A Long-Duration Study of User-Trained 802.11 Localization

Andrew Barry, Benjamin Fisher, and Mark L. Chang

F. W. Olin College of Engineering, Needham, MA 02492
{andy, benjamin.fisher}@students.olin.edu,
mark.chang@olin.edu

Abstract. We present an indoor wireless localization system that is capable of room-level localization based solely on 802.11 network signal strengths and user-supplied training data. Our system naturally gathers dense data in places that users frequent while ignoring unvisited areas. By utilizing users, we create a comprehensive localization system that requires little off-line operation and no access to private locations to train. We have operated the system for over a year with more than 200 users working on a variety of laptops. To encourage use, we have implemented a live map that shows user locations in real-time, allowing for quick and easy friend-finding and lost-laptop recovery abilities. Through the system's life we have collected over 8,700 training points and performed over 1,000,000 localizations. We find that the system can localize to within 10 meters in 94% of cases.

1 Introduction

Computerized localization, the automatic determination of position, will augment existing applications and provide opportunities for new growth. One can easily imagine a phone, computer, or other device changing behavior based on location. A phone might disable its ringer when in a conference or classroom. Calendar reminders would only appear if a user was not already in the event's location. A laptop could automatically select the closest printer when printing. Finding a colleague would be as simple as looking up a phone number.

Localization abilities have spawned a number of companies including GPS navigation [5][16], asset tracking [4][17][20], and E911 systems [6]. The most common form, GPS, performs well in many instances, but it cannot achieve good accuracy indoors. To provide indoor localization, researchers have examined the use of dedicated hardware including ultrasound, IR, and RF beacons. Most of these platforms provide good resolution, but often have high installation, maintenance, and usage costs. With the advent of 802.11 wireless networking, researchers have turned to utilizing wireless access points as fixed RF beacons. This method mitigates the high hardware and installation costs of earlier systems, but often requires a substantial amount of off-line training, or the collection signal strength samples in many locations. Here we describe a large scale deployment of a system that uses 802.11 access points to localize, but transfers the training burden to the system's users, providing a cheap, fast, accurate, and low maintenance method for automated indoor localization.

We use the following terminology to describe our system: a wireless *fingerprint* denotes the signal strengths of surrounding access points at a given location. A *bind* is the act of associating a fingerprint with a location. An *update* is a location scan and localization calculation. Fingerprints are collected automatically, binds are performed by users, and updates are performed automatically or by user request.

2 Related Work

Location-aware computing is not new. Perhaps the best-known location-discovery platform is GPS, which uses U.S. government satellites to compute latitude and longitude [8]. While the high costs associated with GPS systems have disappeared, a receiver can only obtain a location with a clear sky-view. Moreover, GPS experiences substantial drift and is often not accurate enough to obtain room-level localization. Another set of commonly available localization systems serve the FCC's E911 initiatives [6]. These systems focus on approximately 100 meter accuracy and thus, like GPS, are of limited utility in indoor, room-level environments.

The first indoor location-aware systems, such as Active Badge and MIT's Cricket, succeed with specialized hardware [13][18]. Active Badge uses wearable transmitters and a network of sensors to gather location information and report it back to a server. The Cricket system uses a combination of RF and ultrasound to provide accurate and private location data. These systems avoid training, but instead require a substantial hardware installation phase. Both Active Badge and Cricket require location-bound hardware that necessitates prior access by trained personnel to each desired localization area. This installation and the associated time and hardware costs limit these systems' wide-scale use.

As 802.11 networks became common, researchers began utilizing existing hardware to compute location. Microsoft's RADAR and later Haeberlen *et al.* show success in using the signal strength of 802.11 nodes to determine fine-grained indoor location [1][7]. These and similar systems [11][14] require specialized training to create a database of location–signal strength tuples. Training demands a substantial upfront effort and physical access to all of the desired areas. Moreover, after some time, the training data needs to be refreshed to account for changes in the environment and access point locations.

To reduce the expense of training, Intel Research demonstrates an algorithm that can estimate location with only minimal data by expanding its known area with continued use [10]. While the self-mapping algorithm costs little to implement, it requires a significant period of time to gain acceptable localization accuracy and coverage. Moreover, multiple radio configurations complicate the implementation of a shared-training system. Wardriving can be used to seed the algorithm, but those data are often not dense enough for accurate indoor localization.

Both Bolliger and Teller *et al.* introduce crowdsourcing methods that allow users to train and correct the system [2][15]. Teller's work, conducted in parallel with our own, is similar to the system described here although on a smaller scale in time, space, and number of users. They studied 16 trained users limited to a single building with a specialized platform for only 20 days. Here we present a year-long deployment of a similar crowdsourcing method with over 200 untrained users spanning five buildings

operating on personal laptops. Bolliger's system engineering is also similar to our own, but, again, he does not present results from a significant deployment.

In the commercial space, Ubisense has deployed accurate localization based on UWB signals in industrial environments [17]. Ekahau has been working on 802.11 localization for a number of years [4]. Skyhook Wireless has combined crowdsourced data with a substantial set of training data to improve their worldwide 802.11 localization system [14]. Navizon uses exclusively user-produced data, but like Skyhook, their system focuses on outdoor localization [12].

