Infrastructure-free Collaborative Indoor Positioning for Time-critical Team Operations

Gokulkrishna Pillai, Pragadheeshwaran Thirumurthi, Priyanka Girase, Shweta Phirke
Computer Science, UCLA
{gkrishna, pragadh, priyanka, sap19}@cs.ucla.edu

Tutor: Youngtae Noh (ytnoh@cs.ucla.edu) Professor: Mario Gerla (gerla@cs.ucla.edu)

ABSTRACT

Indoor localization has attracted much attention recently due to its potential for realizing location-aware application services. This paper considers a time-critical scenario with a team of soldiers or first responders conducting emergency mission operations in an area where infrastructure-based localization is not possible (e.g., due to management/installation costs, power outage, terrorist attacks). To this end, we design and implement a collaborative indoor positioning scheme (CIPS) that requires no preexisting indoor infrastructure. We assume that for this area a received signal strength map was constructed a priori from blueprints, say, using a ray-tracing method. Periodic Wi-Fi beaconing (with fixed tx power) allows each node to measure signal strengths from neighbors and to identify a set of self coordinates that are consistent with the signal strength map. Then, CIPS takes over and removes invalid candidate coordinates leaving only the correct one, by performing dead reckoning over a floor map and by sharing discovered location coordinates amongst the team. Our experimental results with an Android-based testbed show that CIPS converges to an accurate set of coordinates much faster then existing non-collaborative schemes.

Keywords

Indoor Localization, Collaborative Positioning, Inertial Navigation

1. INTRODUCTION

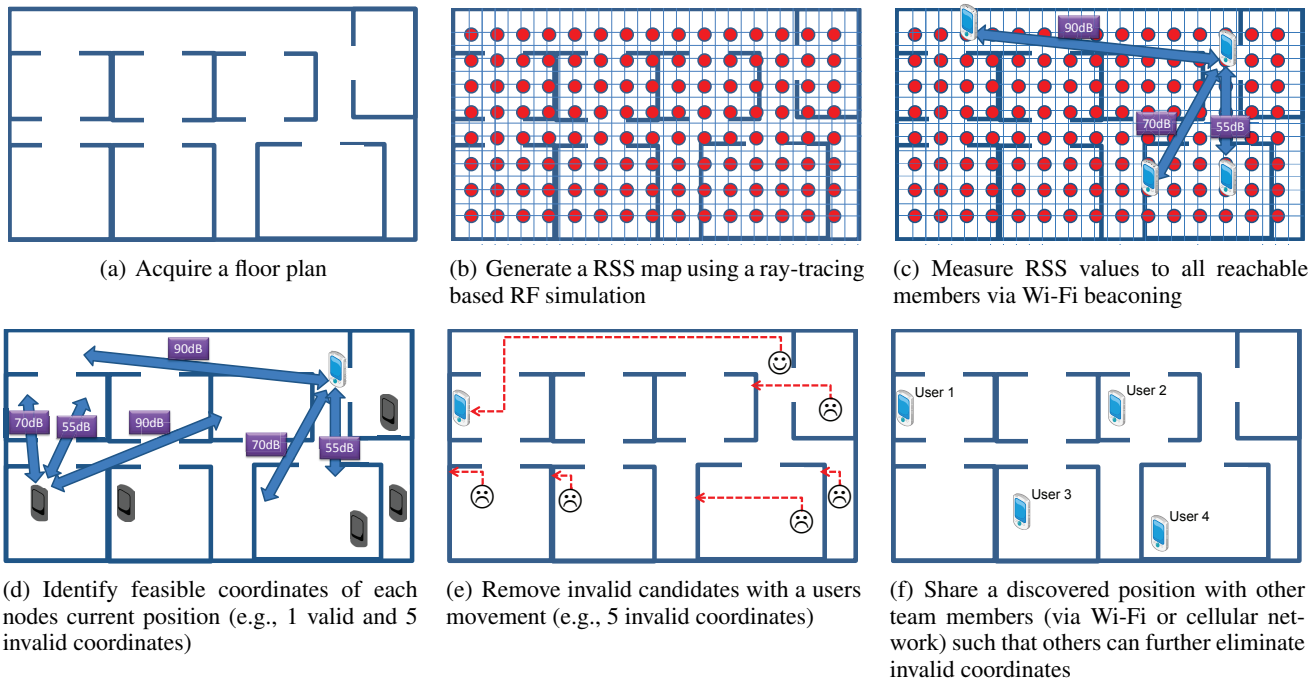
Location-based services received a lot of attention in recent years as they can deliver customized services based on people's locations. Outdoor services can be efficiently delivered with standard localization techniques supported in the off-the-shelf mobile devices (e.g., GPS, cell-tower localization). In contrast, indoor location services generally require some form of infrastructure, e.g., Wi-Fi access points [31, 12, 6, 1, 3, 32, 10], acoustic beacons [11, 33, 36], and RFID tags [19, 39, 10, 28] with quite a bit of customization.

Our work is motivated by a time-critical indoor scenario with a team of soldiers or first responders conducting emergency mission operations (e.g., firefighting, rescue, or urban military operations). On the one hand, fast and accurate localization will help the team members to navigate the area of interest and will allow them to share situation-awareness for successful mission operations. On the other hand, infrastructure-based localization is often not available in emergency scenarios for several reasons. The team may not have enough time to (install and) configure localization infrastructure such as Wi-Fi and acoustic beacons—maintaining such infrastructure for the mission operations in every building a priori is neither practical nor economically feasible. Furthermore, the entire infrastructure may not be available due to a power outage or terrorist attacks.

In this scenario, it is imperative to use infrastructure-free localization. The most common solution is inertial navigation (also known as dead reckoning) that tracks the current position of a user by constantly monitoring heading changes and traveled distance with inertial sensors (e.g., gyroscope and accelerometer) [38]. However, the major drawback of this approach is that it takes a fair amount of time to get an initial position fix even with a floor map [23, 17, 13].

To mitigate this problem, we propose a simple collaborative indoor positioning scheme (CIPS) that leverages periodic Wi-Fi beaconing among team members and accurate dead reckoning. In our scheme, mobile users basically measure the received signal strength (RSS) values from other peers. Given that for safety operations a response team can access a floor plan, we propose to use a realistic wireless signal propagation simulator to build a RSS map a priori. This RSS map allows us to search for a set of possible coordinates in which the calculated RSS values match the observed RSS values. While this process would initially produce a large number of false candidates of one's position, they can be eliminated with dead reckoning; i.e., as a user moves along the corridors, invalid candidates will quickly lead to the dead-end.

For fast convergence, a user who has just acquired the position fix broadcasts his location such that the rest of users can further eliminate invalid coordinates (typically leading to a significant removal). Every peer will repeat this procedure, and thus, each team member can quickly locate one's position; in general, the larger the number of members, the faster the convergence time. Whenever all members have



(a) Acquire a floor plan

(b) Generate a RSS map using a ray-tracing based RF simulation

(c) Measure RSS values to all reachable members via Wi-Fi beaconing

(d) Identify feasible coordinates of each nodes current position (e.g., 1 valid and 5 invalid coordinates)

(e) Remove invalid candidates with a users movement (e.g., 5 invalid coordinates)

(f) Share a discovered position with other team members (via Wi-Fi or cellular network) such that others can further eliminate invalid coordinates

Figure 1: CIPS overview

acquired their position fixes, Wi-Fi beaconing can be suspended to conserve battery; dead reckoning will continue to track each user's current position. Note that each team member will repeat the overall process periodically to maintain an accurate position fix. For the subsequent processes, we can drastically reduce the search space (and convergence time) as it is sufficient to consider the close-by locations to a node's current position (e.g., with fixed radius).

