Did You See Bob?: Human Localization using Mobile Phones

Ionut Constandache
Duke University
Durham, NC, USA
ionut@cs.duke.edu

Xuan Bao
Duke University
Durham, NC, USA
xuan.bao@duke.edu

Martin Azizyan
Duke University
Durham, NC, USA
martin.azizyan@duke.edu

Romit Roy Choudhury
Duke University
Durham, NC, USA
romit@ee.duke.edu

ABSTRACT

Finding a person in a public place, such as in a library, conference hotel, or shopping mall, can be difficult. The difficulty arises from not knowing where the person may be at that time; even if known, navigating through an unfamiliar place may be frustrating. Maps and floor plans help in some occasions, but such maps may not be always handy. In a small scale poll, 80% of users responded that the ideal solution would be "to have an escort walk me to the desired person".

This paper identifies the possibility of using mobile phone sensors and opportunistic user-intersections to develop an electronic escort service. By periodically learning the walking trails of different individuals, as well as how they encounter each other in space-time, a route can be computed between any pair of persons. The problem bears resemblance to routing packets in delay tolerant networks, however, its application in the context of human localization raises distinct research challenges. We design and implement *Escort*, a system that guides a user to the vicinity of a desired person in a public place. We only use an audio beacon, randomly placed in the building, to enable a reference frame. We do not rely on GPS, WiFi, or war-driving to locate a person – the Escort user only needs to follow an arrow displayed on the phone. Evaluation results from experiments in parking lots and university buildings show that, on average, the user is brought to within $8m$ of the destination. We believe this is an encouraging result, opening new possibilities in mobile, social localization.

Categories and Subject Descriptors

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

General Terms

Design, Experimentation, Measurement, Algorithms

Keywords

Localization, Mobile Phones, Sensors, Navigation

MobiCom'10, September 20–24, 2010, Chicago, Illinois, USA.

1. INTRODUCTION

The rapid growth in people-centric mobile computing applications has called for improved localization technology. The desired improvement is not only in terms of localization accuracy, but also across multiple application-specific dimensions. For instance, (1) authors in [9, 14, 16, 19, 27, 33, 36] have drawn attention to the tradeoff between energy and location accuracy. The main observation is that while GPS is fairly accurate, its continuous usage can drain a phone's battery in less than 8 hours. Alternative WiFi/GSM based schemes offer longer battery life, but at the expense of lower accuracy. (2) In another thread of research, authors argue today's indoor localization techniques are inadequate because they either require special infrastructure, or rely on continuous war-driving. They propose collaborative methods by which multiple devices can determine the WiFi placement in the building [17], or infer their relative locations with zero WiFi dependence [4, 24]. (3) Breaking away from coordinate-based localization, authors in [2, 28, 29] propose ways to identify logical locations (Starbucks, Wal-mart, RadioShack), as opposed to physical coordinates (as in GPS). The rationale is that a large class of applications do not care about the latitude/longitude of the user; instead they desire to know the "place" where the user is located. These and several other works [1, 7, 8] emphasize the broad and evolving nature of localization technology, driven primarily by the fast changing landscape of mobile, pervasive, people-centric computing.

In this paper, we expand the notion of localization to the social context. We describe the problem with a hypothetical scenario. Imagine that MobiCom 2010 is in progress in a big hotel in Chicago, and as Alice arrives, she intends to meet up with her friend, Bob. The general approach today is to either

stroll around in the hotel until Alice can visually spot Bob, or to ask "have you seen Bob around?". A phone call may be a third option, but sometimes inappropriate because Bob might be in an important meeting or attending a technical session. Even if Bob answers the phone, he may ask Alice to come over by the swimming pool, leaving Alice to locate and navigate to the (unknown) destination. Navigating through a big conference hotel, especially without precise knowledge of Bob's location, may not be a trivial task. Moreover, Bob himself might be moving, potentially making the exercise frustrating. An ideal solution would be to have a third person know the location of Bob, who can then escort Alice to him.

When presented with an informal questionnaire, 80% of the participants agreed that an escorting system would be the easiest option; some people shared anecdotes and frustrations in trying to find an individual in a hospital or a library. Motivated by this, we generalize the hypothetical scenario above to a broader problem space. We ask, *in a human populated public place, can we develop an electronic system that can localize and route a person A to a specified person B.* This form of localization and routing can become a primitive for social computing applications, useful in public settings such as airports, shopping malls, libraries, museums, and universities. We show the basic feasibility of such a system, called *Escort*, and evaluate its performance in real life scenarios.

The core idea with Escort is simple. Mobile phones capture users' "*movement traces*" using accelerometer and compass measurements. When other phones are in the vicinity, each phone also records these "*encounters*" with a corresponding time stamp. A <*movement trace, encounter*> report is periodically uploaded to a remote server. Using this information, the remote server assimilates a global view of the users' positions and paths. At any time, if A expresses interest to navigate to B, the server can create a route, composed of segments between distinct encounters. For instance, if A had met C in the past, and C met B recently, A can be routed back to the point where she met C, then routed along the path that C walked till she encountered B, and finally A can be guided along B's path to reach B's current location. This situation is depicted in Figure 1. In general, there could be a number of walkable paths between A and B – the Escort server could choose the most efficient one. This path may not be a straight line joining A and B, however, is likely to be a reliable one since all segments in that path are walkable.

Although a simple idea, translating Escort into a usable system entails a variety of challenges. (1) Accelerometers and compasses in mobile phones are noisy, causing the measured path to increasingly deviate from the true path. (2) The absence of a global reference frame disallows natural error correction. (3) Even if a user's location can be corrected at certain time points, it is non-trivial to correct her entire trail. Yet, such corrections are necessary for routing one user to another.

We cope with these challenges by assuming a fixed beacon transmitter placed at an arbitrary location, and then applying proposed methods of *location diffusion* and *drift cancellation*. The beacon's location need not be known; instead its location can be viewed as the origin of a virtual coordinate system (like a landmark). Any user that encounters this beacon can be approximately repositioned to this origin, and thus, her error

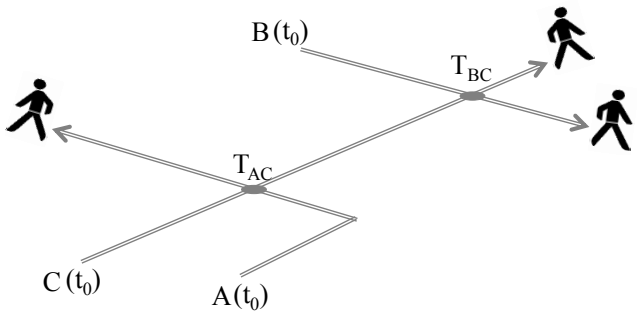

Figure 1: The Escort server assimilates a global view of users' movement and intersection patterns. The human icons show the current location of the user U, while $U(t_0)$ denotes user U's first recorded location. T_{UV} denotes the intersection time of users U and V.

gets reset. Additionally, when this repositioned user (X) encounters other users (for some time in the future), the other users can correct their own positions by exploiting the observation that X has a reasonably good estimate of her own location. In general, while the system error grows with time, the encounters among users and beacon help in keeping the error under control. Encounter detection is achieved by audio signaling; the fixed beacon and all phones periodically transmit audio tones to make their presence known. Together, these building blocks are consolidated into a prototype system implemented on a mix of Nokia and Google Android phones. Experiment results with real users exhibit promising results – on average, a user A was navigated to within 8m of the destination user B. If A is still not able to identify B (perhaps because A does not know B from beforehand), Escort offers a visual identification service. Using this, A can look at her surroundings through the phone's camera viewfinder. When B is within view, an arrow would be displayed on top of B, enabling a new form of end-to-end human localization.