3 Architecture

3.1 Overview

Like [2] and [15], we use a client-server architecture to enable fast, accurate localization and provide a mechanism for feedback. To localize, clients perform an *update* in which they collect wireless 802.11 signal strength information (a *fingerprint*) to send to a server. The server computes the client's location and sends the estimate back to the client for optional user review. The server also updates the friend-finding interface with the client's new location in case another user wants to locate the first. When the client receives the location estimate, it offers the user an opportunity to confirm or correct it (Figure 1). If the user chooses to take this opportunity (*binding* a fingerprint), the client sends the new ground-truth data back to the server, which stores the record for use in all future localization computations.

Our system architecture is similar to MIT's Organic Indoor Location system (OIL) [15] with a server-based localizer and without client-side caching. For brevity, we will examine only the novel aspects of our system in depth and direct the reader to the OIL implementation for other details. To compute locations, we use a Euclidean distance algorithm, comparable to RADAR's Nearest Neighbor in Signal Space (NNSS) [1]. We have implemented the algorithm in SQL, interfacing PHP and wxPython clients that run on Windows, Linux, and Mac operating systems. We are planning to implement clients for smartphones and PDAs in the near future.

3.2 Deployment Site

A number of our design decisions were driven by the context in which our system was deployed. We developed and continue to run our localizer at Olin College, a small residential engineering school near Boston. Olin houses its entire 300-student population on campus with five buildings in total, encompassing more than 300,000 square feet. Our primary user base is students, with faculty and staff comprising less than 3% of users. Each student owns an institution-issued laptop although some use their own systems. Since each entering class has a slightly newer laptop model, we find a variety of similar but distinct radio/antenna combinations on campus.

In asset tracking, medical, or warehouse situations, one might choose to localize every few seconds, but students remain in the same place, be it a classroom, library, or residence hall, for extended periods of time. Thus, we chose to localize once every five

minutes, an unscientifically chosen but reasonable interval, in order to preserve system resources. Finally, we note that with the system providing a useful service, users have an incentive to provide accurate data and very little impetus to falsify locations.

3.3 Client Interface

The user interface is designed to be as non-invasive as possible. Since the system runs on personal laptops, it must have a small resource footprint minimizing CPU, memory, and power consumption. The client autostarts minimized in the system task tray and, to encourage use, never prompts the user without request, even when the location estimate is known to be poor. We found that we could collect enough training data without interrupting users and annoyed far fewer people in the process. If users want to train the system or access others' locations, double-clicking the task-tray icon brings up our deliberately simplistic interface (Figure 1). When an update is performed, the client displays its location estimation in question form, prompting a training response. The user can then accept the given location, choose from a list of likely locations, or create a new point. Figures 1–3 show typical GUI screens.

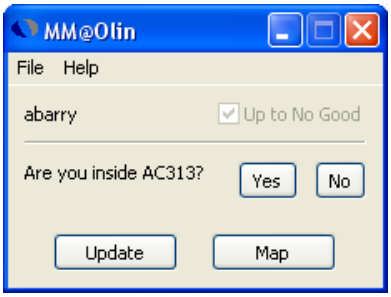


Fig. 1. Typical interface. The client has localized and asks the user to confirm its estimate. The upper left displays a username and the upper right contains a humorous checkbox.

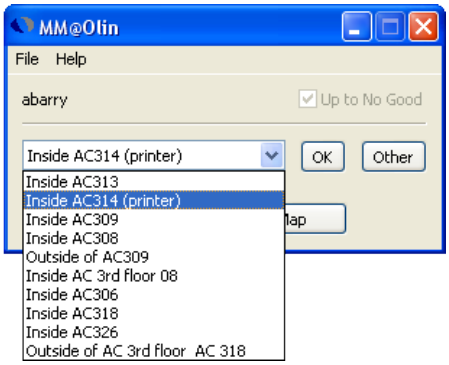


Fig. 2. Training interface. The user has clicked "No" in Figure 1 and is now prompted with nearby locations. Clicking "Other" allows the user to create a new location point.

Should the user determine that the localization is not correct, he or she can create a new location point. The client prompts for the building, floor, and a text name, suggesting a room number or descriptive phrase for the text entry (Figure 3). Once a user has entered those data, a labeled map appears allowing the user to select where the new location will appear visually. We found that making this process intuitive is key to ensuring the success of a crowdsourced data collection application.

New point creation is challenging for both the user and GUI designer. Most users are unable to correctly identify their location on an unlabeled blueprint, so the user inter-

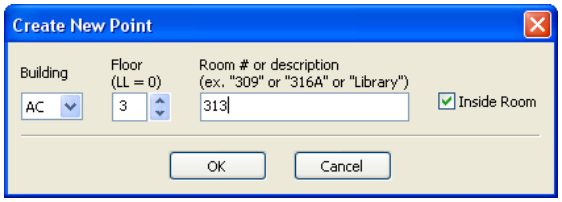


Fig. 3. New location creation interface. The user has clicked "Other" in Figure 2 and is now prompted to enter the details of his or her location. After clicking "OK," the user will be prompted to select the location on a map of the local area.