The main contribution of this paper is to show the feasibility of infrastructure-free indoor localization with fast position fix by using collaborative Wi-Fi beaconing and accurate dead reckoning. Lack of infrastructure installation/maintenance is realistic in such an emergency scenario, thereby CIPS a practical solution. Our field experiment results show that CIPS provides the position fixes much faster than non-collaborative schemes and with sufficient accuracy for representative scenarios.

The rest of paper is organized as follows. In Section 2, we provide a brief overview of CIPS. In Section 3, we provide a detailed description of CIPS and its challenges. In Section 3.3, we provides description of an approach to improve indoor dead reckoning and its validation. In Section 4, we validate the performance of CIPS through field tests. In Section 5, we review the related work in the field. In Section 6, we discuss the limitations of current system and indicate possible improvement directions. In Section 7, we conclude the paper and discuss future work.

2. CIPS OVERVIEW

Our objective is to help a group of rescuers localize themselves on an unfamiliar floor-plan without any infrastructure support. For emergency operations, we assume that the team has a floor plan and building information, and team members use homogeneous mobile devices (smartphones). The entire procedure is described as follows. As depicted in Figure 1(a), we first obtain floor plans of a target building. The floor-plan is divided into $N \times N$ grids as shown in Figure 1(b). Here, the area size of a grid is determined by the required accuracy level of an application; in our scenario, we use a grid size of $2m \times 2m$. We perform a ray-tracing based RF propagation simulation to generate $N \times N - 1$ received signal strength (RSS) values (for each coordinates on the grid, there are $N - 1$ RSS values). We basically assume that before the mission starts, each team member downloads this signal strength map, along with the detailed floor-plan information.

Figure 1(c) depicts what actually happens once the team members enter the target building/floor. They start Wi-Fi beaconing to measure the RSS values of the reachable team members. These measured RSS values are then matched against the RSS map generated with a RF simulation; the detailed matching algorithm is elaborated in Section 3.2. The matching outcome is a list of feasible coordinates of a user's position. The next step is to remove all the invalid coordinates and to pinpoint a single coordinate that matches with the user's actual position. We use dead reckoning that tracks the heading changes and distance walked to estimate the current position of a user. Dead reckoning is applied to every candidate coordinate to find whether it leads to dead end; i.e., the candidate coordinate is invalid. The illustration is presented in Figure 1(e). Finally, once a user has localized himself correctly, this location is broadcast to the rest of the nodes such that they can further eliminate invalid coordinates. This way, every team member knows whereabouts of

all the team members as shown in Figure 1(f).

3. SYSTEM DESIGN AND CHALLENGES

We have implemented our project as an android application. Before going into the details, we will explain the working systems stages: (1) floor plan pre-processing and RSS map generation, (2) a slack parameter α to adapt RSS fluctuations due to small scale fading

3.1 Map Generation and Pathloss Simulation

Our positioning scheme needs to prepare a floor map for each site. Such a floor map can be obtained as Geographic information system (GIS) data in known places, or can be given as a floorplan. We may assume such a service provider that recognizes a given floorplan image and builds a digital floor map, because architectural floorplan image recognition has been extensively studied [15] and recently similar services have been launched such as "Google Maps Floor Plans" for WLAN indoor positioning accepting users' upload of floorplan images. Building administrators can register a digital image to such a service provider in case for emergency situation.

For pathloss simulation purpose, an RF propagation model (e.g., the log-distance path loss (LDPL) model) can be used to predict RSS at various locations. The advantage of using these models is that it reduces the number of RSS measurements significantly compared to RF fingerprinting schemes, albeit at the expense of degraded localization accuracy. Since RF propagation characteristics vary widely (especially indoor), the model parameters would have to be estimated specifically for each indoor space in question. For example, [8] presents measured data for 2.5GHz in-building partition loss.

Ray-tracing has also been widely used, and some tracing models consider all the details in the place, such as walls, windows, doors as well as desks, chairs and the thickness of walls to pursue the best accuracy. Ray-tracing technique approximates the radio propagation with a finite number of isotropic rays emitted from a transmitting antenna by ray imaging technique. In this technique, the transmitter is assumed to be reflected at each surface around it to produce image transmitters, the reflected rays to the receiver from the real transmitter are considered as direct paths from the mirror images of the true transmitter. Based on geometrical optics, each ray from the transmitter to the receiver can be exactly determined. The major drawback of such techniques is, however, expensive computation complexity. Readers may refer to detailed ray-tracing technique in [22, 34, 16, 26, 30, 21]

In our simulation, any models can be used. However, in order to pursue reasonable balance between the effort of floor map modeling, simulation time and accuracy, we take the following modeling and simulation: we consider all the solid lines in a floorplan as walls where windows are regarded as walls and doors are not drawn (we assume they are always open since mobile nodes walk through), and all the spaces surrounded by the solid lines as accessible spaces. With this indoor modeling, we use a simplified (reduced) 3-ray tracing in the experiments (provided by Wireless InSite software[4]) where diffraction along obstacles are considered. In practice, however, the measured signal strengths tend to fluctuate due to small scale fading. While this phenomenon can be mitigated by conducting multiple measurements to find the average RSS values, a node may still suffer from temporal fluctuation (as a result, we may remove the true position). To solve this problem, we introduce a slack parameter α (usually $\pm$13dB in our setting). We will further justify this approach with measurement results in Section 4.

3.2 Matching measured beacon log with a RSS map

From each beacon log $(i, j, ss)^t$ of node j where ss is a received signal strength from node i at time t, node j estimates the pathloss value in this communication (denoted as $m(i, j)$). For this purpose, we assume that transmission power and other factors such as sender antenna gain/loss and environmental noise are almost common and constant on all mobile nodes. We note that these values may be hardware-dependent, but pre-measurement before localization (before mission starts) is effective to know such values.

3.2.1 Positioning problem formulation

In this section, we focus on the positioning activity of node j.

Given a floor plan of *fp* with *N* grids, let $L : N^2 \rightarrow \mathcal{R}^+$ denote a pathloss matrix among N points where element $(u, v) \in N^2$ is the simulated pathloss from points u to v. Also, let M_j denote the set of node j and the nodes that node j has a beacon log at time t. Let $m : M_j \times \{j\} \rightarrow \mathcal{R}^+$ denote the set of pathloss measurements of node j at time t (*i.e.* $m(i, j)$ is the measured pathloss value from nodes i to j).

The positioning problem of node j is to find node j's position from the N points with the least distance from the true position. Our approach is to find such a position by finding the positioning function $p : M_j \rightarrow N$ with the least "pathloss matching error" between m and L. The objective function is defined to minimize such pathloss matching error as follows;

$$\min \sum_{i \in M_j} |m(i, j) - L(p(i), p(j))| \quad (1)$$

We note that if pathloss increases as the square of distance (like free-space attenuation), we should find such p that has the least square deviations. Nevertheless, we adopt this simple sum of deviations since such pathloss characteristics are highly situation-dependent and it is therefore hard to determine such an attenuation coefficient.