We note that Escort is a simple-to-install, stand-alone system – it does not rely on meticulous calibration, war-driving, GPS localization, or any form of fixed reference frame (such as a building's floor plan). The beacon is placed at an arbitrary location and serves as an origin of a virtual coordinate system. As a result, Escort can be deployed to any location as long as an Internet connection allows access to the Escort server. In view of this advantage, we believe Escort can be a useful localization system in social environments. Our contributions may be summarized as follows.

- **We identify the problem space of social localization.** While creative ideas have been proposed in proximity sensing, the opportunity to route one individual to another enables new applications.
- **We design Escort, a war-driving free navigating system for social environments.** Mobile phone sensors, combined with inter-user/user-beacon encounters are used to form and correct a relative graph between users. Navigation is realized by routing over this graph.
- **We implemented Escort on a testbed of 4 Nokia Navigators and Google Android phones.** The core engine draws on ideas from delay tolerant networking. Performance results show the feasibility of guiding Alice all the way to Bob.

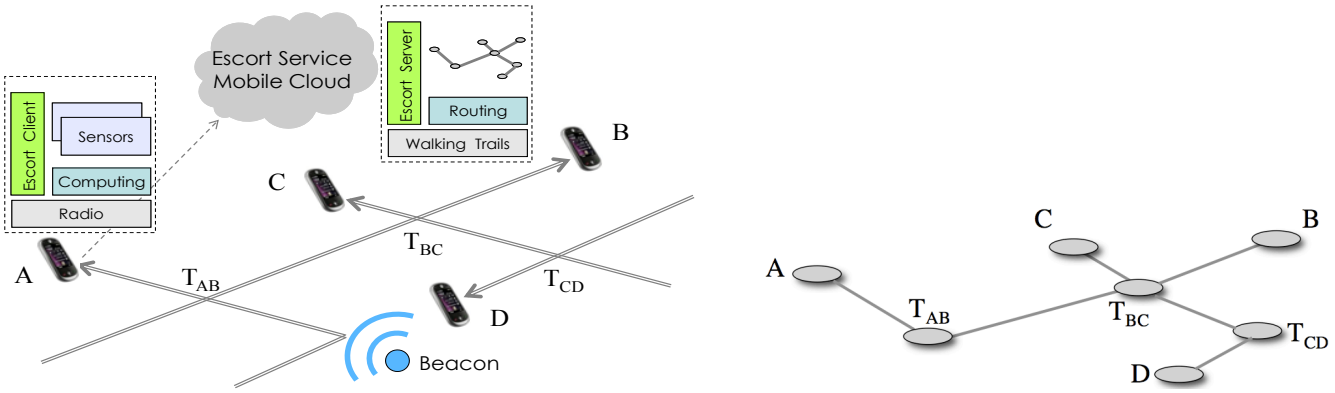

Figure 2: (a) The overview of the Escort system: Users report accelerometer and compass readings as well as user encounters; The server forms user trails. (b) The graph theoretic representation of the server's global view of the users' trails and intersections.

The rest of the paper is organized as follows. Section 2 presents a high level system overview, followed by the challenges and solutions in Section 3. The system evaluation is presented in Section 4, while limitations and future work are discussed in Section 5. Section 6 surveys the related work, and the paper concludes with a brief summary in Section 7.

2. SYSTEM OVERVIEW

Figure 2 shows the envisioned client/server architecture for Escort. We briefly describe the system components, followed by a discussion on challenges and solutions.

The Escort client periodically measures the accelerometer and compass readings from user-carried mobile phones. Since human walking patterns generate identifiable accelerometer signatures, the Escort client can compute the number of steps walked by the user. When multiplied by the user's step size [8, 13, 34], we obtain the user's approximate displacement. The compass readings offer the direction of movement, and when viewed in time-sequence, the <*displacement, direction, time*> tuples capture the user's movement pattern. We call this a user's walking *trail*.

The user trail is periodically reported to the Escort server over a WiFi/3G connection. Due to sensor noise, the walking trail drifts from the actual user movement. However, two opportunities are exploited to compensate for the drift: (i) encounters with the beacon, and (ii) encounters with users who passed the beacon recently. Both these events, if detected, provide a better estimate of the user's location. The difference between the new/better estimate and the old sensor-computed estimate presents information about the nature of the drift with compass and accelerometer. Escort uses this drift information to correct the user trail and thus calculate more accurate routes between users.

To detect encounters, the user device periodically signals its presence while also listening for other devices. Signaling happens via audio, and consists of playing a specific tone at (almost) inaudible frequencies. This tone is assigned to the user's phone at registration with the Escort service. When device A learns about its encounter with device B, device A logs this intersection and uploads it to the Escort server together with her trail. The server corrects for trail drifts and builds a global view of user movements and spatial intersections. This global view translates to a graph of paths between users. The graph vertices are spatial intersections of user paths, while the edges are walked trails between these intersections. Figure 2(b) shows this translation for the scenario in Figure 2(a).

Practical challenges arise over time because the graph grows large. Pruning heuristics are applied so that routing paths can be computed efficiently. An assumption is made that a user's trail is bidirectional, i.e., if the user has walked from north to south, then it is possible for some other user to walk in the reverse direction (south to north) along that same path. While this may not be true for one-way street traffic, we believe this holds for most walking applications.

Thus, when Alice requests to meet Bob, the Escort-computed route to Bob is returned as a sequence of displacement and angle tuples, i.e., $< step_i, \theta_i >$. The Escort client can display a compass angle, θ_i, on the phone's display, and as the user takes the i^{th} step, the following θ_{i+1} can be shown. Continuing this till the maximum value of i is expected to guide Alice to Bob's vicinity, achieving the goal of an electronic escort.

3. CHALLENGES AND SYSTEM DESIGN

The design of Escort in real environments presents practical challenges. This section characterizes the nature of these challenges and presents our approaches in addressing them.

3.1 Noisy sensors

Accelerometer

To determine a user's walking trail, her displacement needs to be computed from accelerometer measurements (Figure 3(a)). One possible solution is to double-integrate the acceleration readings. Figure 3(b) shows the unacceptable results from two test users who were asked to walk along straight lines. The phones were held in the users' hands. Evident from the results, the error (the difference between the estimated and actual displacement) increases sharply, reaching more than $100m$ only after $30m$ of actual displacement. This is an attribute of a noisy accelerometer, as well as jerky movements of the phone while carried by the user.

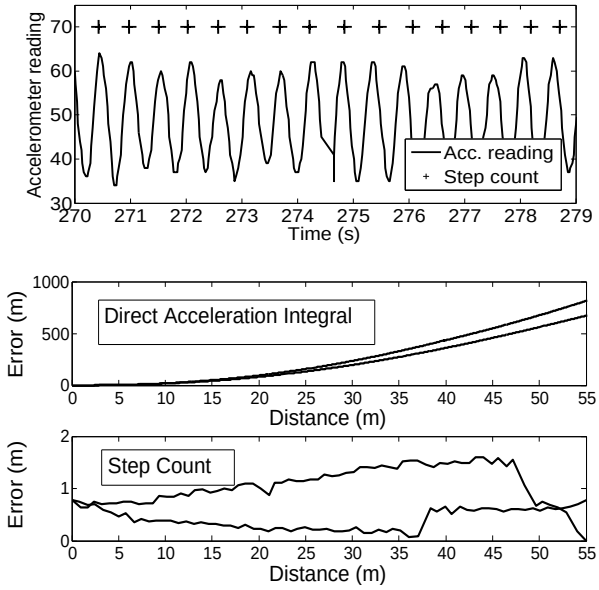

Figure 3: (a) Accelerometer readings (smoothened) from a walking user. (b) Displacement error with double integration for two users. (c) Error with step count method.

A better approach to compute displacement from accelerometers is described in [8,13]. The idea is to identify an acceleration signature in human walking patterns as in Figure 3(a). This signature arises from the natural up and down bounce of the human body while walking, and can be used to count the number of steps walked. The physical displacement can then be computed by multiplying step count with the user's step size, a function of the user's weight and height [34]. We will use this approach, but unlike the work in [8], we vary the step size with a error factor drawn from a Gaussian distribution centered on 0 and standard deviation $0.15m$. This better accommodates the varying human step size. Figure 3(c) shows the far less accumulated error with the step count method for the same 2 test users. The step count accuracy (as verified across ten other users) is 96% on average.