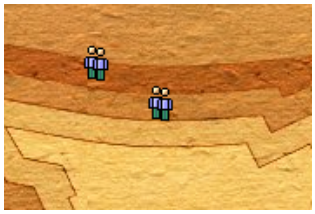


Fig. 4. Floor interface. Darker edges indicate a lower floor. In this case, two people are located on the ground floor and two more are located on the first floor.

face must be copiously labeled to prevent errors. We chose to allow users to customize location names, making their descriptions much more useful to others. For example, instead of calling a room "335," users labeled it "3rd floor lounge."

Allowing for free-form input, however, provided substantial possibilities for error. Although "lounge" is a natural input, it is not useful without context. To obtain both flexibility and accuracy, we constrain building and floor choices while allowing for a textual description. Thus, points have reliable context information and custom labels.

Finally, we note that custom naming lowers the entry barrier for new localization systems. Where we need only rough building diagrams, other systems require fully digitized blueprints or CAD models to generate an initial map.

3.4 Friend-Finding Service

To motivate users to train the system, we implemented a friend-finding service that publishes user locations on a map-like interface. The goals of the service are threefold: allow users to quickly and easily locate specific people, display all users' locations on one screen, and act as an advertisement for the project.

To satisfy our goals, the interface must allow for both searching and browsing while maintaining an attractive look and feel (Figure 5). To encourage use and help explain the concept, we themed the project as the "Marauder's Map at Olin," a reference to a magic map that displays people's locations in the popular Harry Potter series. While this theming might seem trivial, we found that it was critical to building and maintaining a user base. The map theme helped people understand why the service is useful and encouraged them to share with their friends.

Displaying hundreds of users on multiple floors proved to be a challenge. Early tests showed that users clustered near the edges of buildings, so we expanded our building representations to show the edges repeatedly, utilizing the new space to indicate vertical displacement and avoid icon overlap (Figure 4). This is not a perfect solution, as some buildings have popular locations in their centers, but we found these overlaps to be relatively minor.

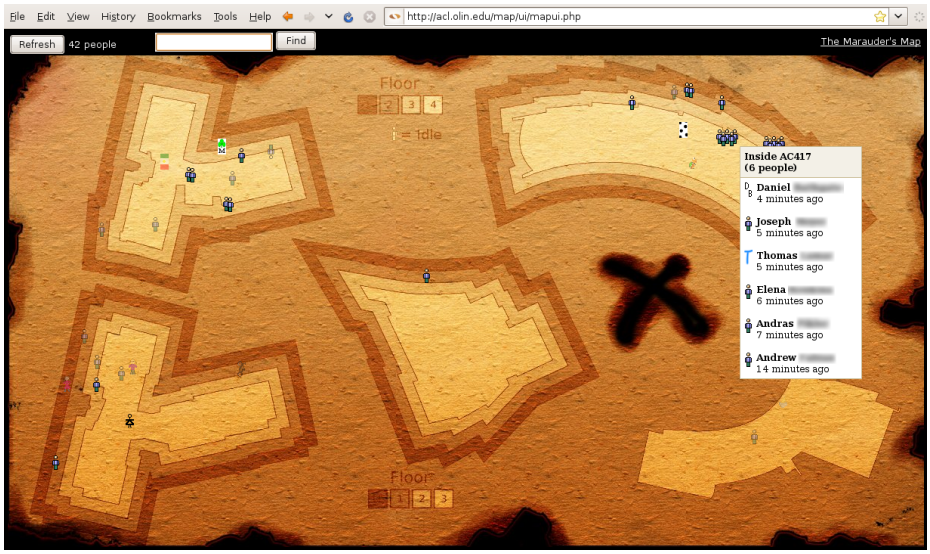


Fig. 5. Friend-finding webpage frontend. The interface is themed like an old magical map. The user has moved the mouse over a cluster of people, prompting a drop-down list of location, names, and update times.

3.5 Privacy

Privacy is a concern in any localization system. Given that the primary application of our implementation is a friend-finding service, we found that concerned users were aware of the implications and simply chose not to participate. Some users requested the ability to remove their location report at any time, a feature we implemented. All of the system's services reside only on internal servers to ensure that location information is not published outside of our institution.

4 Approach to Crowdsourcing

4.1 Motivation

The primary motivation for crowdsourced data is the reduction in time required to train the localizer. Moreover, with crowdsourcing, users provide most of the data while the system is already localizing, reducing the time before the localizer can be used. When training, researchers found that gaining access to private and semi-private spaces (offices, residence hall rooms, etc.) was difficult and awkward, a problem that user-trained systems avoid [7].

A pre-trained system requires retraining to account for changes in the environment, but a user-trained system is continuously updated with no overhead. In addition, crowdsourced training naturally produces data that are dense in places that are commonly

visited. Users tend to bind in places they frequent, causing common locations to have dense data. Because our localizer treats each bind separately, it weights common locations more heavily, resulting in a natural location-frequency dependency.

