Pedestrian Localisation for Indoor Environments

Oliver Woodman
Computer Laboratory
University of Cambridge
ojw28@cam.ac.uk

Robert Harle
Computer Laboratory
University of Cambridge
rkh23@cam.ac.uk

ABSTRACT
Location information is an important source of context for ubiquitous computing systems. This paper looks at 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. We show how to handle multiple floors and stairways, how to handle symmetry in the environment, and how to initialise the localisation algorithm using WiFi signal strength to reduce initial complexity.

We evaluate the entire system experimentally, using an independent tracking system for ground truth. Our results show that we can track a user throughout a $8725\,\mathrm{m}^2$ building spanning three floors to within $0.5\,\mathrm{m}$ 75% of the time, and to within $0.73\,\mathrm{m}$ 95% of the time.

ACM Classification Keywords
D.2.8 Mathematics of Computing: Probability and Statistics—*probabilistic algorithms*

General Terms
Algorithms, measurement, experimentation

Author Keywords
Inertial tracking, particle filters, localisation

INTRODUCTION

Some of the first ubiquitous computing systems to come out of research laboratories made use of location information to provide useful clues as to the context a person or device was situated within. Today, GPS provides localisation outdoors, but precise indoor tracking of people remains an open research problem. We have seen indoor location systems based on infra-red, ultrasound, narrowband radio, WiFi signal strength, UWB, vision, and many others [11]. However, few can be easily deployed over large buildings whilst still providing accurate localisation.

UbiComp'08, September 21-24, 2008, Seoul, Korea.

To minimise deployment and infrastructure costs, we wish to develop a wearable location system that can position itself absolutely within a complex structure. The immediate parallel is with robotics, where mobile devices typically use inertial sensors, laser range-finders and computer vision to provide accurate localisation without the requirement of fixed infrastructure. Applying the same systems to people is, however, fraught with difficulties; laser range-finders and cameras are impractical, we lose the ability to control the subject to maximise our chances of precise localisation, and the techniques are usually developed with a single floor in mind (and certainly no stairs!). One type of sensor which does seem applicable to people tracking is inertial measurement units (IMUs). Recent advances in micro-electro-mechanical systems (MEMS) technologies have made such units smaller and cheaper, however also more prone to error. Accurate people tracking in a general environment using small and wearable inertial sensors has yet to be reliably shown, although previous attempts have been made [8, 9].

In this paper we demonstrate how to locate and track a person in a building for which we have an accurate model, using an off-the-shelf wearable inertial system and a particle filter to tackle the traditional drift problems associated with inertial tracking. With this setup, we find that we are able to track a person throughout a $8725\,\mathrm{m}^2$ building to within $0.73\,\mathrm{m}$ 95% of the time, all without the user providing any explicit initialisation information. Since the system requires very little fixed infrastructure, the monetary cost is proportional to the number of users, rather than to the coverage area as is the case for traditional indoor location systems. We propose that such a system could be used to enable the deployment of location-aware applications in large buildings, where the installation of a high accuracy absolute location system is either too expensive or impractical.

The structure of this paper is as follows. We first review relevant literature and how a foot-mounted IMU might be used to provide a sequence of relative locations. We then discuss a computational model for the representation of buildings, and how such models are input to our particle filter, along with the relative locations, to derive absolute positions. We discuss how to gather initialisation information autonomously, and describe a WiFi-based scheme to reduce the computational overheads involved at the start of the localisation process. Finally, we evaluate our system experimentally, by quantitatively comparing it with an independent, high-accuracy tracking system installed in the building.

RELATED WORK

Localisation is a problem often encountered in mobile robots, where sensor-based localisation has been recognised as a key problem [7]. In the robot localisation problem a robot is placed at an unknown location in a known environment. By sensing the environment the robot must determine its absolute location. Robots used in localisation research typically sense their environment using a laser range-finder, which measures the distance to the nearest obstruction in the direction in which it is pointed.

Solutions to the robot localisation problem have been implemented using both grid [4] and particle-based [7] Bayesian filters. Both are able to track the multimodal distributions that often arise during localisation due to symmetry in the environment. Particle filters are preferred because they require significantly less computation and have a smaller memory footprint than comparable grid-based filters [7].

There are several differences between robot and pedestrian localisation as presented in this paper. Firstly, the robots used in existing literature have been unable to climb stairs. As a result, a 2-dimensional map of the environment has been sufficient. Secondly, the movement of a robot can be actively controlled. This is clearly not possible in a pedestrian localisation system. Thirdly, robot localisation is typically solved knowing both the relative movement of the robot *and* measurements obtained from a laser range finder. In contrast, we use only relative movement information (in the form of step events) to solve the pedestrian localisation problem.

Aside from their use in robot localisation, particle filters have also been applied in absolute location systems. Most notable are The Location Stack [10, 12] and Placelab [13], which use particle filters to update the location of a user based on measurements received from a variety of different sensor systems. This work differs to our approach because they deal with absolute measurements and do not use the environment to constrain the user's movement. In contrast we use relative location information and environmental constraints to locate and track a user.

TRACKING USING A FOOT MOUNTED IMU

An IMU contains three orthogonal rate-gyroscopes and accelerometers, which report angular velocity and acceleration respectively. In principle, it is possible to track the orientation of the IMU by integrating the angular velocity signals. This can then be used to resolve the acceleration samples into the global frame of reference, from which acceleration due to gravity is subtracted. The remaining acceleration can then be integrated twice to track the position of the IMU relative to a known starting point and heading [18].

Unfortunately the error, or 'drift' in the calculated position grows rapidly with time. The main cause of drift is small errors perturbing the gyroscope signals, which cause growing tilt errors in the tracked orientation. A small tilt error causes a component of acceleration due to gravity to be projected onto the globally horizontal axes. This residual is double integrated, causing an error in position which grows cubically in time in the short term [8]. The drift incurred by a high end MEMS IMU will typically exceed 100 metres after 1 minute of operation [19].