Compass

The compass noise is caused by several factors, including user sway, movement irregularities, magnetic fields in the surroundings, and the sensor's internal bias. Since these sources are related to the user, her surroundings, and the sensor itself, the noise is difficult to predict and compensate. For example, we have experienced both changing compass biases and large random oscillations even while a user walked multiple times on the same path.

To characterize the compass noise we ran 100 experiments using 2 Nokia 6210 Navigator phones and have observed an average bias of 8^o and a standard deviation of 9^o degrees. Figure 4 shows the compass noise distribution. In addition to this large noise range, we made two consistent observations: (1) when the user is walking in a constant direction, the compass readings stabilize quickly on a biased value and exhibit only small random oscillations, and (2) after each turn, a new bias is imposed. Based on these observations, we identify two states of the user: walking in a constant direction, and turning. Turns are identified when the compass headings change

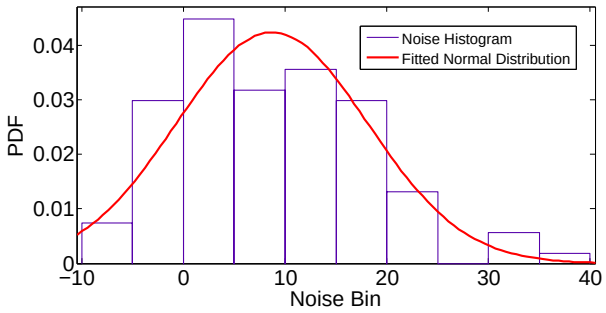

Figure 4: Compass noise distribution

more significantly than due to random oscillations. The turn identification algorithm uses the following condition:

$$Avg(t_{i+1}) - Avg(t_i) \geq \frac{StdDev(t_i) + StdDev(t_{i+1})}{2} + G$$

where $Avg(t_i)$ denotes the average compass readings over a t_i time period (e.g. 1 second), $StdDev(t_i)$ is the standard deviation of compass readings during t_i, and G, a guard factor. While on a constant direction, we compensate the stabilized compass reading with the average bias, and report the resulting value as the direction of the user. During turns, we consider the sequence of readings reported by the compass.

Compass bias and turn identification are only heuristic methods to compensate for the various sources of noise. In reality, the user's compass-corrected direction may still differ from the true direction due to an incorrect bias estimation. Errors from accelerometer-computed distances exacerbate the estimation. Our initial attempts to track the user's trail proved impractical – despite step count and compass models, the estimated location and trail diverged in time from the true location and trail of the user. In order to correct these errors, we exploit opportunistic encounters, either with peers or with the fixed beacon located in that area.

3.2 Correcting User Position via Diffusion

As mentioned earlier, the noise in the accelerometer and compass sensors cause the user position to diverge from her actual location. To correct for this, we exploit a fixed beacon, which, if in range, can be detected by Escort users. When this happens, the user is repositioned at the beacon. The beacon position is considered to be at the origin of a virtual reference frame. Locations of users are computed with the beacon at $(0,0)$ and the X and Y axis as absolute north-south and east-west directions, respectively. A user trail in the Escort virtual reference frame is represented in Figure 5.

Escort users are also repositioned when encountering other users, say Eve, who have passed the fixed beacon in the recent past (e.g., less than 1 minute ago). Thus, if Alice meets Eve, and Alice has not encountered the beacon within the last

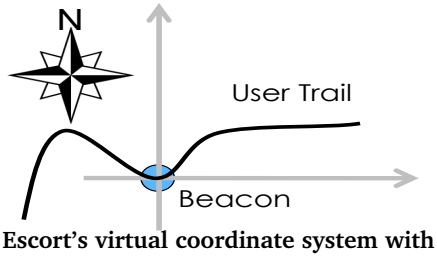

Figure 5: Escort's virtual coordinate system with the beacon at the origin.

minute, then Alice is repositioned to Eve's Escort-estimated position. Through this mechanism, fresh location information can be "diffused" into the system. Put differently, the beacon corrects the location of nearby users, who in turn correct the locations of other encountered users. This resists the system error from diverging.

In order to benefit from encounters, Escort needs to capture them accurately. An encounter may happen for only a few seconds, therefore the detection mechanism has to be reliable. A number of existing works have used Bluetooth based techniques for neighbor discovery [20,30]. Our attempts with Bluetooth failed primarily because Bluetooth is too slow for detecting short-lived encounters. Our results showed that on average, 50% of the encounters remained undetected, deeming this approach unusable. If future Bluetooth devices export more overheard information to the device drivers, then Bluetooth can certainly be applicable.

We tackle the practical problem of detecting encounters by employing audio signals. The mobile phones and the beacon are instrumented to play a unique tone, while simultaneously listening for tones from other phones. The tones are on a specific narrow-band frequency; the selected frequencies are almost outside the audible range in our implementation, and can be made inaudible for actual deployments. A natural question is whether tone detection is reliable under different noise scenarios. To verify this, we ran tone detection experiments in three locations: a student dormitory with heavy AC noise, a meeting room with several people talking, and a shopping mall with loud white noise. The phones were placed close to each other, approximately $1m$ apart. Figure 6 shows that the tone frequencies (vertical dotted lines) are clearly distinguishable from ambient sounds.

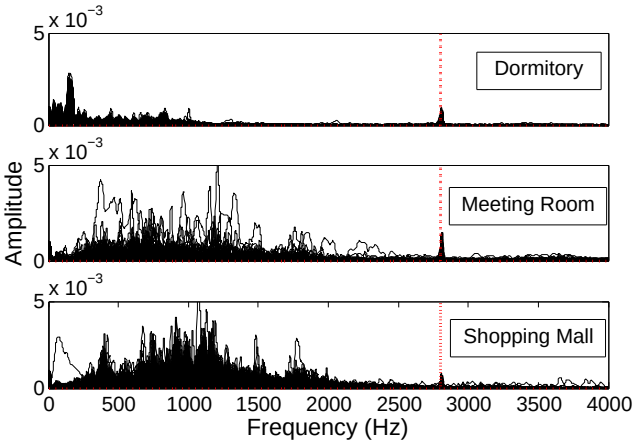

Figure 6: Tone detection in three test scenarios: (a) student dormitory, (b) meeting room, and (c) shopping mall. The red dotted line marks the tone frequency.

Detecting audio tones (transmitted by the fixed beacon or other phones) is not sufficient. Escort needs to ensure that the tone transmitter is nearby. otherwise, repositioning a user to the tone transmitter's position can introduce large errors. To this end, we investigated the possibility of computing the transmitter-receiver distance from the tone amplitudes (Figure 7). Evidently, the amplitude does not exhibit a clear trend with increasing distance; instead we observed a clear reduction in amplitude at 5m. We observed this reduction consistently, and thereby thresholded it. Only if users overhear tone amplitudes greater than this threshold (i.e., the transmitter is roughly within 5m), the encounter is registered. Repositioning users based on such encounters will naturally incur this 5m error. Ideas from [24,25] may lower this error, and Escort can certainly benefit from them.

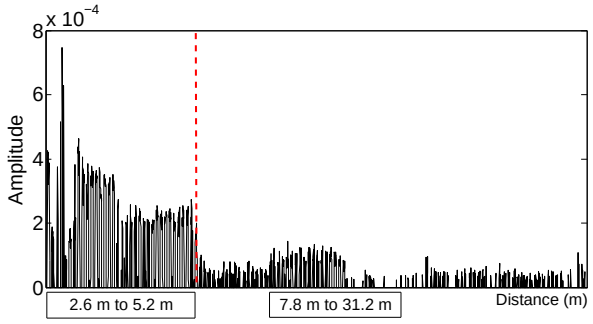

Figure 7: Variation of tone amplitude with distance – two classes of distances are marked.