Finally, traditionally trained systems suffer from a conflict between coverage, the number of distinct locations with data, and accuracy, the measure of how often the localizer chooses the correct location. If the system is aware of many often unoccupied locations, it will suffer from a decrease in accuracy. Crowdsourcing helps mitigate the problem by naturally ignoring rooms that users do not frequent. For example, there are small, narrow trash rooms in each wing of the residence halls that were never bound in our system (Figure 6). A traditionally trained system might incorrectly place a user in one of these rooms, even though the probability of a user being located there is very small. Our system will always avoid these areas because no users have bothered to train them.

4.2 Initial Training

While almost all of our data are provided by users, we found that a minimally trained system was important to convince users that the software was compelling. Typically, this type of initial training takes about 1-3 minutes per location [7]. At this rate, a system covering over 350 spaces would take approximately 16 person-hours to train manually, but we can train the system to a minimally usable state, in about 1.5 hours. With that training, we create a sparse map including hallways and common areas, allowing for reasonable (within 10-20 meters) estimates that support further training by users. This training is easy to perform because all fingerprints can be collected in public areas and with relative infrequency. To train our system we simply walked down most hallways and bound one or two points per hall.

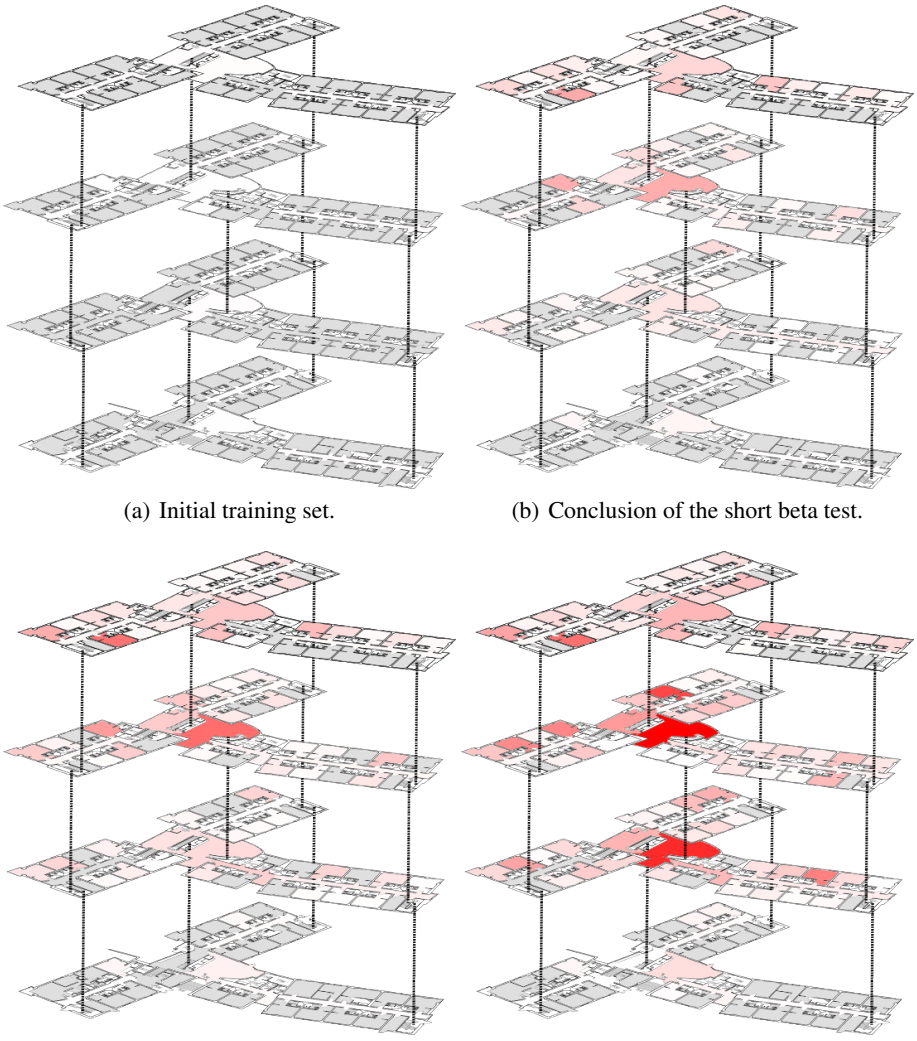
5 Results

After a short beta test, we deployed our system campus-wide at Olin College in April 2008 and have continued operations for over a year. We announced the project with an email to a common list in which we explained the concept and encouraged users to download and run the client. To date, we have had more than 200 unique users, 8,700 binds (95% of users contributing), and over 1,000,000 location updates.

5.1 System Accuracy

A localization system's performance is determined by both *coverage* and *accuracy*. Coverage measures how much of the deployment area has associated fingerprints while accuracy measures of how often the localizer reports the correct location. We first discuss coverage and then proceed to examine our localizer's accuracy.

At the beginning of deployment, only our initial survey supplied data, so we started with poor coverage, especially in private rooms (Figure 6(a)). We found that within 30 days of launch, our coverage stabilized at a reasonably complete level, with over 75% of all known locations at the year's end having already been bound. Coverage progression



(a) Initial training set.

