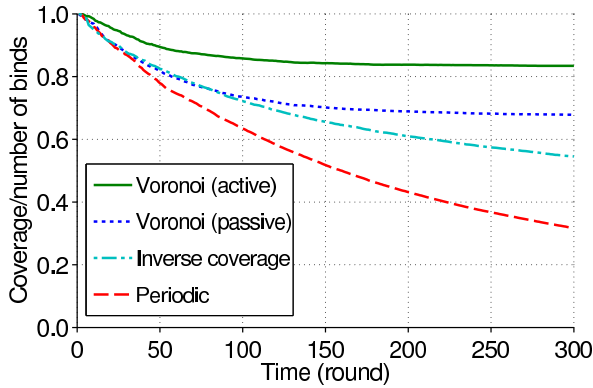


Figure 9: Voronoi-based user prompting significantly increases coverage at low user effort as compared to other methods in our simulations. Active users, who are willing to move to an adjacent un-bound space to contribute a bind, provide the most benefit.

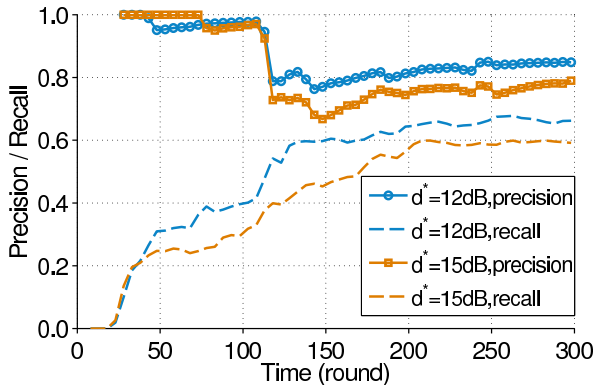


Figure 10: After each space’s fingerprint acquires a sufficient number of binds, detection success significantly increases. In addition, conservative detection (12dB) improved both precision and recall in our experiments.

These observations suggest that while Voronoi prompting can be helpful, it could be made more adaptive and personalized. For users who make few binds or do not bind when prompted, the system could prompt them less, whereas the system could continue to prompt users if it appears to be advantageous to do so.

6.4 Erroneous Bind Detection

We studied the performance of our erroneous bind detector. First, we examined the effect of time on the outlier detection. Next, we evaluated the end-to-end effect on accuracy as we varied the fraction of erroneous binds. Both evaluations were performed using simulation on organic data from an earlier OIL deployment, which had 16 users and lasted for 20 days [35].

6.4.1 Effect of Time on Detection

We used a discrete event simulator to see if erroneous binds could be detected after each space’s fingerprint con-

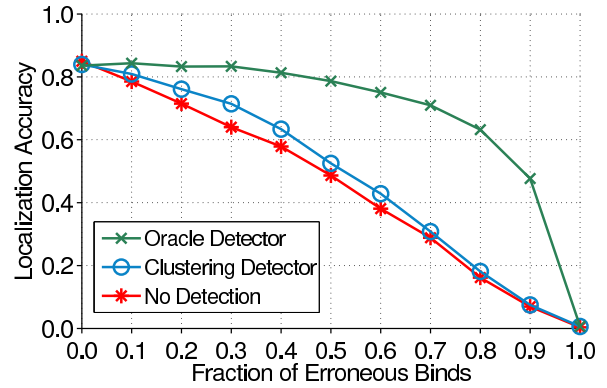


Figure 11: Filtering Erroneous Binds. We varied the fraction p_e of erroneous binds given to the system and computed overall system accuracy.

tained enough binds to measure a valid signal distance to its physical neighbors. We also varied the clustering threshold (Section 4.2) to observe its effect on overall detection. At each round of simulation, a correct bind is taken from the data set. Before being added to the fingerprint database, its location is changed to a random one with probability p_e , emulating an erroneous bind. Then, the bind is added to the fingerprint of the potentially-erroneous space.

Figure 10 shows the *precision* and *recall* for conservative (12dB) and lenient (15dB) thresholds, which correspond to physical separations of approximately 50 and 125 feet, respectively. Precision is the fraction of truly erroneous binds detected over all binds. Recall is the fraction of erroneous binds identified over all erroneous binds. We used $p_e = 0.1$ for this test.