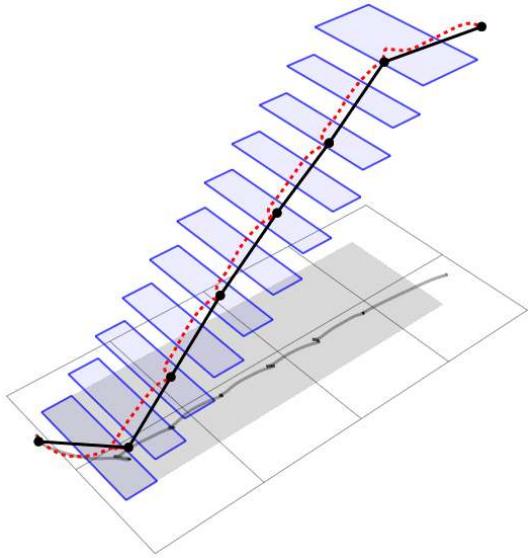

Figure 1. A foot mounted IMU trace (dashed red line) obtained by descending a flight of stairs, and the corresponding step events (solid black lines separated by dots) reported by the inertial navigation component. Grid size = 1 $\mathbf{m^2}$.

For foot-mounted IMUs the cubic-in-time drift problem can be reduced by detecting when the foot is in the stationary stance phase (i.e. in contact with the ground) during each gait cycle. Zero velocity updates (ZVUs) can be applied during this phase, in which the known direction of acceleration due to gravity is used to correct tilt errors which have accumulated during the previous step. The application of such constraints replaces the cubic-in-time error growth with an error accumulation that is linear in the number of steps [8]. The detection of stationary stance phases and the application of ZVUs is documented extensively in the literature [8, 3, 15, 9] and hence is not discussed further here.

In this paper we consider the inertial navigation component to be a black box. This component performs inertial navigation (with ZVUs), and segments the integrated trace into step events corresponding to strides taken by the pedestrian, as shown in Figure 1. The i^{th} step event is reported as a tuple

$$\boldsymbol{u}_i = (l, \delta z, \delta\theta) \tag{1}$$

in which l is the horizontal step length, δz is the change in height over the step, and $\delta\theta$ is the change in heading between the previous and current steps. The pedestrian localisation problem is then to use these (noisy) step events to determine and track the absolute position of the user in a known environment, as shown in Figure 2. Note that the initial position and orientation of the user are unknown.

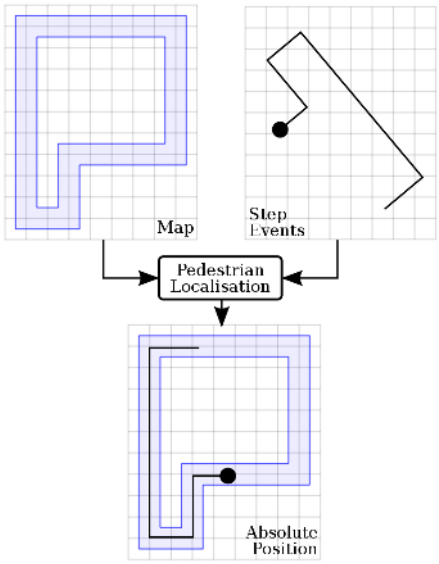

Figure 2. The pedestrian localisation problem: To determine the absolute location of a pedestrian given a path of step events describing their relative movement, and a map of the environment.

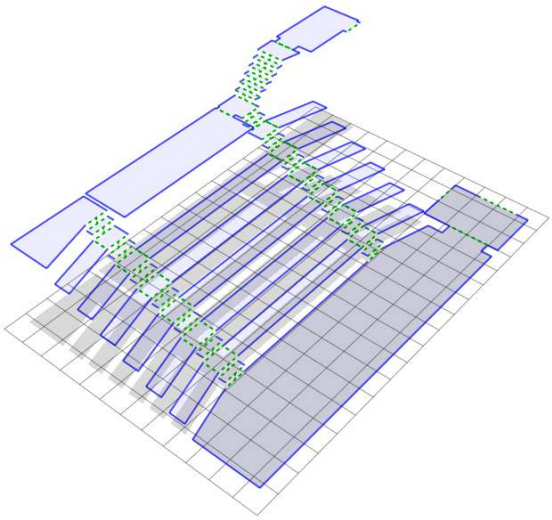

Figure 3. The 2.5D representation of a lecture theatre. The edges separating adjacent rows of seating are considered as walls (solid blue lines), whereas the edges between stairs in the aisle are connections (dashed green lines). Grid size = 1 $\mathbf{m^2}$.

2.5D BUILDING REPRESENTATION

There are many obstacles that limit the possible movement of a pedestrian within a building. In particular walls are impassable obstacles.

In order to enforce such constraints it is necessary to have a computer-readable plan of the building. Since it is reasonable to assume that a pedestrian's foot is constrained to lie on the floor during the stance phase of the gait cycle, a 2.5-dimensional description of the building (in which each object has a vertical position but no depth) is sufficient. Hence we define a map to be a collection of planar floor polygons. Each floor polygon corresponds to a surface in the building on which a pedestrian's foot may be grounded. Each edge of a floor polygon is either an impassable wall or a connection to the edge of another polygon. Connected edges must coexist in the (x,y) plane, however they may be separated in the vertical direction to allow the representation of stairs. It is possible to represent even complex rooms using this format, such as the lecture theatre shown in Figure 3.

The use of a 2.5D format avoids the additional complexity that would be required to construct and use a fully 3-dimensional map.

PARTICLE FILTERS

Bayesian filters probabilistically estimate the state of a dynamic system based on noisy measurements. A Bayesian filter represents its belief about a system at time t as a probability distribution over the state space

$$\mathrm{Bel}(\boldsymbol{s}_t) = p(\boldsymbol{s}_t|\boldsymbol{z}_1, ..., \boldsymbol{z}_t) \tag{2}$$

where $\boldsymbol{s}_t$ is a state and $\boldsymbol{z}_1, ..., \boldsymbol{z}_t$ are noisy measurements made up to (and including) time t.

Under the Markov assumption $\mathrm{Bel}(\boldsymbol{s}_t)$ can be calculated by updating $\mathrm{Bel}(\boldsymbol{s}_{t-1})$, allowing a Bayesian filter to track the state of a dynamic system through time. To do this the belief is first propagated according to a motion model; a probability distribution which defines the possible transitions from one state to the next. The propagated belief is then corrected using measurements of the system. Each measurement has a corresponding measurement model; a probability distribution defining the likelihood of observing the measurement given that the system is in a particular state at the same point in time.