(b) Conclusion of the short beta test.

(c) 30 days after launch.

(d) One year after launch.

Fig. 6. Map of training density in one of the five deployment buildings. Gray indicates no training data. Light to dark red indicates progressively more fingerprints for that space. Note that common areas (center of the building) have far more data than individual's rooms.

for one building can be see in Figure 6. Other buildings show similar patterns, although places students are unlikely to spend time, such as faculty offices, never achieve good coverage.

Our second metric is system accuracy. Accuracy starts poor and improves with the number of binds. To measure accuracy, we note that we should not simply test the system at random locations. Real accuracy is determined by how often the localizer correctly estimates *users'* locations. To use the aforementioned example, the localizer's performance in a small trash room is of little importance to true accuracy.

To test our system in this manner, we chose to survey our users during deployment. After the system had reached a steady state, we emailed our users asking them to report the localizer's accuracy at that moment. Thus, users opened the client, checked its current estimate against their real location, and reported performance in meter ranges (ie within 0-5 meters, 5-10 meters, etc.). We received 57 reports representing more than half of all online users within 8 hours. While these reports were self-selected based on which users chose to respond, we do not believe this bias has skewed our data measurably.

Figure 7 shows localization errors. We find that we localize to within 5 meters in 69.9% of attempts and to within 10 meters in 94.9% of cases. This accuracy is approximately equal to other published systems, although it does not achieve the performance of Haeberlen *et al.*'s calibrated or King *et al.*'s dense data techniques [7][9][15].

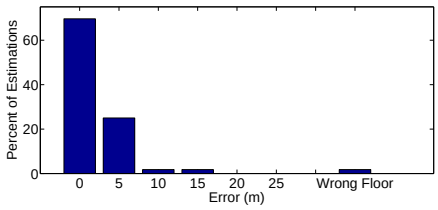


Fig. 7. Localization error. We determined error by asking users to perform spot checks in their current location. We find that we localize correctly in 69.9% of attempts and are within 10 meters in 94.9% of cases. We localize to to the wrong floor only 1.8% of the time.

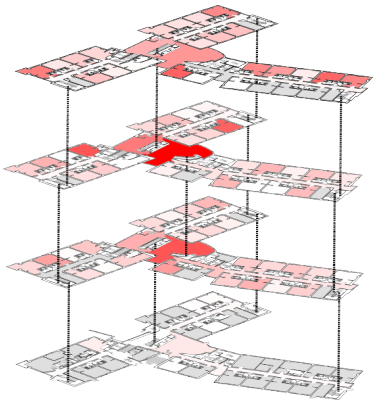


Fig. 8. Combined density of location reports for one year in one residence hall. Clearly, common areas are visited far more often than individuals' rooms. No students live on the first floor so we are not surprised to see few reports in that location.

5.2 System Vulnerabilities

We note a number of conditions that degrade our performance. First, during our deployment, a large fraction of the campus's access points received firmware modifications that resulted in a change in MAC address. The system does not recognize the new configurations and assumes that none of the old access points exist. While our architecture is designed to easily adapt to new access points with a single dictionary replacement, we found that losing these access points did not significantly degrade performance, so we allowed the system to operate without intervention.

In addition to changing MAC addresses, network administrators moved a small fraction of access points to improve wireless performance. While users have retrained the system, this movement continues to degrade our performance. To mitigate this issue, we are considering a weighting system that favors new fingerprints, marginalizing old and possibly outdated data. The design of the weighting system requires further study to determine if it should universally downgrade old data or only ignore old fingerprints when newer ones are available for a location. The first implementation would cause the localizer to favor newly bound points while ignoring old fingerprints, effectively reducing coverage over time. The second implementation retains coverage, but might not reflect newer user-movement patterns.

A third potential degradation of performance is the automatic and manual gain control on access points. During our deployment both automatic and manual power adjustments were made, but the system showed no noticeable decrease in performance. Without a clear indication that this was causing degradation of localizations, we have not spent the engineering resources to study this effect further. It is worth noting, however, that these power changes affect only the local area around the modified access points and simply shift the localizer's tendency closer or further from the respective wireless node.

A fourth potential degradation of performance is our support of a wide range of laptops, including Mac hardware and at least four models of the Dell Latitude D-series. We do not currently account for radio/antenna configurations, although the system has a natural correction based on user binding patterns, as described below.

In the user space, we note that user-generated data are not as reliable as professionally generated scans. It is difficult to determine how often localization errors are due to user error when training the system or from signal strength variations resulting from dynamic objects, antenna orientation, radio chipset, access point power fluctuations, or a host of other phenomenon. General accuracy statistics provide some insight for an upper bound on errors, but we have not yet developed a metric to study these errors explicitly. A time-based weighting on data, as examined above, would provide some mitigation for mistaken inputs, allowing them to be corrected as new, better data become available.

Finally, our system does not address the possibility of malicious users, beyond marking each bind with an identifier that allows the elimination of all binds by a particular user. While this is a concern, we are not aware of a single instance of such activity. Moreover, to be effective, malicious users would need to create a substantial number of false data points in many locations to overwhelm the existing fingerprint set. Many binds in one location would overwhelm that particular location, but the damage would

be confined to the local area. To be truly successful, a malicious user would need to bind incorrect data throughout the system's coverage, a far more difficult task.