As mentioned earlier, the fixed beacon is the origin of the virtual coordinate system. A relevant question is *how do users join this coordinate system?* When the Escort client starts, the user trail is initially computed in an isolated coordinate system that is only relative to the initial (unknown) location of the user. However, encounters with users who encountered the fixed beacon, provide opportunities to bring this new user into the Escort's coordinate system (with the beacon at the origin). Escort coalesces the position of a new user to that of an encountered user who is already tracked by Escort. As a generalization, coordinate systems can be coalesced even if none of the users have met the beacon. Later, when any one of these users encounters the beacon, the entire coordinate system gets updated with the beacon as the origin.

3.3 Correcting User Trail (Drift Cancellation)

Correcting the instantaneous location of a user (as discussed above), provides an opportunity to correct the user's past trail as well. Consider Figure 8. The solid black line depicts the user's true path, and the dotted line depicts the user's computed trail (using accelerometer and compass). At the beginning of this trail, at time t_{r1}, the user encounters the beacon and repositions herself. Since the estimated location right after the reposition is likely to be accurate, the starting point of this trail can be "nailed" with good accuracy. At some later time on this trail, another reposition operation happens at time t_{r2}. At this time, we learn how much correction we need to apply to the estimated location at time t_{r2} to reposition the user. We mark this correction vector with $\vec{V}$, which is an estimation of the cumulative drift over time. Since the estimated position after repositioning at t_{r1} (the starting point) is accurate, the overall drift is accumulated only during time $t_{r2} - t_{r1}$. Based on this observation, we can correct the trail between two consecutive re-positionings. We call this method "Drift Cancellation".

The key idea of drift cancellation is to amortize the correction vector $\vec{V}$ over time. We assume that the user's projected path deviates from the true path linearly over time. For ex-

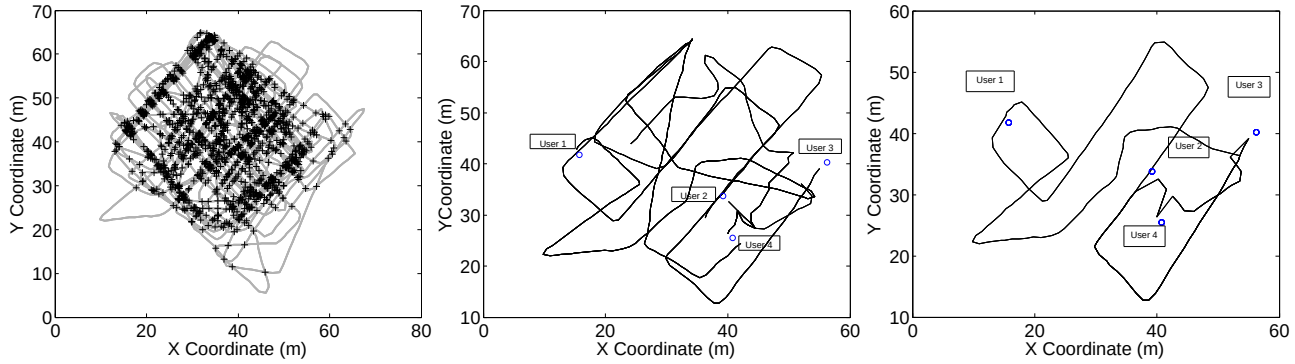

Figure 9: (a) The trail graph after tracking 4 users for 10 minutes. (b) The resulted graph after applying the pruning heuristic. (c) Running Floyd-Warshall and the graph of user paths.

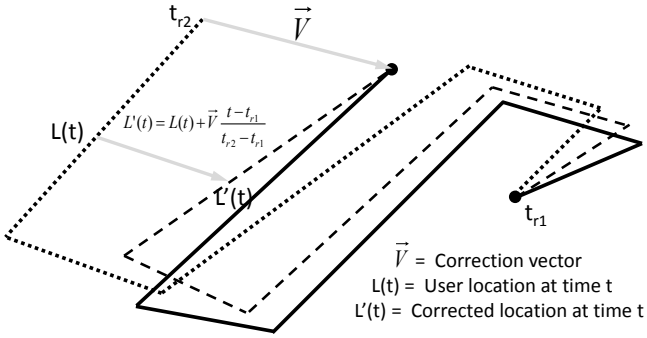

Figure 8: Drift Cancellation: the solid line is the actual user path, the dotted line represents the user computed trail, and the dashed line denotes the user trail corrected via Drift Cancellation.

ample, if the time elapsed between two re-positionings was 100 seconds, then, by this assumption, the correction vector at 50 seconds is exactly half of $\vec{V}$. Mathematically, drift cancellation can be described as follows. Let $L(t)$ be the user estimated location at time t. Then for $t \in (t_{r1}, t_{r2})$, we define the corrected location as:

$$L'(t) = L(t) + \vec{V}\frac{t - t_{r1}}{t_{r2} - t_{r1}}$$

In Figure 8, the result of applying this correction to the trail is represented as a dashed line. We expect this to better approximate the true trail, shown in solid black.

3.4 Computing Routing Directions

Escort monitors user trails continuously. In the background, the server computes the current position of each user, together with the spatial intersections of the users' trails. Based on this, Escort builds a "trail graph". The graph edges are segments of user trails, while the vertices are either spatial intersections between trails or the current user locations. Figure 2 showed the abstraction and Figure 9(a) shows a trail graph from our actual system. Since the number of vertices and edges grow over time, the trail graph quickly becomes dense (Figure 9(a) is a result of 10 minutes of operation). Since maintaining such a graph (and routing over it) becomes inefficient, Escort employs a graph pruning heuristic as described next.

Without loss of generality, assume that Escort is running in the steady state, i.e., all users have joined the Escort coordinate system. To explain the pruning heuristic, let us consider two users A and B, and their respective Escort estimated paths P_A and P_B (Figure 10). Let us say that P_A and P_B intersect each other at several positions (I and J in Figure 10). Observe that the intersections are "spatial", meaning that the users may have crossed these locations at different points in time. Consequently, the intersections are at different distances with respect to each user – I is closer to user A, while J is closer to B. Escort's pruning heuristic selects the closest (spatial) intersections for both A and B, and retains them as vertices in the trail graph. Other intersections are eliminated. The two paths joining the two intersections are also retained in this phase. This process is repeated for every pair of nodes in the system, resulting in a fully connected graph as shown in Figure 9(b). This graph is then pruned again by applying the Floyd-Warshall algorithm and keeping the graph of the shortest paths between users. The new smaller graph is shown in Figure 9(c). As users move, Escort will keep adding edges and vertices to the trail graph, however, it will periodically re-apply the pruning heuristic. With the trail graph pruning mechanism in place, Escort routing directions can be computed efficiently and presented to the requesting users.

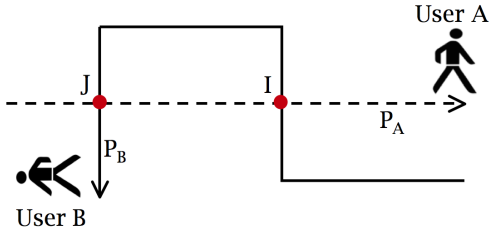

Figure 10: Spatial intersections between users.

3.5 Visual Identification

Assume that Escort has routed Alice to within a few meters of Bob. If they are acquainted, Alice will easily identify Bob. Otherwise, Alice may be unable to identify him, especially if multiple people are present around her. A possible solution is for Alice to acquire a picture of Bob, and attempt manual identification. However, such a picture may not be easily available. Even if an Internet search returns a few pictures, Bob may not be facing Alice, making manual face recognition a cumbersome task. To solve the human localization problem

"end-to-end", Escort should be able to pinpoint Bob, so Alice knows exactly whom to approach. We propose the following solution that users may optionally use in certain mutually trusted environments, such as a MobiCom conference.