As we addressed earlier, we need to accommodate measurement errors caused by fluctuation of RSS values. We should allow some deviation parameter α where a measured pathloss l' and a simulated pathloss l are regarded identi-

cal *iff* $l' \in [l-\alpha, l+\alpha]$, and choice of appropriate α will further be investigated in Section 4. Then using function z where $z(l', l) = 1$ *iff* $l' \in [l-\alpha, l+\alpha]$ and 0 otherwise, we may use the following objective function instead of (1).

$$\max \sum_{i \in M_j} z(m(i,j), L(p(i), p(j))) \quad (2)$$

This let us find p that maximizes the number of pathloss-matched edges in m and L.

3.2.2 Algorithm and Complexity

Assume graph $G = (M_j, m)$ and complete graph $H = (N, N^2)$. The above optimization problem is the *maximum common subgraph isomorphism problem* (MCSI problem) in graph theory, which finds an induced graph G' of G in H with objective function (2). Although the general MCSI problem is known to be NP-hard, our problem is a special class of MCSI where given graph G has a star topology centered at node j. Therefore, if j is allocated to point $v \in N$, calculation of objective function (2) in this case can be induced to the following problem; (i) for each edge $(i,j) \in m$ find all possible edges $(u,v) \in N^2$ that satisfies $z(m(i,j), L(u,v)) = 1$, and (ii) find the positioning function $p : M_j \to N$ from (i) with objective function (2). Part (i) needs exhaustive tests of edge pairs in G and H respectively, and part (ii) can be induced to the *maximum bipartite matching problem* in the graph where M_j and N are bipartite vertices and possible allocations of nodes in M_j to points in N found in part (i) as well as (j,v) are edges. The computation complexity of part (i) is $O(|M_j||N|)$ and that of part (ii) is also $O(|M_j||N|)$ by the path matching algorithm [14]. Thus our optimization algorithm totally needs $O(|M_j||N|^2)$ to apply the above parts (i) and (ii) to all points in N. The pseudo code is given below.

Algorithm 1 Positioning algorithm for node j

```
1: procedure find_possible_positioning_functions (M_j, m, N, L)
2: k ← 0; F = ∅
3: for each v ∈ N do
4:     E ← (j, v)
5:     for each pair of edges (i, j) ∈ m and (u, v) ∈ N^2 do
6:         E ← E ∪ (i, u) if z(m(i, j), L(u, v)) = 1
7:     end for
8:     E' ← bipartite graph matching result for ((M_j, N), E)
9:     if |E'| > k then
10:        F ← E'; k ← |E'|
11:    else if |E'| == k then
12:        F ← F ∪ E'
13:    end if
14: end for
15: return F as a set of positioning functions
16: end procedure
```

3.3 Location convergence

Our algorithm converges to the user's location by examining the individual's indoor path trace and eliminating untraversable and thus invalid points (e.g., crossing walls, moving outside the building). Our indoor path tracing mechanism over a floor plan (i.e., dead reckoning) uses the mobile phone's sensors to collect movement direction and distance.

For the direction, we use Android's heading and magnetic field sensor to identify direction of the movement. During our initial implementation, we observed that the compass's readings are fluctuating due to user sway, movement irregularities, magnetic fields in the surroundings, and the sensor's internal bias. Significant heading changes (observed when a user takes a turn at corner) can be detected with a recently proposed method by Constandache *et. al.* [13]; i.e., the compass headings change is more significant than random oscillations. Thus, we use the following condition for turn detection:

$$Avg(t_{(i+1)}) - Avg(t_i) \geqq \frac{StdDev(t_i) + StdDev(t_{i+1})}{2} \quad (3)$$

where $Avg(t_i)$ denotes the average compass readings over a t_i time period, $StdDev(t_i)$ denotes standard deviation of compass reading during t_i, and G denotes a guard factor.

For the distance calculation, it has been reported that double-integrating the acceleration readings results in highly inaccurate results (i.e., introduces more than 500m error with 55m traveled distance [13]). Thus, we compute a user's displacement by multiplying step count (i.e., number of steps taken) with the user's stride length (i.e., distance covered in one step).

Extracting step count when holding a phone in the palm: In general, a regular pedometer logs the number of steps taken in a given duration of time for a user. However, today's popular smartphones do not contain a reliable pedometer. Thus, we devised an algorithm to collect sensor data from the smartphone's accelerometer which measures the change in gravity in the x, y, and z direction. The challenge is then to reliably infer the user's step count from change in acceleration while filtering noisy data. The difference in observations is shown in Figure 2 and 3. Figure 2 shows more distinct readings to extract step count (13 steps). When the phone is in pocket, the forces that the accelerometer experiences are in line with the user's movement. In our target scenario of an emergency situation, an individual will hold the smartphone in the palm while walking through the building for communication purposes. We take advantage of this by running our localization algorithm and updating our convergence algorithm.

To devise an algorithm to infer a user's step count from the accelerometer sensor data, we performed experiments with different walking speeds for both male and female users. We discovered that the number of steps taken were equal to the peaks in the graph obtained with accelerometer readings (for example, 13 steps are 13 peaks). However, it is not so simple to identify the peaks for sensor data as there is noisy data. Our algorithm for peak detection uses a band pass filter which identifies small intermittent peaks and measures only the large peaks corresponding to steps.

Figure 4, shows the extracted y-axis acceleration readings

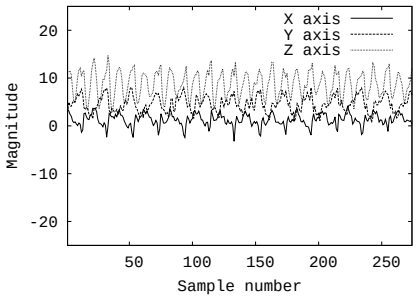


Figure 2: x, y, and z accelerations in the pocket

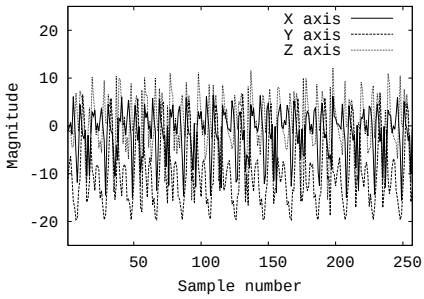


Figure 3: x, y, and z accelerations on hand

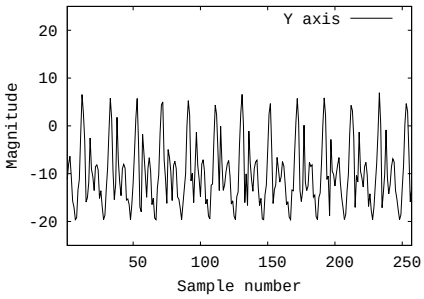


Figure 4: y acceleration on hand

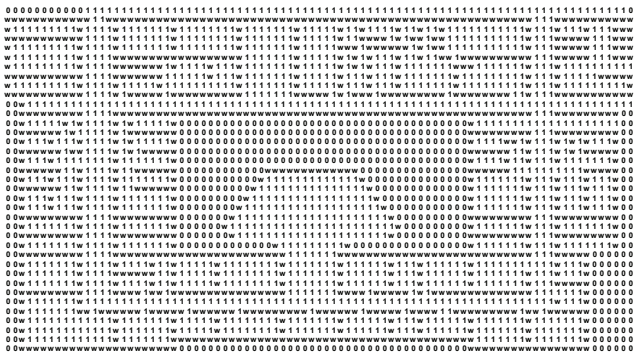


Figure 5: CS department building maze: 1 **(accessible),** 0 **(inaccessible), and** W **(wall)**