In the event that malicious activity becomes an issue, we have discussed the implementation of weights based on how similar new data are to existing fingerprints. In this way, outliers are rendered harmless automatically. If a significant number of outliers were added to the system by different users, the weights would begin to skew towards those new fingerprints, accounting for dramatic environment changes such as access point relocation.

5.3 User Behavior

After release, we collected about 71 binds per day and within 2 months our data set had grown to 27 times the initial training, a collective effort of approximately 25 person-hours in ideal conditions. While training rates decreased as accuracy increased, users still train the system over one year later. Figure 9 shows these trends in database size over time.

As expected, we collected more fingerprints in common and public locations than in private rooms. The median number of binds per room is 7 and the maximum is 305, which occurs in a residence hall lounge (a common socializing space with couches and a TV.) 17% of known locations have only one bind and 53% have 10 or fewer. As the system's coverage grew, the number of new locations bound quickly decreased. Our initial survey bound 16% of total locations, beta testers bound 48%, and our general user base bound the remaining 36%. Thus, we find that users are far more likely to pick an existing location than they are to bind a new one.

In addition, we find that the user contribution profile is similar to other mass-interaction applications [15][19]. A few enthusiastic users bind an inordinate number of times. Interestingly, these users do not update their location as often as we might expect, with very little correlation between bind and update frequencies.

We hypothesized that our data collection method would result in dense data in frequented places, and our expectations are confirmed. We see this in the similarity between Figures 8 and 6(d) which show where locations are reported and bound, respectively. These data confirm our second hypothesis, that users bind in places that they, individually, frequent. 51.2% of all location updates occur in places that users had bound themselves. In other words, half the time a user is in a place where he or she has contributed data. Thus, we confirm that one of the primary advantages of crowdsourced data collection is that users are willing to train where they frequent and that they tend to reside places where their own data are best.

5.4 Application Use

While we have not yet performed a formal study, we feel that users are satisfied with the localization application. In one year we logged over 14,000 friend-finding page loads, averaging well above 50 hits per day during both fall and spring terms. We note that not only did users utilize the system at launch, they continue to use the service throughout its deployment. We also note that with 100 active users, we are localizing about 1/3 of the student population's systems.

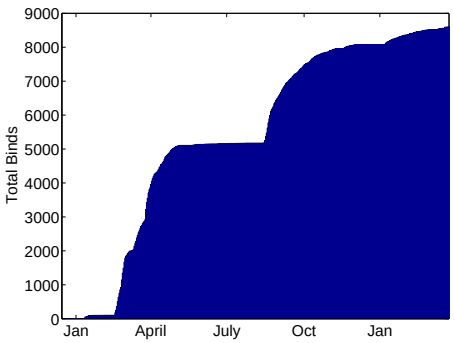


Fig. 9. Fingerprint database size over time. Training rates were steady after release and have reduced as the system became more accurate. We are not surprised to see little data added during the summer term when students are not on campus.

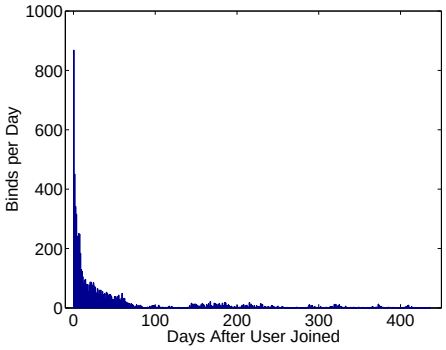


Fig. 10. All binds sorted by days since the binding user first localized. Clearly, users bind a significant amount in their first day and steadily less throughout their usage. We find that 11.5% of all binds occur on a user's first day.

Our last method of evaluating user satisfaction is purely anecdotal. Users tell us that they enjoy using the system and have only rarely contacted us with complaints, despite our contact information being readily available. Perhaps our favorite example of users' creativity is using the system in a scavenger hunt. An on-campus business group was running a promotion in which they planted clues advertising the whereabouts of free product samples. To create one of the clues, the group, unbeknown to us, managed to emulate a client and change the reported name and user icon to their name and logo, respectively. They then used the reported location to advertise where free samples could be found. Months later, when performing data analysis, a developer found the odd entry and finally traced it to the group who reported that it was the most popular clue in their entire game.