For a given event/venue, each mobile phone will opportunistically take pictures of its owner. The phone may be able to infer when the user is facing the device by leveraging the second camera on the front of the phone (iPhone 4 already has a front-facing camera to support video conferencing). The pictures need not be of high quality – they may be partial, oriented incorrectly, or include parts of other people and objects. However, out of a multitude of pictures, the characteristics of the user's appearance, such as clothing color and texture, can be summarized into a "fingerprint". If the <user's name, fingerprint> tuple is periodically beaconed, other mobile devices can utilize them to pinpoint the individual.

In an ideal setting, where Alice wants to identify Bob, the system would require her to turn on the camera and look at the surrounding people through the phone's viewfinder. Recorded images will be analyzed and each individual in the picture isolated through image processing (edge detection and segmentation algorithms). The fingerprint of the isolated individuals can be computed (denoted as test fingerprints) and compared against the beaconed fingerprint from Bob. The test fingerprint that best matches Bob's can be pinpointed on the phone's viewfinder. Figure 11 illustrates the idea – Bob is pinpointed with an arrow. We envision this to happen in real time, perhaps by engaging cloud services for the CPU-intensive image processing algorithms [10]. In this paper, however, we have addressed the problem offline, using some degree of user participation. The methods are described next.

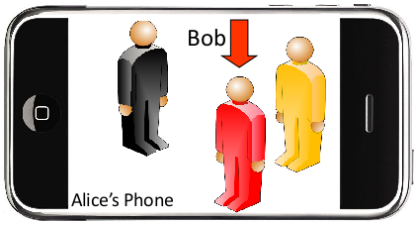

Figure 11: Camera-based user identification.

Visual identification entails three main challenges: (1) *segmentation*: automatically cropping out the human boundaries in an image, (2) *fingerprinting*: creating a feature set of colors for a cropped segment, and (3) *matching*: an algorithm to compute the similarity of two fingerprints.

(1) Segmenting images based on human contours is a well-studied problem in computer vision [11, 31]. These solutions may be applied in our case too. However, instead of automated segmentation, we simplify the problem by asking Alice to select all regions on the viewfinder where some person is located. Assuming multi-touch input technology, this translates to swiping the finger on different parts of the screen. The swiped regions are cropped out and processed to build the fingerprints of these people.

(2) Fingerprinting translates to extracting a set of (color) features that will discriminate that person from others. For this, the cropped region is divided into an upper and lower subregion, corresponding to the upper and lower parts of the person's body. This process is illustrated in Figure 12. Next, the pixels in each subregion are transformed into the hue-saturation-value (HSV) space (instead of RGB) to better cope with lighting variations, shadows, etc. Then, three-dimensional Gaussian distributions [5] are fitted to the upper and lower subregions using maximum likelihood estimation (MLE). The parameters (mean and variance) of the upper and lower Gaussians form the fingerprint of the person's image. Bob's phone computes this fingerprint from Bob's pictures and beacons it along with his name. Alice's phones computes these fingerprints for all the people in her surroundings – we refer to these as "test fingerprints".

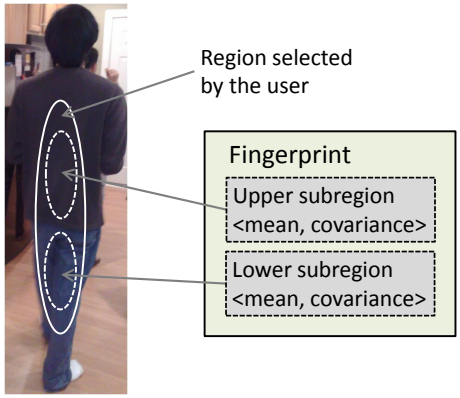

Figure 12: Generating the user fingerprint.

(3) Matching requires comparing the test fingerprints to Bob's fingerprint (since Alice is near Bob, her phone should have overheard his beacons). We employ the following fingerprint matching heuristic. For each test fingerprint, we compute the similarity values S_{up} and S_{low} of the upper and lower Gaussians. We measure the similarity of two Gaussians as the volume of their intersection:

$$S_i = \int min(f(X|\mu_{test\_i}, \sigma_{test\_i}), f(X|\mu_{recv\_i}, \sigma_{recv\_i}))\, dX$$

where $i = up, low$ and μ_{test_i}, μ_{recv_i} and σ_{test_i}, σ_{recv_i} are the mean and variance of the Gaussians in the test and received fingerprints, and $f(X|\mu, \sigma)$ is the density of the normal distribution at point X. Then, we rank all test fingerprints based on the product of the two similarity values $S_{up} * S_{low}$, and declare the top-ranked fingerprint to be the "best" match. Finally, we return to the corresponding region of the image, and display "Bob" with an arrow. The effect is as in Figure 11.

3.6 Running Escort in the Cloud

Escort relies on users to periodically upload their compass, accelerometer and sound recordings. The service can be implemented in a computing cloud. In such a context, the computational cost for routing is not a concern. Similarly, FFT transforms of sound recordings and image color processing can be computed quickly. In our tests, we instantiated the Escort service on an off-the shelf laptop and monitored 4 users. Under this small load we experienced no performance issues.

4. EVALUATION

4.1 Testbed

To evaluate Escort, we need the accurate location of each user (the ground truth). Otherwise, we cannot quantify both the errors of the Escort-computed positions and the final distance between an escorted person and her destination (e.g., Alice and Bob). Since the system targets meter-scale accuracy, the real position of the user should be estimated within a meter. The first solution we considered was to track users through GPS. However, our experiments showed consistent errors of $10m$, making GPS impractical for our requirements. In response, our solution was to run experiments in an area containing markers with known positions. In this scenario, users move between markers and manually record each marker they pass, providing us with the ground truth.

Deploying a grid of markers is challenging because of two constraints: (i) we require a dense deployment of markers, with small distances between them, so that we can track the user at a fine granularity (ii) we need accurate marker positions to evaluate Escort's instantaneous error and final-distance error after routing to a destination.

We have found a convenient way to fulfill these requirements using a university parking lot. We placed markers at the intersections and end-points of the white lines delimiting the parking spots. Since the parking lot has several parallel lines, the number of markers we obtained was large, with an average distance between adjacent markers of around $5m$. The markers' positions are obtained through their GPS coordinates from the Google Maps Satellite View. Figure 13 shows the parking lot and some of the markers in white crosses.

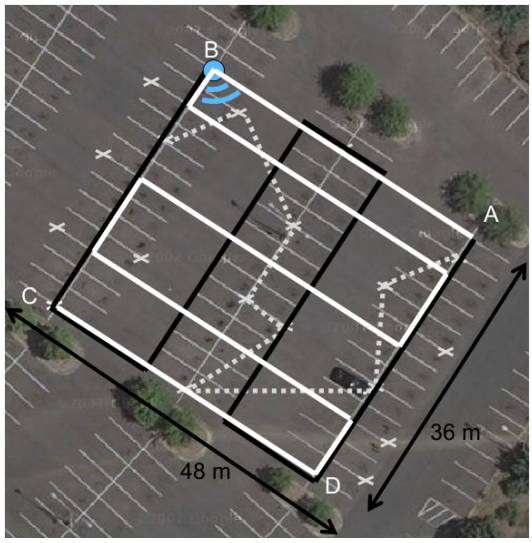

Figure 13: Parking Lot Grid. White crosses indicate the position of some of the markers in our experiments. The solid white and black lines show the fixed trails walked by two of our test users. The dashed line represents part of a random user's trail. The beacon is placed at position B.

We also ran experiments indoors, in one floor of our department building. To obtain a ground truth, we applied the same solution as employed in the parking lot – we deployed 7 markers inside the building and determined their GPS positions using Google Satellite View. The position of the markers and the beacon are shown in Figure 14, along with one of the walked paths.