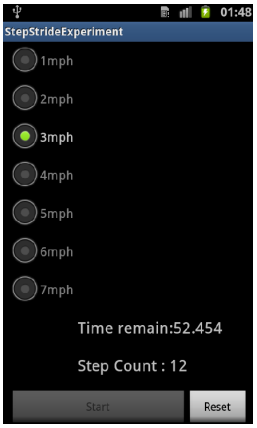


Figure 6: Step Profiling App: generating user's a stride length for $3\ mph$

from Figure 3, and how the number of steps are equal to the number of peaks observed (excluding minor fluctuations). Since peaks represent the highest deviation from the mean, our algorithm detects when the observed accelerometer reading crosses a threshold value, and only then increments the counter. For a window of observed accelerometer readings, we use two thresholds—an upper bound and a lower bound. The upper bound is the sum of average and standard deviation of this window (peak threshold), and the lower bound is the difference of two (valley threshold). Because a stationary user still experiences the force of gravity, the accelerometer sensor reports these slight fluctuations as well. However, since the user is actually stationary, the standard deviation of theses observed values is very low. Observing this fact, we incorporate this event into our algorithm. Thus, continuously monitoring the sensor data, we process it for detecting a step only when the standard deviation of the observed values is above a certain threshold, representing user movement. Then, when the upper bound (peak) is crossed followed by lower bound (valley), we increment the step count. The accuracy of step detection is directly proportional to how accurately these thresholds are set. Section 3.3 further discusses how we improved the accuracy of user displacement in a given time by using step profiling. This step detection module is run every second and monitors the upper and lower peaks with respect to standard deviation of the sensor observed values, in order to detect user's step.

For our experiment, we converted an example floor plan into a $2m \times 2m$ matrix as shown in Figure 5. This maze comprises of $\{1, 0, W\}$, which represent accessible points, inaccessible points, and walls respectively. After the first Wi-Fi scan, all feasible coordinates on this grid (i.e. all possible 1s) are enlisted. Every time a distance equal to that between two coordinates on the grid ($2m$) is traveled by the user, the path tracking module records user's direction of movement. Then it scans through the current list of all feasible candidates, and updates each point based on user's movement in the observed direction. The system then eliminates all those points which either fall on a "0" or "W" on the grid after the update. This process of elimination continues iteratively as the user moves around by changing directions until the system can narrow down to a single grid point. Once the user converges to a single unique point, he can broadcast it to other team members such that they can further remove invalid coordinates and facilitate their team mission. *Accurate stride length estimation*: To enhance the dead reckoning precision, we profile a user's stride length. In general, stride length can be a function of multiple factors. A user's step size can be thought of as a function of his gender, weight, and height [7]. The stride length of a male user is calculated by $0.415 * height$ while a female's stride length is calculated by $0.413 * height$. Experimentally we observed that walking speed also plays a crucial role in the calculation of the stride length. Therefore, we incorporated the moving speed

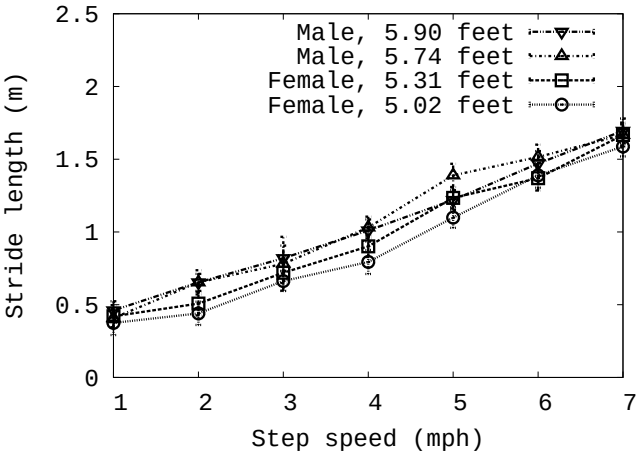


Figure 7: Stride length as a function of speed of walking (different gender and height)

and the corresponding stride length into our system.

Our profiling approach consisted of two phases. In the first phase, a user trains the system for his specific stride lengths for different walking speeds. This builds a reference data for the system to refer to. The GUI for this is shown in Figure 6. It shows different speed options of 1 mph, 2 mph and so on till 7 mph and also the corresponding lap time. For our tests, we created four separate profiles for users of different heights and gender. The test participants walked the test area of length $100m$ to train the system for different speeds and corresponding stride lengths.[1] Figure 7 shows the results of our experiments. It can be observed that independent of the height and gender, user's stride length increases as the walking speed increases. It is also closely related to the user's walking styles. For example, a female of height 5.83 feet walking at a speed of 5 mph has similar stride length as that of a male of height 5.90 feet. This example justifies why it is necessary to use step profiling for each user to obtain more accurate traveled distance. It is noteworthy that height is not directly related to stride length, as claimed in [7].

In the second phase of offline training, the system uses the reference data to dynamically determine a user's stride length based on his current walking speed. Table 1 and Table 2 compares the error percentage (distance deviation from the ground truth) in distance traveled by a male user and a female user respectively, for average (statistical) stride length and profiling stride length. According to average stride length [7], stride for a male of height 5.83 feet is a single fixed value $(0.737m)$ for all walking speeds. Similarly for a female of height 5.5 feet, it is 0.637. Since stride length naturally increases with walking speed, stride analysis tends to become more inaccurate at higher speeds increasing the error percentage. However, profiling the stride length does well here.

[1]One can automate this calibration process with GPS: a user simply walks a straight line of say 100 meter. By walking a long distance, one can effectively eliminate the impact of GPS errors.

Table 1: A male's (5.83 feet) average stride length and step-profiling results at three different step speed levels

Speed		Mod	Fast	Run
Distance (m)		100	100	100
Lap Time (sec)		74.5	44.7	31.9
Observed # of step		127	77	68
# of Step (Ground Truth)		127	84	71
Male Stat	Step len (m)	0.737	0.737	0.737
	Dist Err	6.4	42.04	49.884
Male Profile	Step len (m)	0.79	1.28	1.42
	Dist Err	0.33	1.44	3.44

Table 2: A female's (5.5 feet) average stride length and step-profiling results at three different step speed levels

Speed		Mod	Fast	Run
Distance (m)		100	100	100
Lap Time (sec)		74.5	44.7	31.9
Observed # of step		121	77	75
# of Step (Ground Truth)		125	86	77
Female Stat	Step len (m)	0.673	0.673	0.673
	Dist Err	18.93	48.179	49.525
Female Profile	Step len (m)	0.82	1.32	1.42
	Dist Err	0.78	1.64	5.08

In the first phase, the user has created his profile for different stride lengths corresponding to different speeds. The system then dynamically selects the correct stride length looking at the current walking speed of the user. For instance, for a male of height 5.83 feet, the stride at moderate speed was observed to be 0.79, 1.39m at fast speed and 1.67m at running speed. When this data was used, instead of average stride length, the difference in distance actually covered and distance calculated by the method of $stridelength \times stepcount$, dropped drastically. This improved the system performance tremendously. Finally, a dynamic system is more suitable for our target scenario where the firefighter could be walking and running intermittently.