6 Detailed Analysis

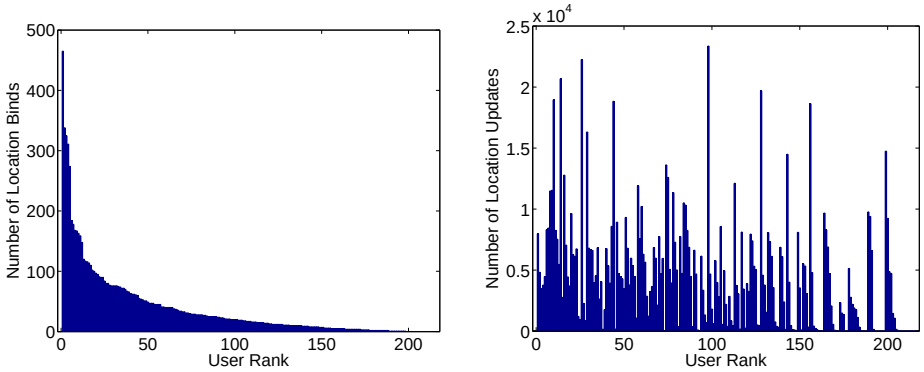
We now discuss the system's usage in detail. We examine trends in both when and which users train the system and the profile of where users localize. We find that new users are the primary providers of ground-truth data and that, despite our high participation rate, a small set of users provide most of the system's data. These often-training users, however, are neither more or less likely to localize than their non-binding peers.

6.1 Training Rates

Users tend to bind data in their first few days of use and rapidly stop providing ground-truth information thereafter. Throughout the system's life, 43% of all binds occur within 10 days of the binding user's first application use. We offer two explanations for this

phenomenon. First, users may train the system because of the novelty of providing data. Once that novelty fades, users become less interested and only localize. Second, student movement patterns do not change substantially from one day to the next. Once a user has trained his or her habitual places, the system may not require further training for accurate use and thus exclusively localizing is acceptable. Figure 10 shows bind occurrences sorted by days since adoption.

While new users train the system more often than longtime users, we find that some provide a disproportionate amount of data. Figure 11(a) shows a sorted profile of users based on number of binds. In our system, 20% of users provide 66% of the data. As mentioned above, we notice that there is little correlation between users who provide data and those who localize often. This is manifested in the difference between 11(a) and 11(b).



(a) Quantity of binds for each user. Users are sorted and assigned an ID by decreasing number of binds.

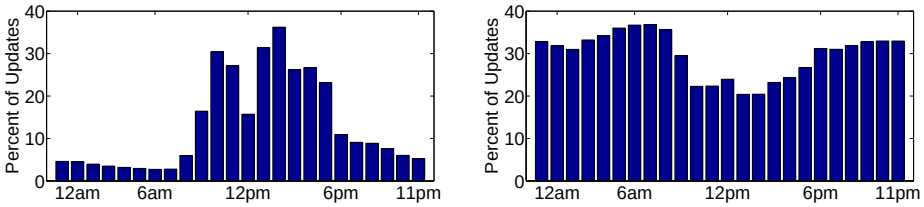
(b) Number of updates per user. Users have the same ID as in (a).

Fig. 11. Number of binds and updates for each user. Users are sorted by decreasing number of binds. In (a) we see that 20% of users bind 66% of the ground-truth data. In (b) we find that those often-binding users are not significantly more likely to update their location often, showing that the primary producers of data are not necessarily the primary consumers.

6.2 Application Usage Patterns

We find cyclic patterns in usage, both throughout each day and throughout the year. As expected, we find that users cluster in academic buildings during the day while returning to residence halls at night. Figure 12(a) shows localizations on an hourly basis in an academic building while 12(b) displays a similar plot for a residence hall.

We find that long-term training patterns are driven by new users. It appears that while localizing holds longevity as a useful service, the novelty of binding data fades,



(a) Academic building. (b) Residence hall.

Fig. 12. Percentage of localizations per hour in both an academic building and a residence hall. As expected, we find that users occupy academic buildings during daylight hours (excluding lunch) and residence halls at night. We also note that while users arrive to classes in large groups, they tend to leave in a more gradual manner.

causing users to stop binding. This trend might also be influenced by an increase in accuracy in the places that users individually frequent, requiring less training as the system learns from the user. In addition to Figure 10, we see this pattern in the similarity between Figures 13 and 14, displaying when users join and when users bind, respectively.

7 Future Work

This paper presents an implementation of a user-trained localization service covering an entire college campus. To utilize this framework in other scenarios, we consider a number of additions including new applications and novel ways to collect training fingerprints.

To augment our user base, we are considering implementing additional localization-based services. For example, we might create a tagging system for files that records location information on creation and modification. With this information, a user could search for all files created in a particular room. Users tend to create different types of files in different places, such as minutes in a meeting room and source code in a lab, so searching based on location might be helpful. Other potential services include location-based printer selection and an API to support new developers and new deployments.

Porting our code to hand-held and smart-phone devices would provide interesting new data sources and challenges. Compared to laptops, these platforms are more ubiquitous and would provide more continuous and varied data with their constantly active radios. This more diverse set of hardware might require a platform identification mechanism and conversion functions between device fingerprints, like Haeberlen's implementation across different radio platforms [7].

To gather more binds, we are considering automatic calendar integration. Many calendar appointments are tagged with a location which we could extract and utilize to automatically train the system. When a user's calendar indicates that he or she is in a specific place, the system could automatically collect fingerprints and bind them to that

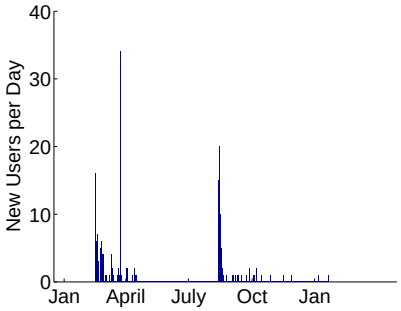


Fig. 13. Number of new users per day. Our campus-wide release occurred in April and a new academic year started in September.