To update the belief it is first propagated forward according to a motion model $p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, \boldsymbol{c}_t)$ to obtain the prior:

$$\mathrm{Bel}^-(\boldsymbol{s}_t) = \int p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, \boldsymbol{c}_t)\mathrm{Bel}(\boldsymbol{s}_{t-1})\, d\boldsymbol{s}_{t-1}. \tag{3}$$

where $\boldsymbol{c}_t$ is control information describing the transition from the previous to the current state (for pedestrian localisation step events can be used). The updated posterior is calculated from the prior as

$$\mathrm{Bel}(\boldsymbol{s}_t) = \alpha_t p(\boldsymbol{z}_t|\boldsymbol{s}_t)\mathrm{Bel}^-(\boldsymbol{s}_t) \tag{4}$$

where $p(\boldsymbol{z}_t|\boldsymbol{s}_t)$ is the measurement model corresponding to the measurement $\boldsymbol{z}_t$, and α_t is a normalisation factor.

Particle filters approximate the posterior distribution as a set of weighted samples ('particles')

$$\mathcal{S}_t = \langle \boldsymbol{s}_t^i, w_t^i \rangle \;\; i = 1, ..., n \tag{5}$$

where $\boldsymbol{s}_t^i$ is the state and w_t^i is the weight of the i^{th} particle. A particle filter update consists of generating the set $\mathcal{S}_t$ from the previous posterior $\mathcal{S}_{t-1}$. Each new particle $\langle \boldsymbol{s}_t^j, w_t^j \rangle$ is generated as follows [5]:

1. **Re-sampling:** Sample a state $\boldsymbol{s}_{t-1}^i$ from the prior $\mathcal{S}_t^-$ according to the distribution defined by the weights w_{t-1}^i.

2. **Propagation:** Sample $\boldsymbol{s}_t^j$ from the motion model distribution $p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}^i, \boldsymbol{u}_t)$, where $\boldsymbol{s}_{t-1}^i$ is from the re-sampling step.

3. **Correction:** Sample w_t^j from the measurement model distribution $p(\boldsymbol{z}_t|\boldsymbol{s}_t^j)$, where $\boldsymbol{z}_t$ is the measurement received at time t.

This procedure is repeated n times, where n is the number of particles in the new posterior. The value of n can be fixed or varied at each step according to an adaptive resampling scheme.

In our localisation framework each particle has a state $\boldsymbol{s}_t$ at time t

$$\boldsymbol{s}_t = (x_t, y_t, \theta_t, \mathrm{poly}_t) \tag{6}$$

where (x_t, y_t) is the horizontal position, θ_t is the heading and poly_t is the floor polygon to which the particle is currently constrained. Since the particle is defined to lie on the surface of poly_t, it is not necessary to explicitly store the vertical position of the particle in the state vector.

We now outline in detail the propagation, correction and resampling steps used by our localisation filter.

Particle propagation

The propagation step generates the state $\boldsymbol{s}_t$ of a new particle by sampling from the motion model distribution $p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, \boldsymbol{c}_t)$, where $\boldsymbol{s}_{t-1}$ is a previous state selected during resampling and $\boldsymbol{c}_t = \boldsymbol{u}_t$ is the step event for the interval $(t-1,t)$.

First, we perturb the step according to an uncertainty model, which describes uncertainty that has built up over the step due to noise perturbing the IMU measurements. We use a simplified model in which it is assumed that both the length and the change in heading of the step are perturbed by Gaussian random variables, drawn from $\mathcal{N}_{0,\sigma_l}$ and $\mathcal{N}_{0,\sigma_\theta}$ respectively. The deviations were chosen empirically to be $\sigma_l = 0.15\,\mathrm{m}$ and $\sigma_\theta = 0.6^\circ$. The perturbed step parameters are

$$l' = l + X \tag{7}$$
$$\delta\theta' = \delta\theta + Y \tag{8}$$

where X and Y are drawn from $\mathcal{N}_{0,\sigma_l}$ and $\mathcal{N}_{0,\sigma_\theta}$ respectively.

The new heading and position are then calculated from the previous state and the perturbed step parameters:

$$\theta_t = \theta_{t-1} + \delta\theta' \tag{9}$$
$$x_t = x_{t-1} + l' \cdot \cos\theta_t \tag{10}$$
$$y_t = y_{t-1} + l' \cdot \sin\theta_t \tag{11}$$

It is also necessary to determine the floor polygon poly_t to which the propgated particle is constrained. We initially assume that the floor polygon in which the particle resides is unchanged by the update:

$$\mathrm{poly}_t = \mathrm{poly}_{t-1} \tag{12}$$

In order for the particle to have exited this floor polygon, the step vector must intersect one of its edges in the (x, y) plane. Let $\boldsymbol{A} = (x_{t-1}, y_{t-1})$ and $\boldsymbol{B} = (x_t, y_t)$ be the start and end points of the step vector in the (x, y) plane. We find the first intersection between $\overrightarrow{\boldsymbol{AB}}$ and the edges of poly_t in the (x,y) plane, using it to update poly_t according to one of three cases:

1. No intersection point is found. The particle must still be constrained to the same polygon.

2. The first intersection $\boldsymbol{C}$ is with a wall. In this case the particle's weight should be set to equal 0 in the correction step, enforcing the constraint that walls are impassable.

3. The first intersection $\boldsymbol{C}$ is with an edge connecting to another polygon (given by $\mathrm{GetDstPoly}(\boldsymbol{C})$). In this case we update the current polygon

   $$\mathrm{poly}_t = \mathrm{GetDstPoly}(\boldsymbol{C}) \tag{13}$$

   Since a single step may span multiple connections, the intersection test is repeated between the remainder of the step vector ($\overrightarrow{\boldsymbol{CB}}$) and the updated polygon. This process continues recursively until one of the first two cases applies.