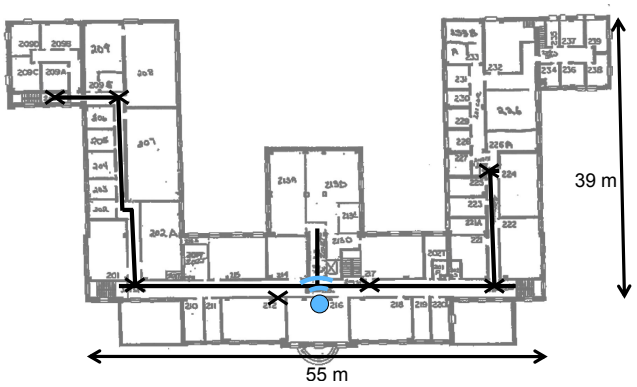

Figure 14: Indoor Grid. Black color crosses indicate the positions of the markers. The solid black lines show paths walked by two of our test users. The beacon is placed at the center of the building.

4.2 Methodology

In order to test the system across a range of users, movements, and encounter patterns, we split the evaluation experiments in two stages. We call these stages: *data collection* and *user routing*. In the *data collection* stage a group of N users register with the Escort service and are monitored for some time T. The users move between markers in preset or randomly chosen paths. Each user has a "shadowing" person who records the markers and the time these markers were crossed. When the data collection phase ends at time T, we have the movement and position history of each person. Note that the markers' GPS positions are translated into Escort's coordinate system. In this way, we obtain both the computed locations and the actual positions of the user in the Escort coordinate system. At any given time, the difference between these two locations is Escort's *instantaneous error*, and is a fine grained indicator of Escort's performance.

The efficacy of *routing users* is also of interest. For this, Escort first computes the trail graph between all users at time t. Then, it provides routing directions to A to navigate to B (we assume that B is static at its known position at time t). The routing directions are a sequence of displacement and angle tuples, i.e., $< step_i, \theta_i >$. Once A finishes following these instructions, we measure the final distance between the real positions of A and B.

Our parking lot experiment had $N = 4$ users (and 4 shadow persons). Users carried the phones in hand, with the screen facing up. The data collection phase was $T = 13$ minutes long, and we executed 40 routing experiments in total. For the indoor experiment, we used $N = 2$ users, a data collection phase of $T = 6$ minutes, and we ran 10 routing experiments.

4.3 Results

Our evaluation results concentrate on two metrics of the system: (1) the variation of instantaneous location error as a function of time, and (2) the error at the end of a rout-

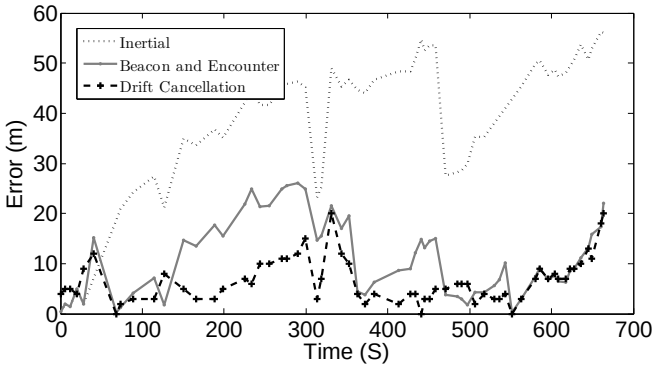

Figure 15: User 2 instantaneous error with time

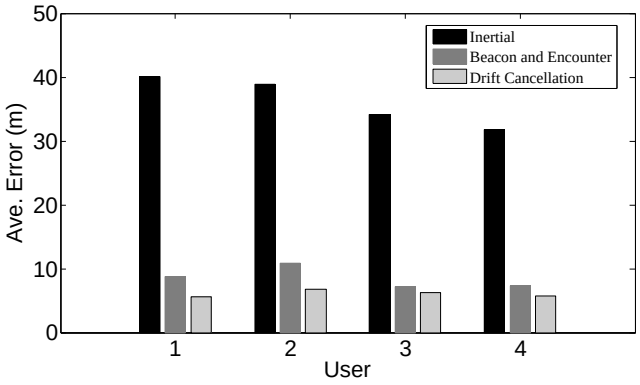

Figure 16: Average instantaneous error per user.

ing stage, i.e., the final-distance between the escorted person and her destination. We present results from the parking lot experiment first, followed by the indoor environment. Thereafter, we report the accuracy of visual identification (through the phone's viewfinder).

Parking Lot Experiment

Figure 15 shows how a user's instantaneous error evolves with time for three different schemes: (1) Inertial, (2) Beacon and Encounters, and (3) Drift Cancellation. The Inertial scheme shows how error grows if only the compass and accelerometer sensors are used for localization. Evident from the graph, the error grows quickly due to the inertial drift of these sensors. The Beacon and Encounters scheme has better performance, as the instantaneous error drops every time a beacon is observed or an encounter happens, resulting in user repositioning. Drift Cancellation further reduces the error by correcting users' traces based on the drift vector estimated at the time of repositioning.

Figure 16 shows the average instantaneous error of the three schemes for each user. The average results across all users are $36.2m$, $8.5m$ and $6.08m$ for the Inertial, Beacon and Encounters, and Drift Cancellation schemes, respectively. The average errors are computed across the entire experiment length. To understand the distribution of instantaneous error, we plot the CDF in Figure 17. As shown, the instantaneous error is less than $10m$ in 68% of the cases. After drift cancellation, 84% of the cases exhibit less than $10m$ error.

Next, we evaluate the accuracy of escorting a user A to B. The metric of interest is the final-distance between A and B after escorting is completed. The average value recorded across all experiments was $8.2m$. Figure 18 shows the distribution of the final-distance.

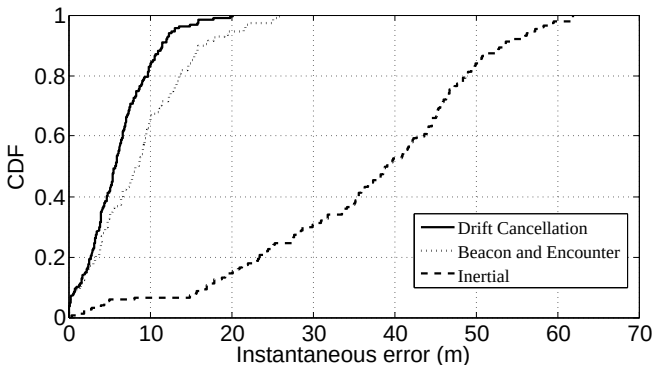

Figure 17: CDF of the instantaneous error in parking lot.

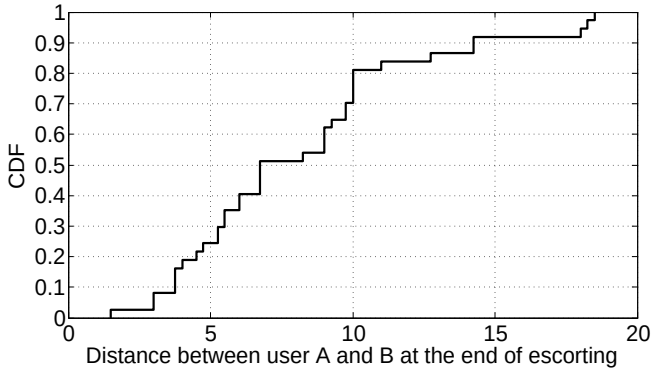

Figure 18: CDF of the final-distance error after routing in the parking lot.

We qualitatively show the Escort route between two random users A and B. Figure 19 shows three examples of routing trails. The shapes of these trails partly outline the patterns of user movement. In an indoor environment, these shapes may mimic the floor plan of the building.

Indoor Experiment

The instantaneous location error is lower in the indoor environment than in the parking lot. This is caused by the shape of the building which confines the user movements to corridors and rooms (as opposed to any random path in the parking lot). Drift cancellation adds to the benefits, achieving around $7m$ accuracy in 80% of the cases.

