Proceedings of Mobiquitous 2004, First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 4-14. August 22-26, 2004, Boston, MA, USA

LOCADIO: Inferring Motion and Location from Wi-Fi Signal Strengths

John Krumm and Eric Horvitz
Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA USA
{jckrumm, horvitz}@microsoft.com

Abstract

Context is a critical ingredient of ubiquitous computing. While it is possible to use specialized sensors and beacons to measure certain aspects of a user's context, we are interested in what we can infer from using the existing 802.11 wireless network infrastructure that already exists in many places. The context parameters we infer are the location of a client (with a median error of 1.5 meters) and an indicator of whether or not the client is in motion (with a classification accuracy of 87%). Our system, called LOCADIO, uses Wi-Fi signal strengths from existing access points measured on the client to infer both pieces of context. For motion, we measure the variance of the signal strength of the strongest access point as input to a simple two-state hidden Markov model (HMM) for smoothing transitions between the inferred states of "still" and "moving." For location, we exploit the fact that Wi-Fi signal strengths vary with location, and we use another HMM on a graph of location nodes whose transition probabilities are a function of the building's floor plan, expected pedestrian speeds, and our still/moving inference. Our probabilistic approach to inferring context gives a convenient way of balancing noisy measured data such as signal strengths against our a priori assumptions about a user's behavior.

1. Introduction

Context is a critical ingredient in many ubiquitous computing applications. Context -- for instance knowledge of a person's location, activity, or goals – can be used to tailor what information is presented, to present it in an appropriate way, and to trigger automatic behaviors that benefit the user. Elaborate context-measuring systems, such as MIT's highly instrumented living spaces and self-reporting devices[1], can be used to make precise context inferences, but are generally considered too expensive and invasive for wide deployment.

Location by itself is a useful component of context, because it enables reasoning about what a user is doing (*e.g.* in a meeting room), what a user is interested in (*e.g.* a painting at a museum), or what user interface devices are suitable (*e.g.* audio speakers in a given room). Beyond providing access to the current status of people, location information can support *presence forecasting* services that provide information about a user's future presence or availability[2]. Recent research has shown how to make higher level context inferences based on location. For instance, Sparacino[3] uses an indoor location system to classify museum visitors as either "greedy", "busy", or "selective", and then tailors extra museum content appropriately. Patterson *et al.*[4] use GPS tracks to classify a user's mode of transportation as either "bus", "foot", or "car", and to predict his or her mostly likely route.

Another important component of context is whether or not the user is in motion. A user in motion is likely not in a meeting and likely not to notice any messages put up on nearby displays. It is theoretically possible to compute motion by differentiating measured location over time, but location measurements are often too noisy for this to be reliable. Motion can be measured by wearing an accelerometer, such as in the SmartMoveX active badge[5], but this requires an extra device.

For measuring location, outdoor applications can rely on decoding timing signals from the Global Positioning Service (GPS) or GLONASS satellite navigation systems to obtain high-confidence location information. Unfortunately, no comparably ubiquitous means of measuring location is available for indoor applications. (See [6] for a review of location-measuring technologies for ubiquitous computing.) Although special-

ized systems such as active badges (*e.g.* broadband ultrasonic[7]) or radio frequency identification (RFID) tags (*e.g.* the IntelliBadge™[8]) can work well indoors, their installation costs can be prohibitive—and they require users to carry an extra device.

A promising alternative to relying on such specialized context-measurement systems is to infer what we can by measuring signal strengths received from a building's existing 802.11 (Wi-Fi) wireless infrastructure. Both Wi-Fi access points and mobile Wi-Fi clients are becoming more ubiquitous. Privacy is enhanced over systems that compute context on a central server, such as active badges, since our context inferences rely only on client-side data and computations.

Our system for inferring motion and location from Wi-Fi signal strengths is called LOCADIO (LOCation from rADIO). For inferring motion, it uses a simple variance measure as input to a two-state hidden Markov model (HMM) to classify a client as either still or moving. This was based on our observation that when a Wi-Fi receiver is moving, the signal strengths it receives are noisier than when it is not moving. Our goal for inferring location was to exploit *a priori* assumptions about a person's motion to enhance accuracy and minimize calibration effort. This motivation led us to explore a combination of several methods, including (1) the spatial interpolation of signal strengths from a sparsely sampled calibration set, (2) path constraints imposed by a building's interior structure (walls and doors), (3) integrating a consideration of human pedestrian speeds, and (4) using our previous inference about whether or not a client device is in motion. These elements are combined in a second HMM whose states are discrete (x, y) location nodes. The motion, path, and speed constraints are encoded as dynamic transition probabilities between the location nodes.