4. EVALUATION

4.1 Field Test: CS department building 3rd floor

To test overall system performance in diverse scenarios, we performed field tests in the CS department building, 3rd floor. To evaluate our proposed positioning system, we generated a preprocessed floor plan with the grid overlay as shown in Figure 8. To measure the system performance dependence on the number of indoor team members, we performed field tests with varying number of team member, ranging from 2 to 9. In addition, to improve indoor navigation accuracy, we used stride length profiling. Also, in order to verify gender effects on the proposed system, we

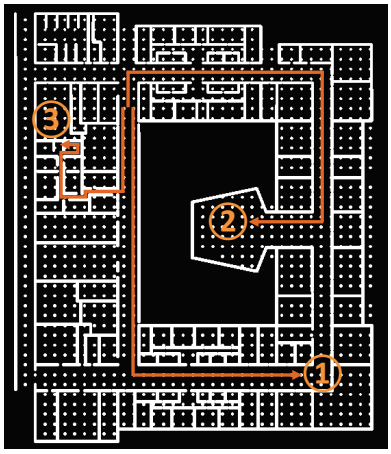


Figure 8: Preprocessed floorplan and Route $\#1, \#2,$ and $\#3$

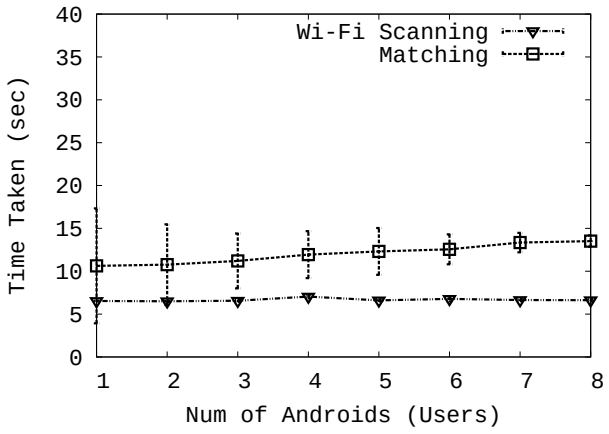


Figure 9: Time taken for scanning and matching as a function of the number of team members

performed field tests for both male and female users, trained the application for their respective profiles, and attempted to localize them separately. Finally, for path tracking, we used the same floor plan as explained previously and shown in Figure 5. To verify how the system performance is impacted by the nature of a route traveled by a user (i.e., the number of corner turns and the length of a straight line taken by a user), we carefully chose three representative routes. These are $Route\ \#1$, $Route\ \#2$ and $Route\ \#3$ shown in Figure 8. These routes start from the same staring point but while $Route\ \#1$ is made of mainly a long straight line path with just one turn, $Route\ \#2$ contains more turns with long enough straight line paths between two consecutive turns. $Route\ \#3$ contains many turns with very short straight line paths after each turn. The results for all these field experiments are jotted down in the subsequent section.

4.2 Experimental Results

We tested and evaluated each component separately to check its impact on the overall system performance. Also, we tested in different field scenarios, for example different routes and using different stride length approaches. Finally, we provide the overall system performance in a realistic scenario.

Time taken for RSS exchange and matching: To measure the delay of Wi-Fi beaconing and matching, we used off-the-shelf Galaxy S2 smartphones that contains Wi-Fi and a set of sensors necessary for dead reckoning. All these devices run on the Android platform of version 2.3.4. When the system starts, each user exchanges Wi-Fi beacons with other members, logs the received beacons himself, and matches the log with the preprocessed floor plan. Figure 9 shows the time taken to detect all peers and matching delay. We observed the scanning matching to be proportional to the number of devices while Wi-Fi scan remains a constant of around 6.5 sec.

The number of peers and α values: Figure 10 shows feasible coordinate ratio with different number of peer devices. We also varied the slack value of α as a configurable parameter to find its optimum value, small enough to eliminate more infeasible points but big enough to not to miss the true location. The feasible coordinates ratios is calculated by dividing the number of outcomes by the all coordinates in the floor plan (48×48 coordinates). While the α values showed a positive relationship with feasible coordinates ratio, we observed that the number of false positives decrease as we increase the number of peers. Figure 11 illustrates the probability of the matching outcomes to contain the true positive position. Like Figure 10 shows, we varied the number of team members and also the size of α value to observe the changing system behavior in different cases. We observed that as we increase the size of α value, the probability of current point being scanned, becomes 100%. Another interesting observation is that at smaller team (with few members) requires smaller size of α value to be able to have the current position in the initial scan. For example, a 1 member team with 9 α value was observed to scan the current position with 100% chance, while an 8 members team requires 15 α value to have the same probability.

Convergence speed based on different routes: Among the overall system performance metrics, convergence speed to the unique point is one of key factors for the proposed scheme in use. We investigated how this factor is affected by different routes and by utilization of peer to peer exchange of RSS and matching with simulation data. To verify this, we carefully chose three representative routes, namely $Route\ \#1$ (including long straight line and a turn), $Route\ \#2$ (including moderate straight line path and few turns), and $Route\ \#3$ (including short straight line and many turns) to see which of the two factors, having long straight line, or the number of turns, affects the convergence speed more. We measured 20 times on each route and evaluated the success ratio of two different stride approaches. Figure 12 show the convergence to the unique point as a function of traveled distance. $Route\ \#1$ shows rapid drop of infeasible coordinates ratio in the beginning due to its distinct route feature but after trav-

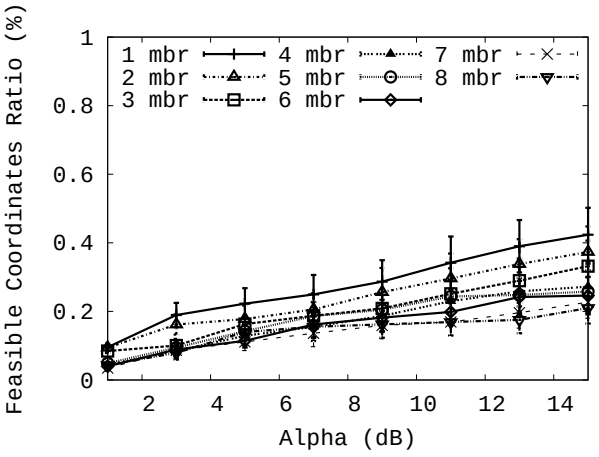


Figure 10: Matching outcome ratio

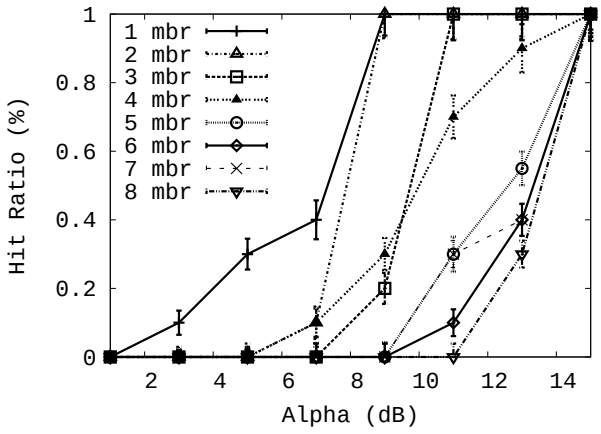


Figure 11: Probability to contain current point

eling $20m$, it showed relatively slow convergence. This was because multiple points on the straight line still remain as candidates, but after $60m$ when a turn is taken, it converges to the unique point. $Route\ \#2$ shows the slowest convergence speed. It is because it contains relatively moderate events such moderate straight lines and turns. $Route\ \#3$ shows the fastest convergence speed to the unique point. It is because it contains many turns. Thus, based on our observation, a complicated route tends to expedite the process of convergence to a unique point. We noted that even without peer to peer exchange of RSS values, and matching, the convergence on the three routes showed similar behavior. However, it is also noteworthy that all three routes, without peer to peer RSS exchange and matching, required the user to travel longer distance in order to converge to a unique point. Figure 13 shows the exact traveled distances on the 3 routes to converge, with and without RSS.