Particle correction

The correction step sets the weight w_t of a propagated particle. We use this step to enforce wall constraints. If a wall is intersected during the propagation step used to generate the state of the particle, then it is assigned a weight

$$w_t = 0 \tag{14}$$

If a wall is not intersected, the particle is assigned a weight based on the difference between the height change δz of the current step and the difference in height between the start and end floor polygons. The height change according to the map is given by

$$\delta z_{\mathrm{poly}} = \mathrm{Height}(\mathrm{poly}_t) - \mathrm{Height}(\mathrm{poly}_{t-1}) \tag{15}$$

We assign the particle a weight

$$w_t = \mathcal{N}_{0,\sigma_h}(|\delta z - \delta z_{poly}|) \tag{16}$$

where σ_h was chosen empirically as $0.1\,\mathrm{m}$. Here we are using the change in height reported in the current step as a measurement in the Bayesian framework. Particles whose change in height over the step closely matches the change in height reported in the step event are assigned stronger weights. This allows localisation to occur quickly when the user climbs or descends stairs. Particles which are not located on stairs will have a height change $\delta z_{\mathrm{poly}} = 0$ according to the map. This will not be close to the height change reported in the step event, causing the particles to be assigned small weights relative to particles that are located on stairs. This will make them less likely to be resampled, resulting in rapid convergence of $\mathrm{Bel}(\boldsymbol{s})$ to stair regions.

The combined propagation and correction algorithm is shown in Algorithm 1.

Algorithm 1 Update - Applies the propagation and correction algorithms to generate a particle at time t from a sampled state at time $(t-1)$.

procedure UPDATE(
$\boldsymbol{s}_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1}, \text{poly}_{t-1})$,
$\boldsymbol{u}_t = (l, \delta z, \delta\theta)$)
// Model step noise
$\delta\theta' \leftarrow \delta\theta + X$
$l' \leftarrow l + Y$
// Calculate the new xy-position
$\theta_t \leftarrow \theta_{t-1} + \delta\theta'$
$x_t \leftarrow x_{t-1} + l' \cdot \cos\theta_t$
$y_t \leftarrow y_{t-1} + l' \cdot \sin\theta_t$
// Initialisation for the intersection algorithm
$\text{poly}_t \leftarrow \text{poly}_{t-1}$
$\boldsymbol{A} \leftarrow (x_{t-1}, y_{t-1})$
$\boldsymbol{B} \leftarrow (x_t, y_t)$
done $\leftarrow$ false
kill $\leftarrow$ false
// Determine the new constraining polygon
while ¬done **do**
$\boldsymbol{C} \leftarrow \text{Intersect}(\overrightarrow{\boldsymbol{AB}}, \text{poly}_t)$
if $\boldsymbol{C} = \text{null}$ **then**
done $\leftarrow$ true
else if $\text{Type}(\boldsymbol{C}) = \text{Wall}$ **then**
// Note the wall intersection
kill $\leftarrow$ true
done $\leftarrow$ true
else if $\text{Type}(\boldsymbol{C}) = \text{Connection}$ **then**
$\text{poly}_t \leftarrow \text{GetDstPoly}(\boldsymbol{C})$
$\boldsymbol{A} \leftarrow \boldsymbol{C}$
end if
end while
// Correction step
if kill **then**
$w_t \leftarrow 0$
else
$\delta z_{\text{poly}} \leftarrow \text{Height}(\text{poly}_t) - \text{Height}(\text{poly}_{t-1})$
$w_t \leftarrow \mathcal{N}_{0,\sigma_h}(|\delta z - \delta z_{poly}|)$
end if
// Return the updated particle
return $\langle (x_t, y_t, \theta_t, \text{poly}_t), w_t \rangle$
end procedure

Re-sampling

The number of particles needed to represent $\text{Bel}(\boldsymbol{s})$ to a given level of accuracy depends on the complexity of the distribution, which can vary drastically over time. As a result it can be highly inefficient to use a fixed number of particles. This is particularly true for localisation problems, where the number of particles required to track an object after convergence is typically only a small fraction of the number required to adequately describe the distribution in the early stages of localisation [5].

Several schemes have been proposed for dynamically adapting the number of particles used to represent the distribution, such as likelihood-based adaptation [7], Kullback-Leibler divergence (KLD) sampling [5] and variants [17]. We use KLD-sampling in our framework since likelihood-based adaptation is not well suited for problems where $\text{Bel}(\boldsymbol{s})$ can be a multimodal distribution, as is often the case during indoor localisation due to symmetry in the environment [5].

The idea of KLD-sampling is to generate a number of particles at each step such that the approximation error introduced by using a sample-based representation of $\text{Bel}(\boldsymbol{x})$ remains below a specified threshold ϵ. KLD-sampling assumes that the sample-based representation of the propagated belief can be used as an estimate for the posterior [6], and that the true posterior can be represented by a discrete piecewise constant distribution consisting of a set of multidimensional bins. Since the propagation step in our framework uses control information (in the form of step events), the propagated belief is a reasonable estimate of the posterior. The second assumption requires that we specify a bin size, which was chosen empirically to be $1\,\text{m}^3$ in position and 10° in heading.

INITIALISATION AND LOCALISATION

To test our framework a hip-mounted ultra mobile PC (UMPC) was used to log data obtained from a foot-mounted Xsens Mtx IMU. The logs were then postprocessed on a desktop machine. In the future we envisage that the system will consist of only a foot-mounted component, containing the IMU, battery, and WiFi capability to offload the data for real-time processing.

Figure 4 shows an example application of our framework in our lab, a three storey building with a floor area of $8725\,\text{m}^2$ ($93915\,\text{sqft}$). Figure 4(a) depicts the route taken by the pedestrian and the steps generated by the inertial navigation component. Note that the trace has been manually aligned with the map to have the correct initial location and orientation, both of which are unknown to the localisation algorithm.

Figure 4(b) shows the initial collection of particles, drawn from the prior $\text{Bel}(\boldsymbol{s}_0)$. Since the initial location of the pedestrian is unknown, we use a uniform distribution over the entire floor surface. The initial heading is also unknown, hence the particle headings are distributed uniformly in all directions. Figure 4(c) shows the particles after the pedestrian has taken five steps. Figures 4(d) and 4(e) show the distribution just before and just after the pedestrian starts to descend a flight of stairs. Note how the pedestrian is quickly localised to stair regions due to the use of height change measurements in the correction step of the particle filter.