In the next section, we review related research. Then we discuss basic methods employed in LOCADIO, first for inferring motion and then location. We then describe experiments on testing the efficacy of the methods.

2. Related Work

Several teams have demonstrated how to identify the location of wireless clients indoors by measuring signal strengths from multiple 802.11 access points (APs). Matching signal strength signatures is the same basic technique used by all location-from-802.11 techniques, including the first one, called RADAR, developed by Bahl and Padmanabhan[9]. Using a manually calibrated table of signal strengths, their nearest neighbor algorithm gave a median spatial error of 2.94 meters. This error was reduced to 2.37 meters using a Viterbi-like algorithm in follow-on work [10]. As part of their research, Bahl and Padmanabhan also precomputed signal strength signatures using a model of radio propagation and a floor plan of the building. This reduced calibration effort at the expense of increasing their median location error to 4.3 meters.

Perhaps the most accurate 802.11 location system to date is described by Ladd *et al*.[11]. Their system used Bayesian reasoning and a hidden Markov model (HMM). They took into account not only signal strengths, but also the probability of seeing an access point from a given location. LOCADIO does this as well. Like other work, it was based on a manual calibration. Their system explicitly modeled orientation and achieved a median spatial error of about one meter using calibration samples taken approximately every 1.5 meters (five feet) in hallways. LOCADIO's added sophistication includes an interpolation technique to allow for sparse calibration data, an explicit probabilistic model of pedestrian speeds, transition probabilities that reflect the building's floor plan, and an independent inference on whether or not the client machine is in motion.

UCSD's ActiveCampus[14] project uses 802.11 to compute the location of wireless PocketPCs both indoors and outdoors. Instead of manual calibration, they use a formula that approximates the distance to an AP as a function of signal strength. Using a hillclimbing algorithm, their system computes location to an accuracy of about 10 meters (35 feet) using signal strengths from multiple APs.

Previous work on inferring a user's motion state has normally been based on time-stamped location inferences and a numerical derivative to estimate velocity. For instance, the MIT Activity Zones project[15] used camera tracking to measure (x, y) and then differentiated to compute instantaneous velocity. Patterson *et al*.[4] take their velocity readings directly from their GPS sensor, which presumably differentiates as well. We know of no other work that uses radio signal strength to directly infer motion of a client.

3. Inference of "Still" *versus* "Moving"

Motion is one important part of a user's context. We conjectured that it might be possible to classify a user as either still or moving based on Wi-Fi signal strength features. Such a conjecture was supported by our qualitative observation that signal strengths from

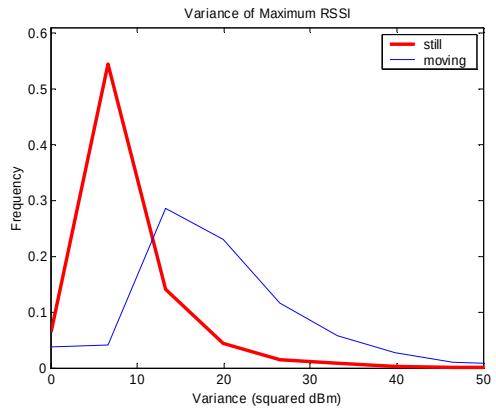


Figure 1: Normalized histograms of the variance of signal strength for a nonmoving and moving device. In general, moving devices show more variance.

APs appear to jump around more vigorously when the device is in motion than when it is still. Our approach is to classify a user as either still or moving based on the variance of a temporally short history of signal strengths from the currently strongest access point. We found that this classification transitioned too often between the two states, so we smoothed the classifications over time with a two-state HMM. The remainder of this section explains our approach and our accuracy assessment based on an experiment in our building.

3.1. Unsmoothed State Probabilities

We capture the "jumpiness" of a time series of signals quantitatively by computing a temporally windowed, running sample variance of the received signal strength of the strongest AP at the given time. That is, at any given time, we first find the AP with the strongest signal and then compute the variance of that AP's signal over a short interval ending at the given time.