The data show that the bind detection algorithm’s performance increases as the fingerprint database becomes more populated. In detecting erroneous binds for a certain location l , initially the algorithm just accepts binds until there is sufficient information to make an estimate. As fingerprints of the neighboring spaces become more populated, the algorithm more readily identifies inconsistent binds. In this sense, the fingerprint database is self-repairing. After 300 rounds, the conservative threshold achieved a precision of 0.85 and recall of 0.68.

6.4.2 Localization Accuracy

We next wanted to examine the effect that different levels of error-proneness among contributors would have on overall accuracy. To do so, we varied the fraction of erroneous binds presented to the system, from none to all wrong, and measured localization accuracy. Figure 11 shows localization performance in three cases: an “oracle”(perfect) detector, our clustering detector, and no detection. The data demonstrate why filtering out erroneous binds is essential: erroneous user input greatly compromises accuracy. The proposed detector enhances localization accuracy by 5–9% over a wide range of p_e . Unsurprisingly, at high error rates, the algorithm results in low accuracy because the fingerprints of neighbor locations also contain many erroneous binds.

We are in the process of adding erroneous bind detection to our live OIL deployment.

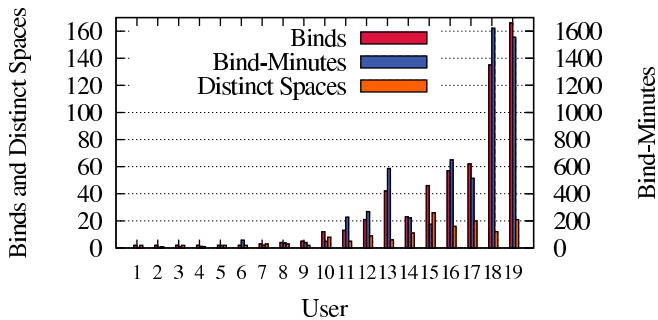


Figure 12: Fingerprints per user. Like Wikipedia and other resources dependent on user contributions, a handful of users were significantly more active contributors than the rest.

6.5 User Participation

We studied the resulting logged binds in order to characterize user behavior. One notable example of an organically-grown resource is the Wikipedia on-line knowledge repository [42]. Previous studies of user-contributed repositories have described a 1/9/90 classification of users by contribution level [40]. In our setting, we expected a few users to perform at least one large-scale survey or to contribute data nearly everywhere they go. Some users might perform a few small-scale surveys, for instance walking the corridors on one floor of their building or providing updates after a change to the local network. The remaining majority of users might not contribute any data at all; these “free riders” would enjoy the service based on the efforts of the more active minority.

While the size of our user study is too small to state conclusively that this breakdown of user behavior would persist at larger scale, we did in fact observe an approximation of this behavior in the number of fingerprints contributed per user (Figure 12). Other slices of the data, such as the distinct spaces covered per user, showed a similar distribution. Our previous deployment showed the same effect [35].

This suggests that the majority of data used by an organic location service will not be from a uniform cross-section of users. Instead, the data will more likely consist of multiple, overlapping, small-scale amateur “surveys.” That is, because this 1% will become relatively experienced at contributing data, their contributions will, in general, be high-quality – more like those of expert surveyors. A preliminary conclusion is that the majority of the data in the database used by an organic location service will likely resemble multiple, overlapping, small-scale surveys. Alternatively, as Reddy *et al.* propose, we could alter user prompting and recruiting strategies based on user geographic and temporal coverage patterns [29], assuming these patterns could be kept private. The main advantage of our service is that it can integrate both kinds of contributions without interfering with the user experience.

7. RELATED WORK

We review four areas of prior work: RF-beacons and fingerprints, Voronoi diagrams, robustness and clustering, and user-generated systems, all in the context of localization.