The example presented illustrates two problems commonly faced during localisation tasks: symmetry of the environment and scalability.

Environmental symmetry

Symmetry in the environment can delay or prevent convergence to a single cluster of particles. Rotational and translational symmetry cause multimodal distributions to arise during the localisation process. Such distributions consist of a

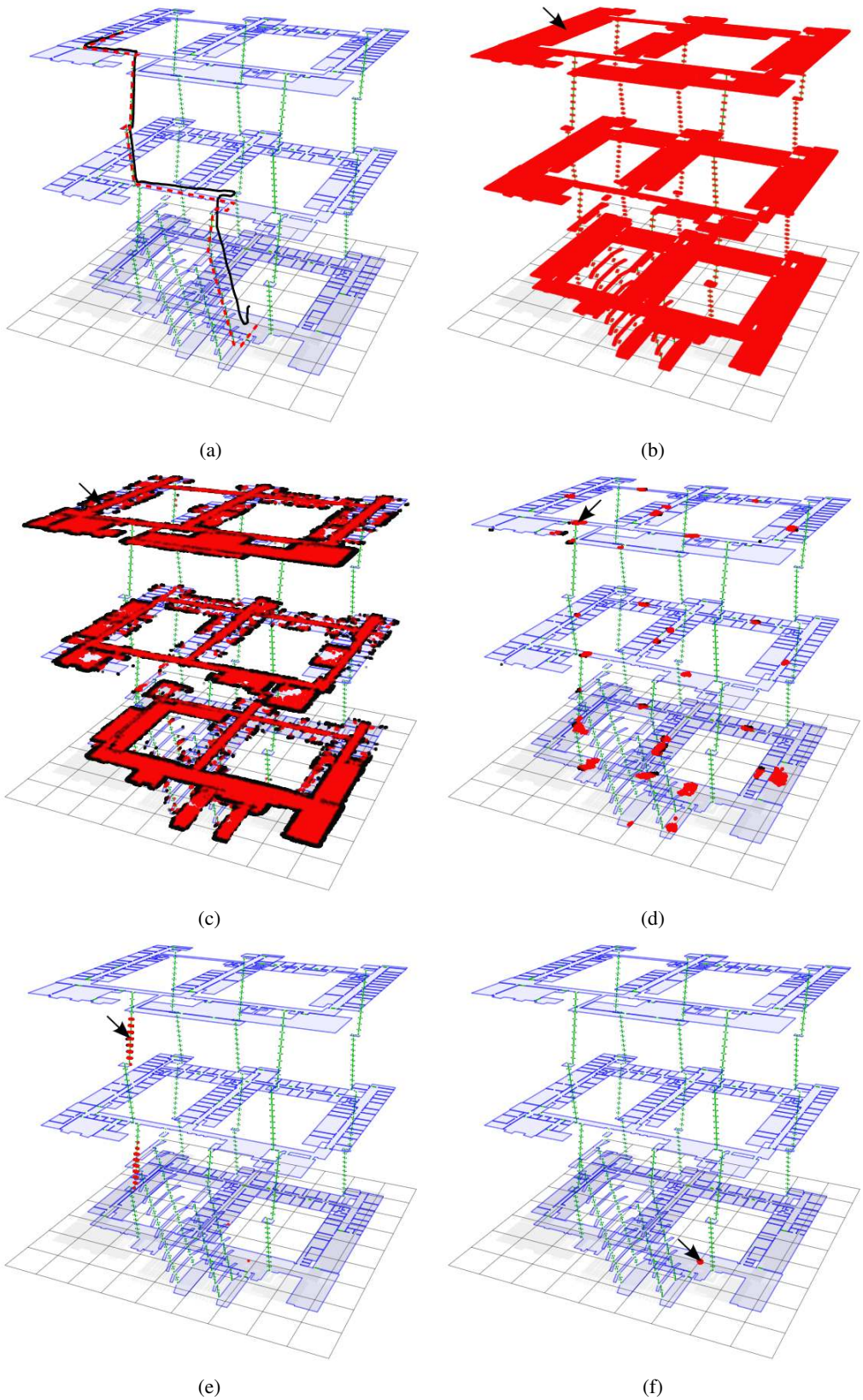

Figure 4. An example of localisation in a three storey building. (a) The route taken by the pedestrian (dashed red line) and a manually aligned overlay of the steps generated by the inertial navigation component (solid black line); (b) The prior distribution of particles; (c-f) The particle distribution at four points during localisation. Particles for which $w = 0$ are coloured black. An arrow indicates the actual position of the pedestrian in each figure. Grid size $10\,\mathrm{m}^2$. Diagrams are exploded 10x in the z-axis.

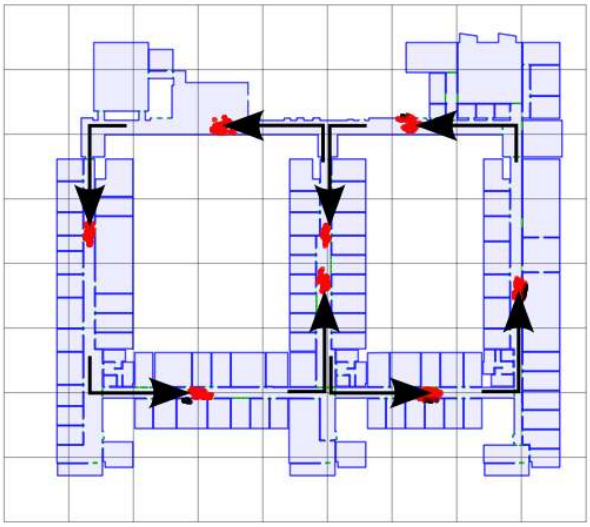

Figure 5. A multimodal distribution caused by symmetry in the environment. Each arrow indicates the path taken by a cluster to reach its current position. Grid size $10\,\mathrm{m}^2$.

set of distinct particle clusters, each of which represents a possible location. For example, the distribution in Figure 4(d) consists of multiple clusters which have arisen due to both translational and rotational symmetry. Figure 5 shows the multimodal distribution on the first floor in more detail.