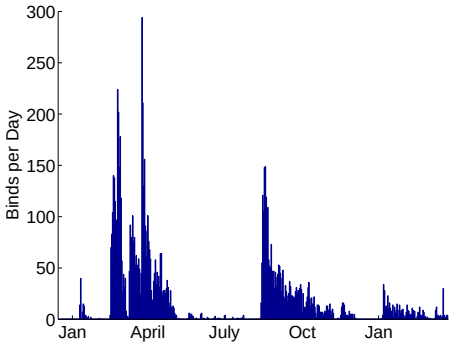


Fig. 14. Binds per day. Given that new users bind data often, we are not surprised to see a significant correlation to Figure 13. One interesting case is the system's second January, where we see no new users but a significant number of binds. In this period, students return to campus after intersession and appear again interested in binding data.

location. Obviously, users and/or their wireless devices are not always located where their calendar indicates, but with intelligent use of idle-times, a limited fingerprint set, and perhaps a movement detector like [3], this type of training could be made reliable. In more extreme cases, we might consider using calendar integration to train an entire system without any user interaction, although the accuracy of these fingerprints would require careful study.

Finally, we are continuing analysis of our data and are planning more formal user surveys to better characterize the system's strengths and weaknesses. Moreover, we are planning more expansive accuracy experiments that will inform the distinction between random-location accuracy and the common-area accuracy that a normal user experiences.

8 Conclusion

We have described a long-running test of a user-trained system that performs accurate indoor wireless localization in areas with existing 802.11 networks. The system can be deployed in any location that has a pervasive network and a group of users willing to train it. By utilizing personal laptops and existing access points, we do not need to build or buy any additional hardware. The system's interfaces are simple and intuitive, allowing users to localize, find others, and contribute training data with no instruction.

After more than one year, our system continues to operate and has accumulated more than 200 users, 8,700 binds, and over 1,000,000 location updates. Usage patterns

provide natural guidance for the localizer, improving accuracy by accumulating dense data in common areas. These methods result in successful localization to within ten meters in over 94% of cases, providing convincing evidence that crowdsourcing is a practical method for cheap, pervasive wireless localization.

9 Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and guidance. We also thank the community at Olin College for their ideas, testing, and feedback. Andrew Barry and Benjamin Fisher are supported by F. W. Olin Scholarships.

References

1. P. Bahl and V. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proc. IEEE Infocom 2000*, pages 775–784. Proc. IEEE Infocom 2000, IEEE CS Press, 2000.
2. P. Bolliger. Redpin - adaptive, zero-configuration indoor localization through user collaboration. In *MELT '08: Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*, pages 55–60, New York, NY, USA, 2008. ACM.
3. P. Bolliger, K. Partridge, M. Chu, and M. Langheinrich. Improving location fingerprinting through motion detection and asynchronous interval labeling. In T. Choudhury, A. J. Quigley, T. Strang, and K. Suginuma, editors, *LoCA*, volume 5561 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2009.
4. Ekahau, Inc. `http://www.ekahau.com`.
5. Garmin Ltd. `http://www.garmin.com`.
6. D. Geer. The E911 dilemma. *Wireless Business and Technology*, Nov. 2001.
7. Haeberlen, Flannery, et al. Practical robust localization over large-scale 802.11 wireless networks. In *Proceedings of the Tenth ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Sept. 2002.
8. J. Hightower and G. Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, Aug. 2001.
9. T. King, S. Kopf, T. Haenselmann, C. Lubberger, and W. Effelsberg. Compass: A probabilistic indoor positioning system based on 802.11 and digital compasses. In *Proceedings of the First ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and CHaracterization (WiNTECH)*, Los Angeles, CA, USA, September 2006.
10. A. LaMarca et al. Self-mapping in 802.11 location systems. In *UbiComp 2005: Ubiquitous Computing*, pages 87–104. LNCS 3660, Springer, 2005.
11. Letchner, Fox, and LaMarca. Large-scale localization from wireless signal strength. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2005.
12. Navizon: Peer-to-peer wireless positioning. `http://www.navizon.com`.
13. N. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *6th Ann. Intl Conf. Mobile Computing and Networking (Mobi-com 00)*, pages 32–43. ACM Press, 2000.
14. Skyhook Wireless. `http://www.skyhookwireless.com`.
15. S. Teller et al. Organic indoor location discovery. Technical Report MIT-CSAIL-TR-2008-075, MIT, Dec. 2008.

16. TomTom NV. http://www.tomtom.com.
17. Ubisense, Ltd. http://www.ubisense.net.
18. R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 40(1):91–102, Jan. 1992.
19. S. Whittaker, L. G. Terveen, W. C. Hill, and L. Cherny. The dynamics of mass interaction. In *In Conference on Computer-Supported Cooperative Work (CSCW)*, Nov. 1998.
20. Wisetrack, brand of TVL, Inc. http://www.wisetrack.com.