Table 1 summarizes the final distance between the escorted person and her destination. The final-distance is low in general with an average of 4.03m. In some of these traces, however, the turning points were slightly over or underestimated due to some residual noise in the trails. Nevertheless, even with these small navigation inconsistencies, humans were able to guess what Escort is trying to convey; when Escort requested the user to turn few meters ahead of an actual corridor turn, the human ended up taking the correct turn. We believe that offering a visual representation of the entire path will facilitate such forms of fault-tolerance (somewhat similar to in-car GPS systems).

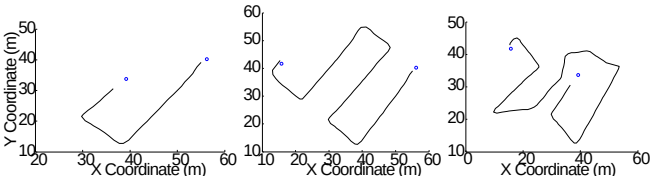

Figure 19: Routing trails between users.

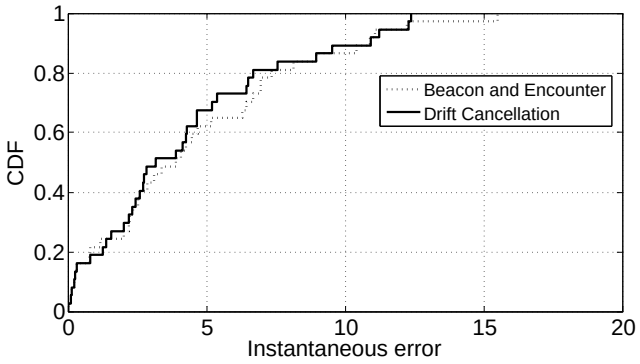

Figure 20: CDF of the instantaneous error in the indoor environment.

Routing experiments 3 and 6 in Table 1 show examples of corner cases. In experiment 3 we have a final-distance of $12m$, primarily because the instantaneous error was large. Thus, even if the routing directions were correct, the user ended several meters short from her destination. In experiment 6, the user could not be routed at all. A large compass bias made the destination user appear as walking through walls, and hence, Escort asked the seeking user to also turn into the wall. Since no turns were in close range, the user could not be routed. However, in a following experiment, at a later time, the user could be routed successfully to her destination. This happens when Escort finds another route, or when another person walking through the same path, provides better movement trails.

Test	1	2	3	4	5	6	7	8	9	10
Err	5.2	3.2	12	0.8	3.7	X	5.2	1.5	3.7	1

Table 1: Final-distance error in meters after routing indoors. X denotes user could not be routed.

We modeled the user movements and encounters from the users' movement traces, and used it to simulate Escort. The simulation mimics a scaled version of the parking lot experiment. Figure 21 shows the variation of instantaneous error (averaged over time and 4 users) for random placements of the fixed beacon. Evidently, the error remains quite stable, demonstrating that Escort's performance is not too sensitive to the beacon placement. Of course, the routing performance could not be evaluated in the simulator because it needs a human being to follow the Escort instructions.

People Identification Accuracy

We evaluated the accuracy of identifying people through the camera viewfinder using pictures from social events. We used these pictures to simulate different configurations of test and beaconed fingerprints. Figure 22 shows the average identification accuracy as a function of the number of beaconed fingerprints in the surroundings. The difficulty of the identification task increases with the number of people. Nevertheless, even with 8 people in the surroundings, our accuracy is close to 80%. These results, combined with navigation accuracy to within $8.2m$ of the destination, is expected to offer an end-to-end solution to human localization.

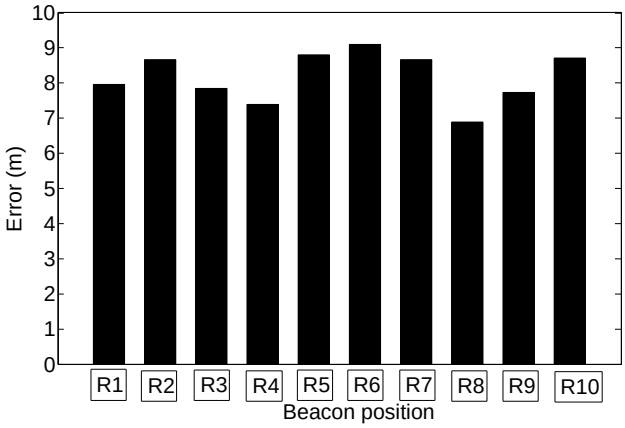

Figure 21: Average instantaneous error across all users with random placements of the beacon.

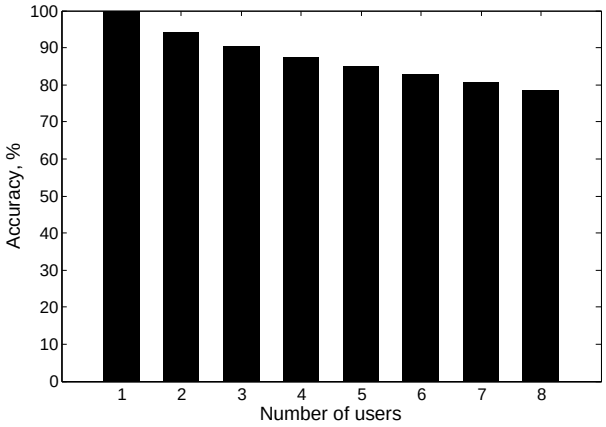

Figure 22: Accuracy of visual identification using the phone camera.

5. LIMITATIONS AND FUTURE WORK

Energy. Escort is not designed for energy efficiency – the accelerometer, compass, and acoustic signaling will certainly consume battery power. Nevertheless, there are various opportunities for system optimization. When users are not moving, Escort can, for example, turn off all the sensors except the accelerometer (which has a reasonably low power consumption). Once the user moves again (detectable by the accelerometer), all other sensors can be turned back on. With many users in a place, each user can considerably reduce the periodicity of audio signaling; in fact placing more fixed beacons further facilitates this opportunity. Also, frequent uploading of sensor data to the Escort server is not required unless someone is being escorted to this user. An energy-efficient version of Escort is left for future research.

Routing through physical obstacles. Escort may give directions to turn through obstacles or walls. However, if Escort

directions reflect a good approximation of the route, humans can make educated decisions about which path to follow. For example asking to take a turn before a hallway, may not be an issue, if the user is within a small distance of the turn. In addition, visual representations of the escorted path may help in guiding the user's decisions.

Long Routing Paths. Escort may route Alice on a longer path even though Bob may be close by. This happens when Escort is not aware of a possible trail along the shortest path (see Figure 19(b)). However, this may be indicative of a blockage on that path. As an optimization, Escort can provide a straight line route between users, since it knows their positions. Then, the escorted user may either follow the direct path or the Escort computed one. We plan to investigate the problem of "best" routing paths as part of our future work.

Routing Instructions under Low Location Accuracy. Routing a user when her position is not accurate may cause Escort to give poor navigation instructions. Nevertheless, Escort has knowledge of the last time the user received fresh location information. If this encounter is too far in the past, the system may instruct the user to walk in a direction in which she can find fresh location information (e.g., ask the user to move towards the direction of a crowd). Once better position information becomes available, Escort may recompute the route directions and give the user a better path to her destination.

Phone Placement. Our assumption of carrying phones in hand may be optimistic. In general, inferring the orientation of the phone is non-trivial. Nevertheless, Nericell [21] showed how to infer the position of the phone relative to a reference frame (a car driving on the road) using the phone accelerometer. A recent project titled "Which Way Am I Facing?" [18] achieved good results in deriving the compass orientation from a randomly-oriented wearable device. We believe these works, coupled with the arrival of gyroscopes in mobile phones, will provide new opportunities to estimate the phone orientation. We plan to address these issues in future work.