Translational symmetry is a particular problem for buildings in which each floor has a similar layout. In such an environment translational symmetry in the vertical direction makes it difficult to localise the pedestrian to a single floor, as demonstrated in Figures 4(d) and 4(e).

Scalability

The time required to incorporate a new step event into the belief is $\mathcal{O}(n\log_2(n))$, where n is the number of particles sampled from the prior. This is due to a binary search algorithm used in the re-sampling step. Since the number of particles required to represent the uniform prior $\mathrm{Bel}(\boldsymbol{s}_0)$ is proportional to the floor area (A) of the building, it is clear that for a large enough building it will not be possible to perform localisation in real time. Our lab is an example of such a building, with a floor area of $8725\,\mathrm{m}^2$. Using KLD-sampling to draw particles from the uniform prior results in an average of 4530000 particles. Our current implementation (written in Java and run on a $2.6\,\mathrm{GHz}$ Linux machine) is only able to resample, propagate and correct 274000 particles during the average step duration[1] of $1.1\,\mathrm{seconds}$.

In order to perform real-time localisation in large buildings, it is necessary to obtain a better (i.e. more constrained) prior. By constraining the prior we reduce the number of particles required during the early stages of localisation, and also reduce the ambiguity that can arise due to symmetry in the environment. There are many ways in which the initial location and heading of a user can be constrained. We considered three options:

- A magnetometer can be used to obtain an initial heading for the pedestrian. If the initial heading were known to within 10° then the number of particles required to represent the prior would be reduced by a factor of 36 for a given floor area (although remaining $\mathcal{O}(A)$ complexity). An initial heading estimate can also resolve ambiguity due to rotational symmetry in the environment.

  Magnetometers work well in the absence of local magnetic disturbances. They can also be used in environments where there are local disturbances in the magnetic field, since it is possible to detect and adapt to the presence of such disturbances [16]. The Xsens IMU used for our trials contains a 3-axis magnetometer. However, strong magnetic components in the floor panels of our building render it useless when in close proximity ($< 30\,\mathrm{cm}$) to the floor, even when using Xsens' proprietary filter to adapt to local magnetic disturbances. An alternative would be to use a magnetometer mounted higher up the body, such as on the hip.

- An altimeter can be used to determine a range of floors, or floor (depending on accuracy) on which the pedestrian is located. This would make the number of particles required proportional to the largest area of a single floor. The use of an altimeter would also solve ambiguity due to vertical translational symmetry in the environment.

- Wireless LANs are deployed in many large buildings. Using the observed signal strength from WiFi access points at known positions, it is possible to estimate the position of a user in the environment. There have been many attempts to build location systems using WiFi access points [20, 14, 2]. The accuracy of such systems is limited by the fact that radio propagation depends heavily on the environment, which can change over time. For example, closing a door between an access point and the user can dramatically alter the received signal strength. It is however possible to obtain a rough estimate of the user's location. For example the RADAR system locates the user to within $5\,\mathrm{m}$ for 75% of the time [2].

  WiFi access points can be used to constrain the prior to a region of the building, solving the scalability problem. By constraining the initial location we also reduce the chance of ambiguities arising due to symmetry in the environment, since multiple candidate locations which can arise due to symmetry are usually well separated in space.

WIFI FOR APPROXIMATE LOCALISATION

WiFi-based location systems typically determine the position of the user via received signal strength indication (RSSI) measurements of multiple access points. Our system obtains this information by querying the WiFi hardware embedded within the hip-mounted UMPC used to log the IMU data. Each query returns a list of visible WiFi access points and corresponding RSSI measurements. We collect the measure-

[1] Note that this time actually corresponds to two steps taken by the user, because step events are only generated for one of the user's feet.

ments using the Placelab[2] framework.

State-of-the-art WiFi location systems use RSSI measurements to attempt to estimate the co-ordinate locations of devices, with the best results coming from complex probabilistic approaches. Here, however, we wish merely to constrain our prior, and thus have much less stringent requirements. Our goal is to determine a (comparatively gross) region of space within which we can be confident the user is located (more 'portion of building' than 'portion of room'). Our key requirement is that the user is actually located within the region determined by our WiFi system. To this end, we avoid attempting to compute accurate point locations; we find that the resultant algorithm is both less demanding computationally and more resilient to radio noise (since only gross regions are ever sought). Our algorithm has two phases; an offline phase during which a coarse radio map is constructed, and an online phase which returns a region of space within which the user is located.

Offline phase

Before starting our offline phase, we divide the 2.5D map into irregular cells, with each cell uniquely mapped to one room. Each cell is then subdivided until all cells have no edge length greater than $8\,\mathrm{m}$. For each cell c_i we manually build a set $\mathcal{V}_i$ of visible access points. At least 20 queries are made within each cell, with the user standing in a variety of different orientations and positions and the doors both open and closed. Every access point sighted by one or more of the queries inside c_i is added to $\mathcal{V}_i$.

For each access point ap_j we build the set of cells $\mathcal{A}_j$ from which it was visible

$$\mathcal{A}_j = \{c_i \,|\, \mathrm{ap}_j \in \mathcal{V}_i\}. \tag{17}$$

We then define the visible region of the access point to be the union of these cells, given by

$$R_{\mathrm{j}} = \bigcup_{c_i \in \mathcal{A}_j} c_i \tag{18}$$

Figure 6 shows the visible region obtained by this approach for a single access point in our building.

Online phase

At the start of a localisation we query the UMPC's WiFi hardware to obtain a list of visible access points and their corresponding RSSI measurements. We assume that any access point with an RSSI measurement greater than $-75\,\mathrm{dBm}$ would have been visible in at least one of the queries made during the offline phase within the user's cell. We use this information to derive a constrained prior as follows.