Convergence accuracy based on different stride length approaches: Positioning accuracy is another key factor of the proposed scheme. To measure this, we performed the field tests 20 times per route using two different stride length approaches, namely average (statistical) stride length and step profiling. Since users are navigating indoors, it is highly necessary to have accurate displacement logging in the maze shown in Figure 5. Unlike the average stride length approach, step profiling uses dynamic profiled stride length to calculate traveled distance by using $StepCount{*}StrideLength$. It is therefore more accurate and quite close to the actual distance traveled by the user. Eliminating false positives on the floor plan based on using this is therefore more accurate. Especially, at the end of a corridor, both late and early, than required, changes of direction will cause failure of convergence to a unique position. Figure 14 shows the convergence ratio of two different stride length approaches, namely average stride length and step profiling approaches, which is well matched with distance deviations for both male and female shown in Table 1 and 2 respectively. Since step profiling has accurate distance measure so it provides 100% hit ration in route 1 but it is slightly degraded in route 3. This can be attributed to the error in compass readings, caused by the magnetic fields in surrounding. Even when profiling calculated the distance travelled correctly, error in orientation caused the inaccurate results. To eliminate these errors, gyroscope and accelerometer can be used in addition to the orientation sensor as explained in [38]. We plan to incorporate this feature in the next version. The average stride length approach, on the other hand, was always observed to calculate shorter than actual distance traveled in a long straight corridor, and therefore performed poorly (32%.) For a path like that of Route 3, however, the average stride length showed relatively reliable performance. This could be because the overall distance error is refreshed after each turn on this route.

Overall convergence delays in different routes: Figure 15 shows the time taken to converge to a unique point with step profiling in the three routes. The initial Wi-Fi scanning and matching takes a constant time for all the three scenarios. The difference can be observed when user move according to the routes. The root cause of the time consumption is traveling required to converge to a single unique point. As in Figure 13, route 2 requires longer distance trip to converge to the unique point, which consequently takes the longest times compared to the other two routes.

5. RELATED WORK

The existing indoor localization can be classified into four categories, namely specialized infrastructure based, place learning based, sensor based, and model based. For surveys, readers may refer to [18, 25].

Specialized Anchor based: Indoor positioning has been intensively investigated recently to provide accurate position information in the indoor environments, where GPS is not accessible. To this end, the early indoor positioning schemes achieved their required accuracies with specialized infrastructure such as RFID tags, Ultra-sound transmitters (or receivers), and infrared(IR) beacons and receivers. Want *et. al.* propose Active Badge [36]. In this indoor localization system, personal wears small device, which transmits a unique

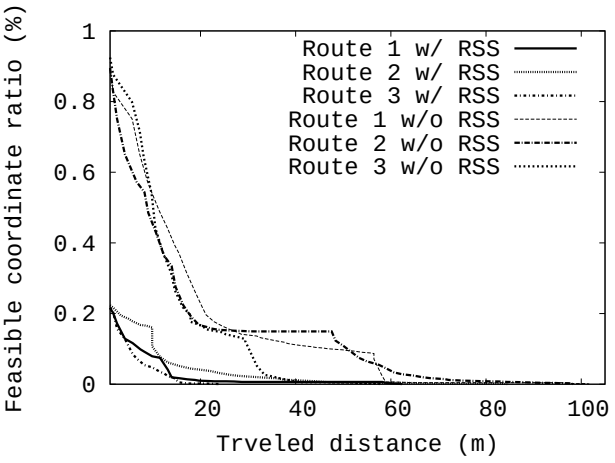


Figure 12: Convergence speed based on different routes with and without beacon RSS

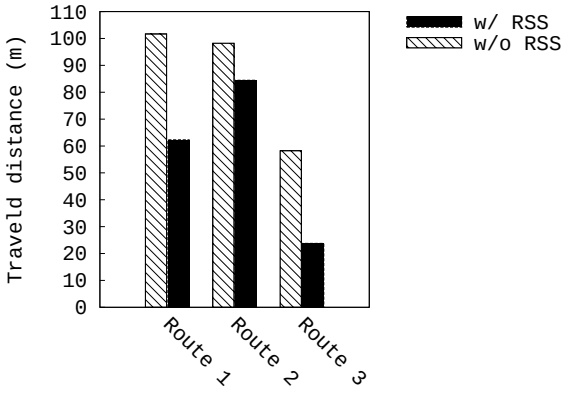


Figure 13: Traveled distance to converge unique point with and without beacon RSS on three different routes

infra-red signal every 10 seconds. Each room in a building is equipped with one or more networked sensors which detect these transmissions. The location of the badge (and hence its wearer) can thus be determined on the basis of information provided by these sensors. As similar work, Cricket [29], Bat [37], and WALRUS [11] rely on ultrasound devices being deployed at various locations within the indoor environment as well as on the mobile devices. Recently, RFID based systems such as LANDMARC [28] also have been proposed. However, the major concern about these approaches is that they need stations (specialized anchors) depending on the area size, which thereafter cause significant cost and effort to be applied.

Place learning based: Existing work on this category is based on fingerprinting/place-learning each indoor location in a building with a vector of received signal strength (RSS) measurements for various transmitters and build a RSS map based on observed RSS readings and localize a mobile device by matching its reading with the map. Bahl *et. al.* proposed RADAR [10], which used a deterministic fingerprint for each location by using the nearest neighbors in RSS map. To enhance this scheme, Youssef *et. al.* proposed Horus [40], which employs a stochastic description of the RSS map and uses a maximum likelihood based approach. Commercial localization products have also been built using these methods [2]. As another work, Varshavsky *et. al.* [35] has demonstrated that GSM-based indoor localization is possible, which uses wide signal-strength fingerprints from various cell towers. Recently, commercialized solutions such as SkyHook [6], Google Latitude [3], and PlaceEngine [5] combine GPS, cell-tower-based estimation, and Wi-Fi fingerprints.