For training we collected a set of labeled signal strengths by alternately walking around and stopping within an office building over a 30 minute period while recording signal strengths on our wirelessly networked laptop PC. As we walked and stopped, we manually indicated to our data-logging program whether we were moving or still. The variances were computed with a 20-second window, which translates to about 63 readings per AP at our signal strength sampling rate of 3.16 Hz.

The normalized histograms of the variances for the still and moving cases are shown in Figure 1. Using $\sigma^2_{\max}$ to represent the windowed variance of the currently strongest AP, we take the normalized histograms to represent the conditional probability distributions $p(\sigma^2_{\max}|still)$ and $p(\sigma^2_{\max}|moving)$. Given a value of $\sigma^2_{\max}$, we want to estimate the probability of moving, $p(moving|\sigma^2_{\max})$, and the probability of being still, $p(still|\sigma^2_{\max}) = 1 - p(moving|\sigma^2_{\max})$. By using Bayes rule, we know that the posterior probability of the client moving is

$$p(moving|\sigma^2_{\max}) = \frac{p(\sigma^2_{\max}|moving)p(moving)}{p(\sigma^2_{\max}|moving)p(moving) + p(\sigma^2_{\max}|still)p(still)} \quad (1)$$

Here $p(still)$ and $p(moving)$ are the *a priori* probabilities of the dynamic state of the device. In lieu of any other information about the priors, we set them equal at 0.5. The classification rule was then simply based on the state-conditional probabilities, which are the same as the normalized histograms:

$$p(\sigma^2_{\max}|moving) \underset{\text{decide still}}{\overset{\text{decide moving}}{>}} p(\sigma^2_{\max}|still) \quad (2)$$

Using the histograms from Figure 1 on a set of 3200 test readings taken several days after the training data, we were correctly able to classify approximately 85% of the data into the "still" and "moving" categories.

3.2. Smoothed Probabilities

The top of Figure 2 shows the ground truth state for our still *vs.* moving test data as a function of time. Shown below is a plot of the *a posteriori* $p(moving|\sigma^2_{\max})$ from Equation (1). It is clear that the *a posteriori* probability jumps from high to low too often given the process that is being modeled, so we seek to smooth the *a posteriori* probabilities by imposing explicit transition probabilities governing the two states.

Instead of simply trying to estimate the probability of a state q_T at time T from a single feature $\sigma^2_{\max,T}$ at that time, we will find the most likely sequence of states $Q = q_1, q_2, \mathsf{K}, q_T$ from a sequence of observations $O = \sigma^2_{\max,1}, \sigma^2_{\max,2}, \mathsf{K}, \sigma^2_{\max,T}$. In our case there are only

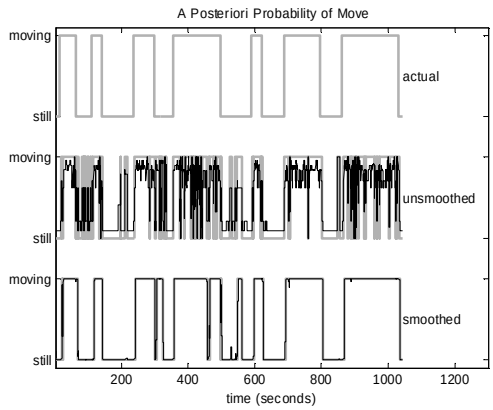


Figure 2: Results from our attempt to infer the probability that a device is moving based on the variance of signal strengths. The middle plot shows the unsmoothed probabilities, which jump around too much compared to the ground truth at the top. The bottom plot shows the probabilities smoothed with an HMM, which come much closer to ground truth.

two possible states, "still" and "moving," *i.e.* $q_t \in \{S, M\}$. For simplicity, we will use the first order Markov assumption to govern the transition between states, which says that the probability of the current state is independent of all but the most recent state, so that $P(q_{t+1} = j | q_t = i) = a_{ij}$, where a_{ij} is a transition probability and $i, j \in \{S, M\}$.

We can estimate the transition probabilities from assumptions about human behavior. We make a new inference after every signal strength sample, which occur with a frequency of $r = 3.16$ Hz, so there will be rs inferences in a period of s seconds. If we guess that a person will make m moves over a period of s seconds, then the probability of a move occurring between two inferences is $m/(rs)$. If we assume that each still-to-move transition is eventually accompanied by a move-to-still transition, then we have

$$\begin{aligned} a_{SM} &= a_{MS} = \min(m/(rs), 1) \\ a_{SS} &= 1 - a_{SM} \\ a_{MM} &= 1 - a_{MS} \end{aligned} \quad (3)$$

The $\min(\,)$ function keeps the transition probability within range. The equations for a_{SS} and a_{MM} come from the constraint that $a_{SS} + a_{SM} = a_{MM} + a_{MS} = 1$.