If the visible access points with RSSI measurements greater than $-75\,\mathrm{dBm}$ are $\{\mathrm{ap}_1, ..., \mathrm{ap}_j\}$, then the constrained prior $\mathrm{Bel}_{\mathrm{wifi}}(\boldsymbol{s}_0)$ is defined to be the uniform prior over the region

$$R_{\mathrm{prior}} = R_1 \cap ... \cap R_j \tag{19}$$

[2] http://www.placelab.org

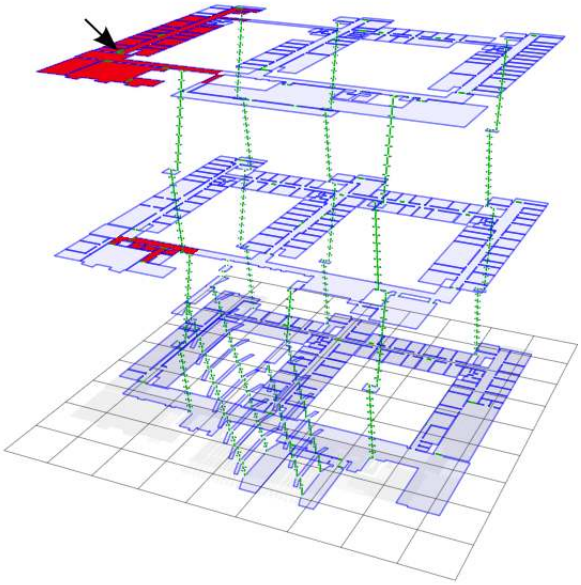

Figure 6. The visible region obtained by the offline phase for a WiFi access point. The arrow indicates the actual position of the access point.

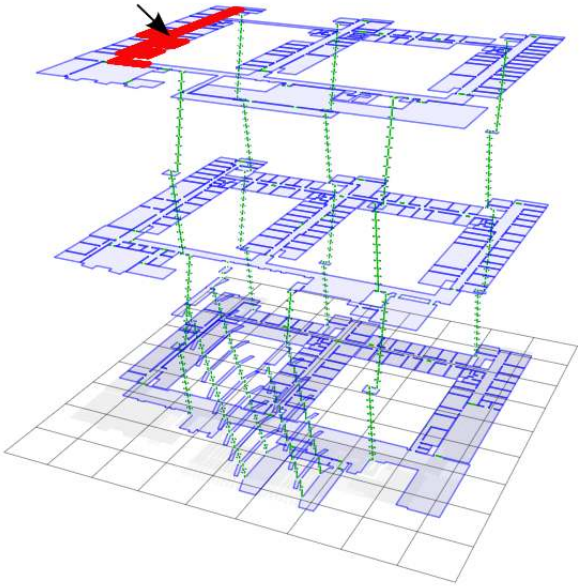

Figure 7. The constrained prior obtained using WiFi access points. The arrow indicates the actual position of the user.

For the example localisation shown in Figure 4 there were 4 access points visible at the initial location with RSSI measurements greater than $-75\,\mathrm{dBm}$. The prior obtained using the WiFi algorithm is shown in Figure 7. Using KLD-sampling to generate the prior results in an average of 136000 particles, less than $1/30^{\mathrm{th}}$ of the number required to represent the unconstrained prior over the whole building.

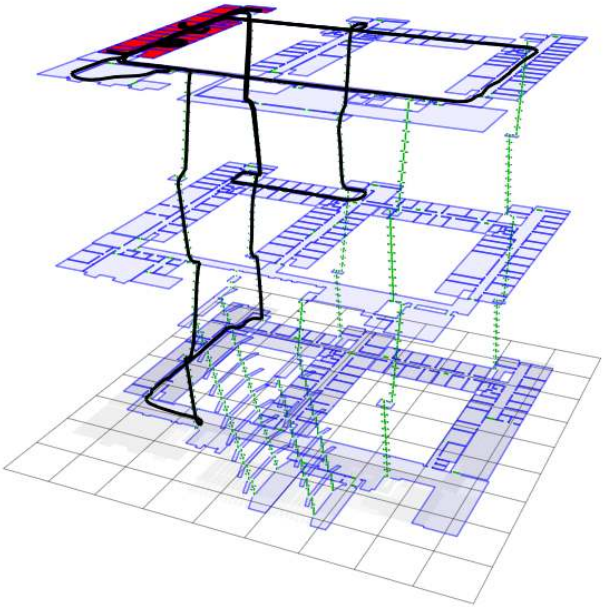

Figure 8. The path of positions (black line) extracted from the particle filter during tracking. The shaded region is the area of the building in which the Bat system is deployed.

TRACKING EVALUATION

When the particles have converged to form a single cluster, the problem solved by the filter is one of tracking rather than localisation. The cluster of particles tracks the position of the user, updating at the end of each stance phase when a new step event is reported by the inertial navigation component. The number of particles used to represent the user's position during tracking in our building is on average 170 and at most 500. Hence tracking is far less computationally demanding than localisation, and it is possible to track hundreds of users simultaneously on a single desktop machine.

During tracking we determine the position (X, Y) of the user by calculating the weighted average of the particles in the cluster

$$X_t = \frac{\sum_{i=1}^{n} w_t^i \cdot x_t^i}{\sum_{i=1}^{n} w_t^i} \quad (20)$$

$$Y_t = \frac{\sum_{i=1}^{n} w_t^i \cdot y_t^i}{\sum_{i=1}^{n} w_t^i} \quad (21)$$

where n is the number of particles in the cloud. To evaluate the tracking accuracy of our framework we compared it to the Bat system, an ultrasonic location system which is installed in one wing on the second floor of our building, as shown by the shaded region in Figure 8. To calculate the position of a Bat the system applies a multi-lateration algorithm to times of flight obtained for a pulse emitted by the Bat and received by multiple receivers installed at known locations in the ceiling. The calculated positions are accurate to within $3\,\mathrm{cm}$ 95% of the time [1].

To compare the two systems, a Bat was attached to the foot mounted IMU. The position of the Bat was queried during each stance phase detected by the inertial navigation component. When a position was returned during this phase it was matched to the corresponding position obtained from the cluster of particles for comparison.

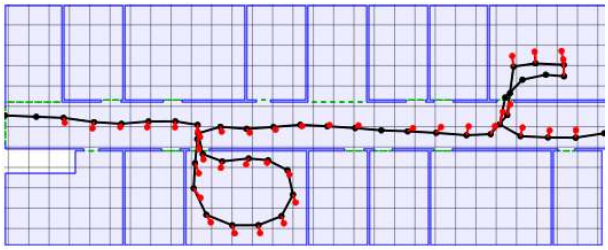

Figure 9. A single walkthrough of the area covered by the Bat system, showing the path of positions (black line) extracted from the particle filter. Each Bat position is shown as a (red) dot, connected by a (red) line to the corresponding particle filter position. Grid size = $1\,\mathrm{m}^2$.