Behavior under heavy user load. We think Escort accuracy will improve with a large number of users and increased mobility. We argue that under such scenarios, Escort users will have a higher chance to pass each other or encounter the beacon. Similarly, placing the beacon in a popular place, or adding extra beacons, may provide additional opportunities to correct and diffuse location information. Nevertheless, since Escort is an early attempt in human localization, we plan to investigate scalability issues as part of future work.

6. RELATED WORK

Previous research considered user localization, but did not address the challenges of providing directions to localize people. Even if accurate user location is available, challenges remain in navigating a user to a destination. Escort is a prototype addressing this issue.

Previous localization solutions considered deploying radios or specialized hardware in the environment to assist localization. The user position is estimated based on the overheard signals and the data collected during a calibration phase. Cricket [26], VOR [23] and Pinpoint [35] rely on these techniques. Unlike these systems, Escort requires minimal hardware support: compass and accelerometer sensors in off-the-shelf mobile phones, and a beacon, e.g., an audio device.

Radar [3], Active Campus [15] and PlaceLab [6] rely on access points already present in the surroundings to enable localization. These solutions require calibrating WiFi signal strengths at many physical locations. The calibration process is time-consuming and may not scale over large areas. Unlike these systems, Escort does not require war-driving or signal calibration. Our approach is straightforward: we rely on trails and encounters to achieve localization and provide routing directions between users.

Another line of research uses floor plans and maps to assist in user localization. Authors in [13] rely on a floor plan coupled with WiFi war-driving and inertial sensors to enable localization. CompAcc [8] is an outdoor localization scheme which builds a user trail similar to Escort. However, CompAcc provides localization by using a combination of GPS and paths retrieved from Google Maps. Escort does not require GPS, war-driving, maps or floor plans. Escort builds its own coordinate system in which it estimates the user location and her trails.

Finally, related work in robotics investigated the problem of simultaneous localization and mapping (SLAM). In this line of research a robot builds a map of the surroundings and localize itself using this map. The key insight behind SLAM is to identify a number of landmarks and use them as reference points. Thus, SLAM translates into solving two distinct problems [32]: (i) identify the landmarks and (ii) approximate the range and bearing to/from these landmarks. Real life deployments solve the two problems jointly, using range sensors, such as lasers [22] or radar [12]. Note that our system relies solely on sensors available in mobile phones, less accurate than radar or laser ranging. Further, landmark identification may be problematic for mobile phones. Unlike SLAM, Escort does not build a map of landmarks. For our purposes localization/escorting happens only along the paths previously walked by the users.

7. CONCLUSIONS

Recent years have seen multiple branches of research in localization technology. This paper identifies the problem of social localization, where the goal is to help a person, Alice, navigate to another person, Bob. Our key observation is that knowledge of users' absolute locations are not necessary. Instead, we show that by recording users' movement patterns (via phone compasses and accelerometers) and mutual encounters (via audio signaling), a graph can be created on a common coordinate system. Using such a graph, a human can be routed to any other human. We use these intuitions to develop Escort, and demonstrate the feasibility of escorting a user to within $8m$ of her destination. We believe a mature version of such a technology can enable a new class of social proximity based applications.

8. ACKNOWLEDGEMENTS

We sincerely thank our shepherd Srdjan Capkun and the anonymous reviewers for their invaluable feedback which helped improve this paper. We also thank Justin Manweiler, Naveen

Santhapuri, Souvik Sen, Sandip Agrawal, Joe Levy, Ga-Young Joung, Trevor Narayan and Bogdan Romanescu for the numerous discussions and their help during the research and evaluation phases of Escort. We are also grateful to Nokia and NSF for partially funding this research through grants CNS-0916997 and CNS-0910846.

9. REFERENCES

[1] G. Ananthanarayanan, M. Haridasan, I. Mohomed, D. Terry, and C. A. Thekkath. Startrack: a framework for enabling track-based applications. In *ACM MobiSys*, 2009.

[2] M. Azizyan, I. Constandache, and R. R. Choudhury. Surroundsense: Localizing mobile phones via ambience fingerprinting. In *ACM MobiCom*, 2009.

[3] P. Bahl and V. N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *IEEE Infocom*, 2000.

[4] N. Banerjee, S. Agarwal, P. Bahl, R. Chandra, A. Wolman, and M. Corner. Virtual compass: relative positioning to sense mobile social interactions. In *Pervasive*, 2010.

[5] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[6] Y. Chen, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *ACM MobiSys*, 2005.

[7] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *ACM MobiSys*, 2005.

[8] I. Constandache, R. R. Choudhury, and I. Rhee. Towards mobile phone localization without war-driving. In *IEEE Infocom*, 2010.

[9] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox. EnLoc: Energy-efficient localization for mobile phones. In *IEEE Infocom Mini Conference*, 2009.

[10] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *ACM MobiSys*, 2010.

[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[12] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Trans. on Robotics and Automation*, 2001.

[13] F. Evennou and F. Marx. Advanced integration of wifi and inertial navigation systems for indoor mobile positioning. *EURASIP J. Appl. Signal Process.*, 2006.

[14] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt. Micro-blog: Sharing and querying content through mobile phones and social participation. In *ACM MobiSys*, 2008.

[15] W. G. Griswold, P. Shanahan, S. W. Brown, R. Boyer, and M. Ratto. Activecampus - experiments in community-oriented ubiquitous computing. *IEEE Computer*, 2003.

[16] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær. Entracked: energy-efficient robust position tracking for mobile devices. In *ACM MobiSys*, 2009.

[17] J. Krumm and K. Hinckley. The nearme wireless proximity server. In *Ubicomp*, 2004.

[18] K. Kunze, P. Lukowicz, K. Partridge, and B. Begole. Which way am i facing: Inferring horizontal device orientation from an accelerometer signal. In *IEEE International Symposium on Wearable Computers*, 2009.

[19] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *ACM MobiSys*, 2010.

[20] L. McNamara, C. Mascolo, and L. Capra. Media sharing based on colocation prediction in urban transport. In *ACM MobiCom*, 2008.

[21] P. Mohan, V. Padmanabhan, and R. Ramjee. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *ACM SenSys*, 2008.

[22] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the National Conference on Artificial Intelligence*, 2002.

[23] D. Niculescu and B. Nath. VOR base stations for indoor 802.11 positioning. In *ACM MobiCom*, 2004.

[24] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan. Beepbeep: a high accuracy acoustic ranging system using cots mobile devices. In *ACM SenSys*, 2007.

[25] C. Peng, G. Shen, Y. Zhang, and S. Lu. Point & connect: intention-based device pairing for mobile phone users. In *ACM MobiSys*, 2009.

[26] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *ACM MobiCom*, 2000.

[27] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely. Energy-delay tradeoffs in smartphone applications. In *ACM MobiSys*, 2010.

[28] N. Ravi and L. Iftode. Fiatlux: Fingerprinting rooms using light intensity. In *Pervasive*, 2007.

[29] N. Ravi, P. Shankar, A. Frankel, A. Elgammal, and L. Iftode. Indoor localization using camera phones. In *IEEE Workshop on Mobile Computing Systems and Applications*, 2006.

[30] T. J. Smith, S. Saroiu, and A. Wolman. Bluemonarch: a system for evaluating bluetooth applications in the wild. In *ACM MobiSys*, 2009.

[31] K. Tabb, N. Davey, S. George, and R. Adams. Detecting partial occlusion of humans using snakes and neural networks. In *Conference on Engineering App. of Neural Networks*, 1999.

[32] S. Thrun and J. J. Leonard. *Handbook of Robotics, Chapter 37, Simultaneous Localization and Mapping*. Springer, 2008.

[33] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *ACM MobiSys*, 2009.

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

[35] M. Youssef, A. Youssef, C. Reiger, A. Shankar, and A. Agrawala. Pinpoint: An asynchronous time-based location determination system. In *ACM MobiSys*, 2006.

[36] Z. Zhuang, K.-H. Kim, and J. P. Singh. Improving energy efficiency of location sensing on smartphones. In *ACM MobiSys*, 2010.