For a typical office worker, we guess that there are $m = 10$ moves in one eight-hour day, giving $s = 28{,}800$ seconds. At our rssi sampling rate of $r = 3.16$ Hz, we have

$$\begin{aligned} a_{SM} &= a_{MS} = 0.00011 \\ a_{SS} &= a_{MM} = 0.99989 \end{aligned} \quad (4)$$

We note that m could be tuned to whatever value is typical for a given user in a given context.

Another element of the Markov model is the initial probabilities of being in the still or moving states, π_S and π_M, respectively. For lack of any other information, we set them both to 0.5.

Because we cannot directly observe the states, we have a hidden Markov model. What we observe at each sample time t is $\sigma^2_{\max,t}$, which is probabilistically connected to the actual state through $p(\sigma^2_{\max,t} | q_t = still)$ and $p(\sigma^2_{\max,t} | q_t = move)$.

We now have all the elements necessary for an HMM: states, transition probabilities, initial state probabilities, and observation probabilities. Following Rabiner's tutorial[16], we use the Viterbi algorithm to compute the *a posteriori* state probabilities $P(q_T = still | O)$ and $P(q_T = moving | O)$ at the current time T conditioned on the sequence of signal strength variances $O = \sigma^2_{\max,1}, \sigma^2_{\max,2}, \mathsf{K}, \sigma^2_{\max,T}$.

The overall effect of using the HMM is that the transition probabilities tend to make the system more reluctant to change states due to slight or brief changes in the state-conditional probability densities. Using the transition probabilities computed above, we computed $P(q_T = moving | O)$ for each sample in our 3200-point test data set. The result is plotted at the bottom of Figure 2. This shows how using transition probabilities and a sense of past state make the state probabilities much less jumpy. The classification error rate drops from 15.5% to 12.6% by using the HMM smoothing. While the gain in classification accuracy is small, the real gain comes in the reduction of falsely reported state transitions. There were 14 actual transitions in the test set. Unsmoothed classification reports 172 transitions (158 too many), and smoothed classification reports 24 transitions (only 10 too many).

This moving *vs.* still indicator is a suitable compo-

nent of context for a mobile user, and it can be easily computed without having to first compute the user's locations over time, which takes considerably more work indoors. Our motion inference is also an important part of our location inference algorithm, which we discuss next.

4. Overview of Location Inference

The remainder of this paper discusses how we infer an (x, y) location from Wi-Fi signal strengths. Our method starts with a relatively sparse set of discrete (x, y) calibration nodes at which we have taken signal strengths readings. Representing these readings as probability distributions, we interpolate them into a denser set of location nodes on which we base our location inferences. These location nodes and their associated Wi-Fi pdfs are processed through an HMM (different from the one used for motion). The transition probabilities between the location nodes are a function of our still *vs.* moving inference, expected pedestrian speeds, and the building's floor plan. These elements are detailed in Sections 5-8.

5. Paths and Constraints

In order to facilitate path constraints imposed by the building's structure (*e.g.* walls and doors), LOCADIO uses a graph of discrete location nodes, as shown in Figure 3(a). The graph's edge weights (connections between the nodes) are the distances between the nodes. The result of the Viterbi algorithm used to infer location is a number attached to each node giving the probability that the device is at that location.

To place the nodes, we first manually draw a set of tracks through the building, as shown in Figure 3(b). These tracks represent the feasible walking paths through the building. We developed a drawing program that displays a bitmap of the building's floor plan as the background. The bitmap for our home building came from our institution's electronic database of floor plans, but this map could just as easily have come from a scanned blueprint. We compute the geometric transformation matrix between pixels and floor coordinates with simple least squares. Once all the lines have been drawn, the program converts the lines to nodes by taking each end point as a node and distributing nodes along the lines at a specified spacing. For our experiments, this spacing was one meter, making a total of 317 location nodes.