In total six walkthroughs of the area in which the Bat system is deployed were made, as parts of a 16 minute continuous trace. Between each walkthrough the user walked a significant distance through other areas of the building, including different floors. Figure 8 shows the path of positions obtained from the particle filter after convergence to a single cluster of particles. Figure 9 shows a single walkthrough which occurs approximately 5 minutes into the trace.

In total 169 positions from the Bat system were matched to stance phases detected by the inertial navigation component. Since particle filters are probabilistic, the calculated positions vary slightly with each run. We ran the filter a total of 5 times, resulting in 845 pairs of matching positions. Since the filter constrains particles to lie on floor surfaces we compared the positions in the (x, y) plane only. Figure 10 shows the cumulative distribution of the Euclidian distances between corresponding Bat and particle filter positions. If we take the Bat positions as ground truth, the particle filter tracks the user with an accuracy of $0.50\,\mathrm{m}$ 75% of the time, and $0.73\,\mathrm{m}$ 95% of the time.

CONCLUSIONS AND FUTURE WORK

In this paper we have developed a tracking system that uses a foot-mounted inertial sensor, a model of a building, and a particle filter to track the wearer to well within $1\,\mathrm{m}$. In doing so, we have taken techniques developed for robot localisation and applied them in a situation where significantly less data is available; extended them to handle multiple floors and stairways; and adapted their constraints significantly to suit pedestrians. We have also presented a novel bootstapping algorithm, which uses WiFi signal strength to constrain the initial location.

We have quantitatively evaluated our system in greater depth than has been possible before, and demonstrated that inertial tracking around a building is at least feasible. Such a system could potentially be used to enable the deployment of location-aware applications in large buildings, where the installation of a high accuracy absolute location system is either too expensive or impractical.

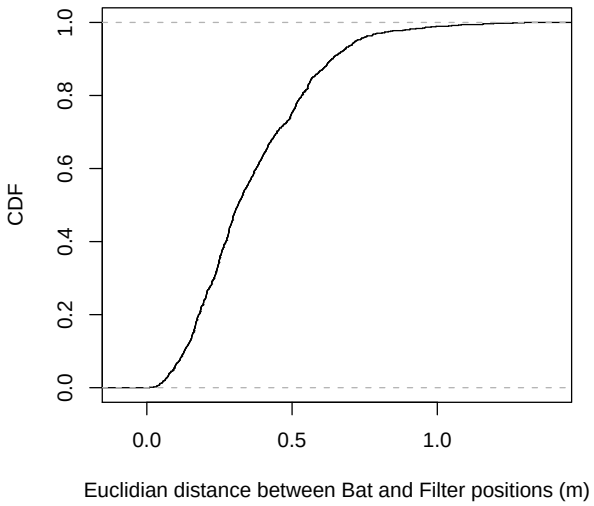

Figure 10. The cumulative distribution of the Euclidian distances between matched Bat and particle filter positions.

Our result of less than $0.73\,\mathrm{m}$ error 95% of the time is very promising, and we hope to improve it in the future in at least two ways:

- **Automated construction of a high resolution radio map.** By querying the visible access points whilst tracking with the system presented in this paper, it should be possible to construct and update a high resolution radio map online. This map could be used to further constrain future priors, and could potentially allow the system to adapt to changes in the WiFi infrastructure such as the removal or addition of access points.
- **Fusion with absolute location systems.** We intend to fuse our inertial-based localisation system with absolute positioning systems, including the Bat system and a WiFi based location system. This differs from our current use of WiFi access points, because positions calculated by the location system would be used throughout tracking rather than just at the initialisation stage.

Acknowledgements

The authors would like to thank Andy Hopper for his insightful comments. This work has been funded by EPSRC.

REFERENCES

1. M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. *Computer*, 34(8):50–56, 2001.
2. P. Bahl and V. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. *INFOCOM*, 2000.
3. S. Beauregard. Omnidirectional pedestrian navigation for first responders. In *WPNC*, 2007.
4. W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Position tracking with position probability grids. In *EUROBOT*, 1996.
5. D. Fox. KLD-sampling: Adaptive particle filters. In *NIPS 14*, 2002.
6. D. Fox. Adapting the sample size in particle filters through KLD-sampling. *IJRR*, 22(12):985–1004, 2003.
7. D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *AAAI*, 1999.
8. E. Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *Computer Graphics and Applications, IEEE*, 25(6):38–46, 2005.
9. S. Godha, G. Lachapelle, and M. E. Cannon. Integrated GPS/INS system for pedestrian navigation in a signal degraded environment. In *ION GNSS 2006*, 2006.
10. J. Hightower. *The Location Stack*. PhD thesis, Department of Computer Science & Engineering, University of Washington, Seattle, WA, 2004.
11. J. Hightower and G. Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, August 2001.
12. J. Hightower and G. Borriello. Particle filters for location estimation in ubiquitous computing: A case study. In *UbiComp*, 2004.
13. A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Place Lab: Device positioning using radio beacons in the wild. In *Pervasive*, 2005.
14. M. Ocana, L. M. Bergasa, M. A. Sotelo, J. Nuevo, and R. Flores. Indoor robot localization system using WiFi signal measure and minimizing calibration effort. In *ISIE*, 2005.
15. L. Ojeda and J. Borenstein. Non-GPS navigation for emergency responders. In *Sharing Solutions for Emergencies and Hazardous Environments*, 2006.
16. D. Roetenberg, H. Luinge, and P. Veltink. Inertial and magnetic sensing of human movement near ferromagnetic materials. In *ISMAR*, 2003.
17. A. Soto. Self adaptive particle filter. In *IJCAI*, 2005.
18. D. Titterton and J. Weston. *Strapdown Inertial Navigation Technology*. The American Institute of Aeronautics and Astronautics, 2nd edition, 2004.
19. O. Woodman. An introduction to inertial navigation. Technical Report 696, University of Cambridge, 2007.
20. M. Youssef and A. Agrawala. The Horus WLAN location determination system. In *MobiSys*, 2005.