7.1 RF Beacons and Fingerprints

Early work on localization that relied on existing RF infrastructure assumed that the location of RF transmitters was known and fixed. This research centered on determining a client’s location *relative* to these *beacons*. For example, Hightower *et al.* focused on removing the need for fixed infrastructure through rapid, flexible RF deployments [14]. A major difficulty in these approaches is that reflection, diffraction, multipath fading, and the presence of new objects, *e.g.* people, often stymied signal models and, in turn, distance and angle estimates [25]. The RADAR system circumvented the problems of signal modeling and triangulation by shifting to RF fingerprints [2]. As constructing fingerprints requires a large amount of human effort, methods to reduce the training burden while maintaining accuracy were proposed, *e.g.*, by Madigan *et al.* [24] and Lim *et al.* [23]. More recent work has focused on ubiquitous alternatives to RF: for example, power lines [26], cell towers [21, 37], and ambient optical, acoustic, and motion attributes [1]. Like Krumm and Platt [20] and Haerberlen *et al.* [13], we focus on room-granularity positioning, rather than on gridding methods.

7.2 Voronoi Diagrams

The Voronoi diagram is one of the fundamental geometric structures in computational geometry and has been used in many other different fields including computer graphics, robotics, physics, and sensor networks [9]. In the context of indoor positioning, Swangmuang and Krishnamurthy used closely related proximity graphs – Delaunay triangulation, Gabriel graphs, and relative neighborhood graphs – to obtain an analytical model for the localization error probability of a given fingerprint [33]. In contrast, we use the Voronoi diagram to approximate the spatial uncertainty that naturally arises from organic user contributions.

7.3 Robustness and Clustering

As localization using RF infrastructure has become widespread, researchers recently investigated its susceptibility to spoofing attacks. Chen *et al.* examined the robustness of several localization algorithms against signal strength attenuation attacks [7]. Tippenhauer *et al.* [36] and Saroiu *et al.* [30] studied various types of attacks including AP impersonation and spoofing as well as injection and corruption of the fingerprint database. Although our focus is on user input errors, malicious attacks are closely related because, in both cases, incorrect wireless signals can be entered into the fingerprint database.

Cluster analysis has been widely used for anomaly detection. For example, Portnoy *et al.* use clustering to detect anomalies in network traffic [27]. Clustering has been used for several purposes in localization systems: for example, Swangmuang and Krishnamurthy use it to improve performance prediction [34] and Lemelson *et al.* use clustering as a measure for error prediction [22]. To our knowledge, clustering has not previously been used to detect erroneous user input to localization systems.

7.4 Organic Localization

Recently, the idea of relying primarily on user input to create a location database has been proposed for both outdoor and indoor localization. Outdoors, GPS coordinates can be used to annotate user input and build the fingerprint-to-place mapping. This process, called *wardriving* [21], gener-

ates fingerprints that can later be used for location determination by devices that lack GPS but have WiFi [32, 41]. Wardriving can be considered a form of organic data collection, but its dependence on GPS limits it to outdoor use.

User input has also been employed in indoor positioning systems. ActiveCampus [4, 12] uses a prediction-correction mechanism: first, the system builds a coarse-grained fingerprint, then users can correct a location estimate by providing a “virtual AP”. OIL is different from ActiveCampus system in three ways: (1) OIL constructs a fingerprint database from scratch, relying only on user input; (2) it maintains probabilistic fingerprints, which are more expressive; and (3) it works without knowing AP physical locations.

Bolliger [5] developed the RedPin localization system which uses WiFi, GSM, and Bluetooth as sensing devices. Like OIL, RedPin does not require an expensive training phase and generates location fingerprints from user input. Krumm and Hinckley’s NearMe infers proximity – not absolute location – by comparing user-generated WiFi signatures [19]. Barry *et al.* [3] conducted a year-long study of a user-trained localization system and showed its utility. None of these systems addressed challenges associated with organic input, such as spatial uncertainty and labeling errors.

8. CONCLUSION

While the concept of organically constructing a localization system is simple, building a working system in practice presents significant challenges. This paper addressed two issues that arise in “growing” an organic indoor location system: modeling uncertainty, and handling erroneous user input. We proposed a method, based on Voronoi diagrams, that suggests to active contributors what spaces around them need coverage, and conveys to all users the level of localization precision they can expect in their current vicinity. We also described a method that watches for erroneous user contributions and automatically discounts them. This method, based on outlier detection through clustering, allows an organic positioning system to maintain its accuracy over time. We examined the proposed methods in simulation and through a test deployment. We found that they contributed positively to coverage, and yielded average localization accuracy on the order of meters, comparable to that of survey-driven indoor deployments.