As another approach for the place learning, Azizyan *et. al.* proposed SurroundSense [9], which builds a map using several features found in typical indoor spaces such as ambient sound, light, color, etc., in addition to Wi-Fi RSS. Tarzia *et. al.* [33] proposed BatPhone, which determines a mobile phone's indoor location, which is based on a new ambient sound fingerprint called the Acoustic Background Spectrum (ABS). After learning indoor places and building a database for each user, location is then determined by measuring the current fingerprint and then choosing the "closest" fingerprint from the database. To eliminate pre-deployment effort and prior knowledge about Wi-Fi APs in a building, Chintalapudi *et. al.* [12] proposed EZ, which is configuration-free indoor localization scheme that uses existing Wi-Fi infrastructure to localize mobile devices. EZ learns locations by collecting data from mobile devices carried by users as they traverse the indoor space of interest in normal course. As an interesting application, Shin *et. al.* proposed a software architecture called FindingMiMo, which tracks and locates a missing mobile device in indoor environments [31]. The system consists of a missing mobile which logs diverse environmental features on a daily basis, and a chaser which traces the trail of the device using the observation log. During daily operation, the mobile device does not perform location estimation; it only observes the ambient features such as radio signals to minimize its operation cost.

Indoor positioning schemes categorized into this criterion entail a considerable amount manual effort to perform detailed measurements across the entire indoor space and maintain the RF map over time. Although some may reduce burden of building DBs (i.e., place learning) by replacing other environmental features such as sound, light, color fingerprints, and daily-based automated RSS logging. All these schemes are, however, still require place learning and cannot be used instantly.

Simultaneous Localization and Mapping (SLAM) based: For a robot to navigate through an indoor environment, it must have the ability to determine its current location. Initial approaches provisioned the robot with a map of the indoor environment, allowing it to determine its location by comparing its observed environment (using ultra-sound, LADAR sensors, etc.) to the map. A significant step in the area of robotics was Simultaneous Localization and Mapping (SLAM) [23], which allowed a robot to build a map of the indoor environment (in terms of walls and other obstructions) while

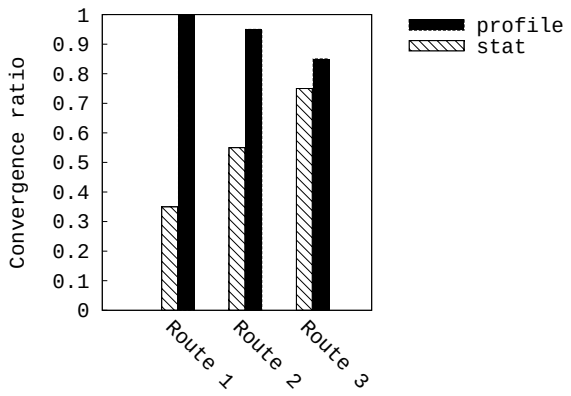


Figure 14: Convergence success ratio of two stride length approaches in three different routes

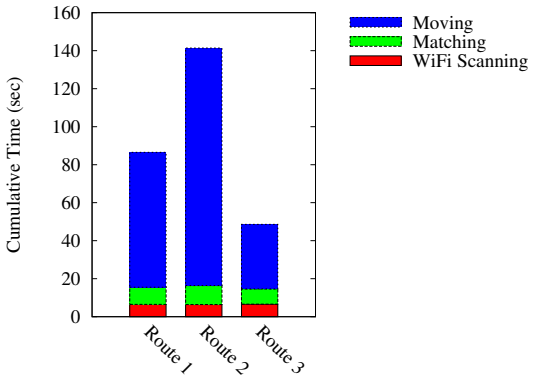


Figure 15: Cumulative Latencies of three modules for different routes

simultaneously determining its location with respect to the constructed map. WiFi-SLAM [17] extends this to building a Wi-Fi RSS map using a mobile robot. The robot uses its onboard odometer to determine the distance it has moved between measurement points. Knowing a few of these locations (using GPS or certain landmarks) allows estimating the parameters of the LDPL model. In contrast, EZ builds an RF model without the benefit of sensors such as odometers and LADARs. As a similar work, Woodman *et. al.* investigated how a foot-mounted inertial unit, a detailed building model, and a particle filter can be combined to provide absolute positioning, despite the presence of drift in the inertial unit and without knowledge of the user's initial location. This work additionally investigated how to handle multiple floors and stairways, how to handle symmetry in the environment, and how to initialize the localization algorithm using Wi-Fi signal strength to reduce initial complexity [38]. Martin *et. al.* proposed an application for indoor localization to be implemented in current state of the art smart phones, leveraging their sensing capabilities to deliver up to 1.5 meters accuracy without the requirements for complex hardware that existing solutions need, which use only the hardware embedded within the phone and integrating both online and offline phases of RSS fingerprinting within the same device [27].

Model based: An RF propagation model (e.g., the log-distance path loss (LDPL) model) can be used to predict RSS at various locations in the indoor environment. The advantage of using these models is that it reduces the number of RSS measurements dramatically compared to RF fingerprinting schemes, albeit at the cost of decreased localization accuracy. Since RF propagation characteristics vary widely, the model parameters would have to be estimated specifically for each indoor space in question. Gwon *et. al.* proposed TIX [20], which assumes that the transmit power and locations of all Wi-Fi APs is known. The APs are modeled to measure the RSS of the beacons from neighboring APs. Linear interpolation is then used to estimate the RSS at every location in the indoor space, which is then used for localization. To allow unmodified, off-the shelf APs to be used, Lim et al. [24] employ Wi-Fi sniffers at known locations. These sniffers measure the RSS from the various APs and use the LDPL model to construct an RSS map. ARIADNE [21] also deploys sniffers at known locations but makes use of a more sophisticated ray-tracing model based on detailed indoor maps and uses simulated annealing to estimate radio propagation parameter. Given the signal measurements for a mobile, a proposed clustering algorithm searches that signal strength map to determine the current mobiles location. While these methods cut down the measurement effort, they still require effort in terms of placing infrastructure such as sniffers.

6. DISCUSSION

Broadcasting converged position: Once a user has performed the instant localization and determined their location within the floor plan, these coordinates are then shared with fellow team members. We allow for either ad-hoc communication or a centralized messaging server using XMPP to disseminate a user's converged location. To broadcast the converged position to other team members, ad-hoc messaging is utilized to disseminate location messages within the group. This becomes even more useful when user's converged position is out-dated. One can use ad-hoc broadcasting information of other team members and start with bounded feasible coordinates to regain the unique point. To achieve ad-hoc communication, each tactical team is responsible for agreeing upon a SSID name before entering the building. To propagate the location messages, UDP broadcast packets are sent and processed by each node. In ad-hoc mode location messages are a best effort delivery. In the case where a packet gets dropped, certain team members may not have the most accurate information. Though as the location messages are sent as the location is updated and an individual moves, a best effort approach ensures that eventually the location information is disseminated within the group.

To support limited Wi-Fi range, we can utilize the cellular network for isolated members to exchange location information using the XMPP protocol when the connection

is available. The Jabber server is a well established protocol that enables group messaging and is designed to be scalable and extensible. Due to mobile IP and a constantly changing device's IP, we identify a user by a logical id (e.g., *user1@ips.com*). Each firefighting group may then form their own logical groups composed of each unit. Our application creates a TCP connection to the Jabber server and sends the message containing the user's location information, the logical id of the sender, and the logical id of the target group. The Jabber server then is responsible for distributing messages amongst all team members once the message is received. Each user signs into the Jabber server and the Jabber server maintains a TCP connection with each client. Thus it is able to forward messages to the logical groups. Due to the sensitive and private nature of location information, we rely on XMPP's support for TLS for transport security and the authentication mechanism to sign on. Additionally, a tactical group may install their own XMPP server to ensure proper data ownership of location information.