The graph is a fully connected, bi-directional graph so that every node is connected to every other node. The edges shown in Figure 3(a) are only the edges between adjacent nodes, and their edge weights are just the Euclidian distance between the nodes. For non-adjacent nodes, the edge weight is the shortest path distance through a sequence of adjacent nodes. We compute the shortest paths using Dijkstra's shortest path algorithm and store all the distances for use by the HMM. The shortest path distances embody the path constraints imposed by the building's structure. Formally, the distance between nodes i and j is called d_{ij}. We use these distances later to compute realistic transition probabilities between all nodes in the graph.

6. Transition Probabilities for Location

One of the essential ingredients of an HMM is the transition probabilities between states. In our case, the states are (x, y) location nodes on the floor, and the transition probabilities govern the probability of transitioning between any two location nodes. Qualitatively, we want the transition probabilities to nearby nodes to be larger than that to far away nodes. To quantify this notion, we use the shortest path distances described in Section 5 along with a probability distribution of human pedestrian speeds. For a more accurate speed distribution, we use the HMM-smoothed estimate of $p\left(moving \middle| \sigma^2_{\max}\right)$ from Section 3.

6.1. Speed Between Nodes

In this section we derive a probability distribution of human pedestrian speeds. In an office building, people mostly walk to get from place to place. We can approximate the distribution of walking speeds using a study of human walking speeds by John J. Fruin[18] (p. 40). We will call this distribution of walking speeds $P\left(walking\ speed \middle| moving\right)$. Further, people sometimes shuffle slowly from place to place, and they sometimes sprint. We model this behavior with a uniform distribution of speeds going from zero to a maximum of 10.22 meters/second[1], calling it $P\left(other\ speed \middle| moving\right)$. We assume that when a person is moving, he/she spends a

[1] As of 14 September 2002, the 100 meter sprint world record was 9.78 seconds set by American Tim Montgomery in Paris, for an average speed of 10.22 meters/second. We believe he was not carrying an 802.11 device at the time.

fraction ω walking and the rest of the time at some other speed. Given a person is moving, his/her speed distribution is then

$$P(s|moving) = \omega P(walking\ speed|moving) + (1-\omega)P(other\ speed|moving) \quad (5)$$

Here s represents speed in meters/second, and we assumed that ω is 0.9 in lieu of any hard data on how often people in buildings move at speeds other than walking speed.

The unconditional $P(s)$ takes into account the probability that the person is either moving or still, which comes from the dynamic inference from Section 3. Abbreviating these as $P(moving)$ and $P(still)$, we have

$$P(s) = P(s|moving)P(moving) + P(s|still)P(still) \quad (6)$$

where $P(s|still) = \delta(0)$, because a person's walking speed is zero when still. Here $\delta(x)$ is the Dirac delta function. This gives us a probability distribution of human pedestrian speeds based on whether or not we think the person is moving, and if so, also based on the distribution of walking speeds and maximum possible running speed.

6.2. Transition Probabilities

The transition probability between two location nodes is proportional to the probability of a human traveling at a speed necessary to traverse the distance between the nodes. If a device has moved from node i to node j, its speed had to be rd_{ij}, where r is the signal strength sampling rate (3.16 Hz in our case) and d_{ij} is the shortest path distance between the two nodes as explained above. The probability of observing this speed is $p_{ij} = P(s = rd_{ij})$. These probabilities must be normalized so that all transition probabilities emanating from a node sum to one. Thus the transition probability is

$$a_{ij} = p_{ij} \Big/ \sum_{j=1}^{N_l} p_{ij} \quad (7)$$

where $N_l = 317$ is the number of location nodes. These probabilities encapsulate what we know about the building's layout and about the speed of the device.

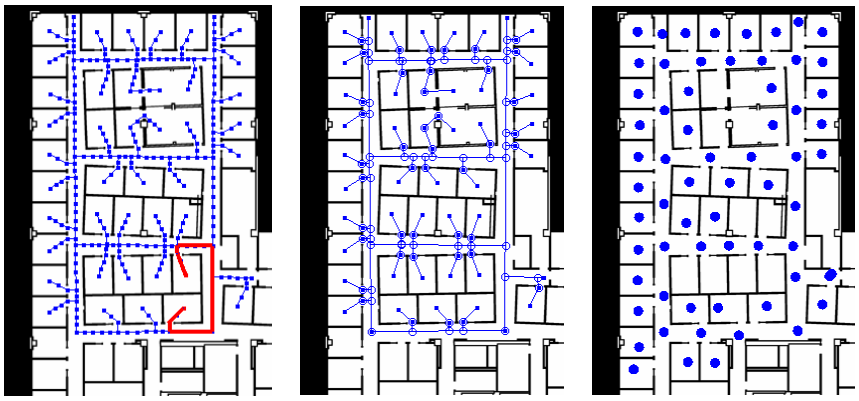
7. Signal Strength Observation Likelihoods