In the future, we plan to continue examining the role of time in organic indoor localization. In addition to discerning erroneous binds, our bind clustering method appears generalizable to other problems in organic localization. For example, we anticipate using it to detect addition, deletion, and movement of APs. Another interesting topic would be to investigate combining contributions from both trusted and untrusted surveyors. Building on the ActiveCampus approach [12], less trusted, organic refinements could complement an initial, trusted survey of mostly public spaces.

Acknowledgements

We thank: Nokia Research Center Cambridge for their financial support of, and technical engagement with, this effort; the early adopters in the Stata Center who volunteered to test the system; and Russell Ryan for his contributions to the implementation of the OIL server.

9. REFERENCES

- [1] M. Azizyan, I. Constandache, and R. R. Choudhury. SurroundSense: Mobile Phone Localization via Ambience Fingerprinting. In *Proc. MobiCom*, pages 261–272, Beijing, China, Sept. 2009.
- [2] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. In *Proc. INFOCOM*, pages 775–784, Tel Aviv, Israel, Mar. 2000.
- [3] A. Barry, B. Fischer, and M. Chang. A Long-Duration Study of User-Trained 802.11 Localization. In *Proc. Mobile Entity Localization and Tracking in GPS-less Environments*, pages 197–212, Orlando, FL, Sept. 2009.
- [4] E. S. Bhasker, S. W. Brown, and W. G. Griswold. Employing User Feedback for Fast, Accurate, Low-Maintenance Geolocation. In *Proc. PerCom*, pages 111–120, Orlando, FL, Mar. 2004.
- [5] P. Bolliger. RedPin: Adaptive, Zero-Configuration Indoor Localization. In *Proc. Location and Context Awareness*, pages 55–60, Oberpfaffenhofen, Germany, Sept. 2008.
- [6] P. Bolliger, K. Partridge, M. Chu, and M. Langheinrich. Improving Location Fingerprinting through Motion Detection and Asynchronous Interval Labeling. In *Proc. Location and Context Awareness*, pages 37–51, Tokyo, Japan, May 2009.
- [7] Y. Chen, K. Kleisouris, X. Li, W. Trappe, and R. Martin. The Robustness of Localization Algorithms to Signal Strength Attacks: A Comparative Study. In *Proc. Distributed Computing in Sensor Systems*, pages 546–563, San Francisco, CA, June 2006.
- [8] J. Cussens. Bayes and Pseudo-Bayes Estimates of Conditional Probabilities and Their Reliability. In *Proc. European Conference on Machine Learning*, pages 136–152, Vienna, Austria, Apr. 1993.
- [9] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, Third edition, 1997.
- [10] S. Džeroski, B. Cestnik, and I. Petrovski. Using the M-Estimate in Rule Induction. *Journal of Computing and Information Technology*, 1(1):37–46, 1993.
- [11] Ekahau Positioning Engine. <http://www.ekahau.com>.
- [12] W. G. Griswold, P. Shanahan, S. W. Brown, R. T. Boyer, M. Ratto, R. B. Shapiro, and T. M. Truong. ActiveCampus: Experiments in Community-Oriented Ubiquitous Computing. *IEEE Computer*, 37(10):73–81, 2004.
- [13] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki. Practical Robust Localization over Large-Scale 802.11 Wireless Networks. In *Proc. MobiCom*, pages 70–84, Philadelphia, PA, Sept. 2004.
- [14] J. Hightower, R. Want, and G. Borriello. SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength. Technical Report UW CSE 2000-02-02, University of Washington, Feb. 2000.
- [15] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *GPS Theory and Practice*. Springer, 1997.
- [16] P. Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura.

- Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.
- [17] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.
- [18] C. Jernigan, C. Bayley, J. Lin, and C. Wright. Locale. <http://people.csail.mit.edu/hal/mobile-apps-spring-08/>, May 2008.
- [19] J. Krumm and K. Hinckley. The NearMe Wireless Proximity Server. In *Proc. UbiComp*, pages 283–300, Nottingham, England, Sept. 2004.
- [20] J. Krumm and J. Platt. Minimizing Calibration Effort for an Indoor 802.11 Device Location Measurement System. Technical Report MSR-TR-03-82, Microsoft Research, Nov. 2003.
- [21] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. E. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. N. Schilit. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Proc. Pervasives*, pages 116–133, Munich, Germany, May 2005.
- [22] H. Lemelson, M. B. Kjærsgaard, R. Hansen, and T. King. Error Estimation for Indoor 802.11 Location Fingerprinting. In *Proc. Location and Context Awareness*, pages 138–155, Tokyo, Japan, May 2009.
- [23] H. Lim, L.-C. Kung, J. C. Hou, and H. Luo. Zero-Configuration, Robust Indoor Localization: Theory and Experimentation. In *Proc. INFOCOM*, pages 1–12, Barcelona, Spain, Apr. 2006.
- [24] D. Madigan, E. Einahrawy, R. P. Martin, W.-H. Ju, P. Krishnan, and A. S. Krishnakumar. Bayesian Indoor Positioning Systems. In *Proc. INFOCOM*, volume 2, pages 1217–1227, Miami, FL, Mar. 2005.
- [25] K. Pahlavan, P. Krishnamurthy, and J. Beneat. Wideband Radio Propagation Modeling for Indoor Geolocation Applications. *IEEE Communications Magazine*, 36(4):60–65, Apr. 1998.
- [26] S. N. Patel, K. N. Truong, and G. D. Abowd. PowerLine Positioning: A Practical Sub-Room-Level Indoor Location System for Domestic Use. In *Proc. UbiComp*, pages 441–458, Orange County, CA, Sept. 2006.
- [27] L. Portnoy, E. Eskin, and S. J. Stolfo. Intrusion Detection with Unlabeled Data Using Clustering. In *Proc. ACM CSS Workshop on Data Mining Applied to Security*, Philadelphia, PA, Nov. 2001.
- [28] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proc. MobiCom*, pages 32–43, Boston, MA, Aug. 2000.
- [29] S. Reddy, K. Shilton, J. Burke, D. Estrin, M. H. Hansen, and M. B. Srivastava. Using Context Annotated Mobility Profiles to Recruit Data Collectors in Participatory Sensing. In *Proc. Location and Context Awareness*, pages 52–69, Tokyo, Japan, May 2009.
- [30] S. Saroiu and A. Wolman. Enabling New Mobile Applications with Location Proofs. In *Proc. HotMobile*, Santa Cruz, CA, Feb. 2009.
- [31] SketchUp. <http://sketchup.google.com>.
- [32] Skyhook Wireless. <http://www.skyhookwireless.com>.
- [33] N. Swangmuang and P. Krishnamurthy. Location Fingerprint Analyses Toward Efficient Indoor Positioning. In *Proc. PerCom*, pages 100–109, Hong Kong, Mar. 2008.
- [34] N. Swangmuang and P. V. Krishnamurthy. On Clustering RSS Fingerprints for Improving Scalability of Performance Prediction of Indoor Positioning Systems. In *Proc. Mobile Entity Localization and Tracking in GPS-less Environments*, pages 61–66, San Francisco, CA, Sept. 2008.
- [35] S. Teller, J. Battat, B. Chawrow, D. Curtis, R. Ryan, J. Ledlie, and J. Hicks. Organic Indoor Location Discovery. Technical Report CSAIL TR-2008-075, Massachusetts Institute of Technology, Dec. 2008.
- [36] N. O. Tippenhauer, K. B. Rasmussen, C. Pöpper, and S. Čapkun. Attacks on public WLAN-based positioning systems. In *Proc. MobiSys*, pages 29–40, Kraków, Poland, June 2009.
- [37] W. ur Rehman, E. de Lara, and S. Saroiu. CILoS: a CDMA Indoor Localization System. In *Proc. UbiComp*, pages 104–113, Seoul, Korea, Sept. 2008.
- [38] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, 1992.
- [39] E. Whiting, J. Battat, and S. Teller. Generating a Topological Model of Multi-Building Environments from Floorplans. In *Proc. Computer-Aided Architectural Design Futures*, pages 115–128, July 2007.
- [40] S. Whittaker, L. G. Terveen, W. C. Hill, and L. Cherny. The Dynamics of Mass Interaction. In *Proc. CSCW*, pages 257–264, Seattle, WA, Nov. 1998.
- [41] Wigle: Wireless Geographic Logging Engine. <http://wigle.net>.
- [42] Wikipedia. <http://www.wikipedia.org>.