Path-loss simulation random coordinates: Random distribution of coordinates instead of a uniform distribution may be beneficial to remove flip and rotation in matching. The density of those points is also important parameter to understand the trade-off between accuracy and computation complexity. It may be possible to aggregate two points that have similar propagation characteristics (i.e., points in the same small room) to reduce complexity and expedite convergence to the unique point. To do this automatically, we need to check if two nearby points after map matching. Both remain our future work.

7. CONCLUSIONS

We proposed a novel infrastructure-free collaborative indoor positioning system called CIPS. Given the floor map and the blueprint of the building, a received signal strength map is built using ray-tracing. When the mission starts, each team member uses the periodic Wi-Fi beaconing and the RSS values to identify a set of feasible self coordinates. We show that CIPS can quickly remove invalid candidate coordinates and accurately converge to a user's current position. This is done using dead reckoning over a floor map and by sharing discovered location coordinates amongst the team. Our experimental results with an Android-based testbed confirmed that CIPS provides accurate localization performance with much lower position fix delay when compared to a non-collaborative scheme.

8. REFERENCES

[1] Apple, MobileMe, Web Site. `http://www.me.com`.

[2] Ekahau. `http://www.ekahau.com/`.

[3] Google, Latitude, Web Site. `https://www.google.com/latitude`.

[4] Insite. `http://www.remcom.com/wireless-insite`.

[5] Koozyt, Inc., PlaceEngine, Web Site. `http://www.placeengine.com/showe/about`.

[6] Skyhook, XPS, Web Site. `http://www.skyhookwireless.com`.

[7] Step size in pedometers. `http://walking.about.com/cs/pedometers/a/pedometerset.htm`.

[8] C.R. Anderson and T.S. Rappaport. In-building wideband partition loss measurements at 2.5 and 60 GHz. *Wireless Communications, IEEE Transactions on*, 2004.

[9] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury. SurroundSense: mobile phone localization via ambience fingerprinting. In *MobiCom*, 2009.

[10] P. Bahl and V.N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *INFOCOM*, 2000.

[11] Gaetano Borriello, Alan Liu, Tony Offer, Christopher Palistrant, and Richard Sharp. WALRUS: wireless acoustic location with room-level resolution using ultrasound. In *MobiSys*, 2005.

[12] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. Indoor localization without the pain. In *MobiCom*, 2010.

[13] Ionut Constandache, Xuan Bao, Martin Azizyan, and Romit Roy Choudhury. Did you see Bob?: human localization using mobile phones. In *MobiCom*, 2010.

[14] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 3nd edition, 2001.

[15] Philippe Dosch, Karl Tombre, Christian Ah-Soon, and Gerald Masini. A complete system for the analysis of architectural drawings. *International Journal on Document Analysis and Recognition*, 2000.

[16] A. Falsafi, K. Pahlavan, and Ganning Yang. Transmission techniques for radio LAN's-a comparative performance evaluation using ray tracing. *Selected Areas in Communications, IEEE Journal on*, 1996.

[17] Brian Ferris, Dieter Fox, and Neil Lawrence. WiFi-SLAM using Gaussian process latent variable models. In *IJCAI*, 2007.

[18] Yanying Gu, A. Lo, and I. Niemegeers. A survey of indoor positioning systems for wireless personal networks. *Communications Surveys Tutorials, IEEE*, 2009.

[19] J.R. Guerrieri, M.H. Francis, P.F. Wilson, T. Kos, L.E. Miller, N.P. Bryner, D.W. Stroup, and L. Klein-Berndt. RFID-assisted indoor localization and communication for first responders. In *EuCAP*, 2006.

[20] Youngjune Gwon and Ravi Jain. Error characteristics and calibration-free techniques for wireless LAN-based location estimation. In *MobiWac*, 2004.

[21] Yiming Ji, Saâd Biaz, Santosh Pandey, and Prathima

Agrawal. ARIADNE: a dynamic indoor signal map construction and localization system. In *MobiSys*, 2006.

[22] H. Kim and H. Ling. Electromagnetic scattering from an inhomogeneous object by ray tracing. *Antennas and Propagation, IEEE Transactions on*, 1992.

[23] J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IROS*, 1991.

[24] H. Lim, L.-C. Kung, J. C. Hou, and H. Luo. Zero-Configuration, Robust Indoor Localization: Theory and Experimentation. In *INFOCOM*, 2006.

[25] Hui Liu, H. Darabi, P. Banerjee, and Jing Liu. Survey of Wireless Indoor Positioning Techniques and Systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 2007.

[26] Matthias Lott. On the Performance of an Advanced 3D Ray Tracing Method. In *In Proc. of European Wireless & ITG Mobile Communication*, 1999.

[27] Eladio Martin, Oriol Vinyals, Gerald Friedland, and Ruzena Bajcsy. Precise indoor localization using smart phones. In *MM*, 2010.

[28] L.M. Ni, Yunhao Liu, Yiu Cho Lau, and A.P. Patil. LANDMARC: indoor location sensing using active RFID. In *PerCom*, 2003.

[29] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The Cricket location-support system. In *MobiCom*, 2000.

[30] K.A. Remley, H.R. Anderson, and A. Weisshar. Improving the accuracy of ray-tracing techniques for indoor propagation modeling. *Vehicular Technology, IEEE Transactions on*, 2000.

[31] Hyojeong Shin, Yohan Chon, Kwanghyo Park, and Hojung Cha. FindingMiMo: tracing a missing mobile phone using daily observations. In *MobiSys*, 2011.

[32] Ian Smith, Jason Tabert, The Wild, Anthony Lamarca, Anthony Lamarca, Yatin Chawathe, Yatin Chawathe, Sunny Consolvo, Sunny Consolvo, Jeffrey Hightower, Jeffrey Hightower, James Scott, James Scott, Tim Sohn, Tim Sohn, James Howard, James Howard, Jeff Hughes, Jeff Hughes, Fred Potter, Fred Potter, Pauline Powledge, Pauline Powledge, Gaetano Borriello, Gaetano Borriello, Bill Schilit, and Bill Schilit. Place lab: Device positioning using radio beacons in the wild. In *In Proceedings of the Third International Conference on Pervasive Computing*, 2005.

[33] Stephen P. Tarzia, Peter A. Dinda, Robert P. Dick, and Gokhan Memik. Indoor localization without infrastructure using the acoustic background spectrum. In *MobiSys*, 2011.

[34] R.A. Valenzuela. Ray tracing prediction of indoor radio propagation. 1994.

[35] Alex Varshavsky, Eyal de Lara, Jeffrey Hightower, Anthony LaMarca, and Veljo Otsason. GSM indoor localization. *Pervasive Mob. Comput.*, 2007.

[36] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 1992.

[37] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *Personal Communications, IEEE*, 1997.

[38] Oliver Woodman and Robert Harle. Pedestrian localisation for indoor environments. In *UbiComp*, 2008.

[39] Guang yao Jin, Xiao yi Lu, and Myong-Soon Park. An indoor localization mechanism using active RFID tag. In *Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2006.

[40] Moustafa Youssef and Ashok Agrawala. The Horus WLAN location determination system. In *MobiSys*, 2005.