In inferring a location, the device's signal strengths are compared against signal strength probability distributions seen previously during calibration at different locations in the building. These previously seen signal strength distributions are estimated based on data from physically carrying the device to a set of known calibration nodes. A straightforward implementation would have us visit all $N = 317$ location nodes for calibration. Since we spend about 60 seconds at each calibration point, calibrating at each location node would be prohibitive. Instead, we take calibration readings at a much smaller number of locations (63 in our case), and use these to interpolate at the location nodes. This means we only had to calibrate at about 20% of the number of points used in the graph of location nodes. This section describes how we gather signal strengths in the building and how we interpolate the signal strength probability distributions to all location nodes.

7.1. Gathering Signal Strength Distributions

We gathered 802.11 signal strength distributions by carrying our wirelessly connected laptop computer to different locations in our building. These 63 locations are shown in Figure 3(c). Running on the laptop was our program for recording both locations and signal strengths. This program allows the user to indicate his or her location by clicking on the building map while simultaneously recording signal strengths from all visible 802.11 access points. The map makes it easy to indicate the device's approximate location for calibration. The calibration locations were only as accurate as we could click our positions on the map, but we felt this was a necessary compromise to reduce the calibration effort to a realistic level for larger deployments.

At each calibration position we took signal strength readings for 60 seconds while slowly spinning around in place. The spinning was to factor out the effect of orientation. This is in contrast to the system of Ladd *et al.*[11] who modeled and recorded orientation explicitly. With this data we constructed discrete probability distributions describing, for each calibration point, the probability of seeing a given access point and the probability distribution of signal strengths from that access point. In mathematical terms, the calibration points are $(x_j^{(c)}, y_j^{(c)}) = \underline{x}_j^{(c)}$, $j = 1 \mathsf{K}\ N_c$, and the building's access points are designated $AP_i, i = 1 \mathsf{K}\ N_{AP}$. Here the $^{(c)}$ superscript indicates a calibration point. The probability



(a) (b) (c)

Figure 3: Location Nodes. (a) Dense set of location nodes. The red path shows the shortest path distance between the centers of two offices. (b) Walking paths that were drawn by hand. (c) Sparse set of calibration nodes at which we took signal strength readings.

of detecting access point AP_i from calibration location $\underline{x}_j^{(c)}$ is $p\left(AP_i \middle| \underline{x}_j^{(c)}\right)$. We estimated this probability simply by the ratio of the number of times the access point was detected to the number of times we scanned for all access points during calibration at the given location.

If we did see an access point from a given location, then we also constructed a normalized histogram of signal strengths to represent $p\left(s_k \le s < s_{k+1} \middle| AP_i, \underline{x}_j^{(c)}\right)$. Here s is the signal strength and the s_k are the edges of the histogram bins. For our implementation, we had s_k range from -120 dBm to 0 dBm in 30 steps. (dBm stands for decibel milliwatts, and is the usual unit for 802.11 signal strength.) The overall result of the calibration captured both how often a given access point could be seen from a given location, and, if it could be seen, the distribution of signal strengths. These probabilities embody the signal strength signatures that we use to infer a device's location from the signal strengths it observes.

7.2. Interpolating Signal Strength Distributions

The 63 calibration points were relatively widely spaced, with an average of 2.64 meters to each point's nearest neighbor. We wanted to achieve higher spatial resolution with a set of location nodes spaced more densely than the calibration nodes. As shown in Figure 3, the location nodes (Figure 3(a)) are much more dense than the calibration nodes (Figure 3(c)). In order to infer location over the dense set of location nodes, we need to have signal strength signatures at each of the location nodes. This means we need to interpolate the probability distributions at the sparse set of calibration nodes into the denser set of location nodes. The values that we actually interpolated from were all the discrete probabilities that comprise the access point detection probabilities and the signal strength probabilities at the calibration nodes.

We perform this interpolation using normalized radial basis functions (rbfs), a common choice for such tasks. The rbf formulation makes a weighted sum of a set of 2D basis functions centered on the calibration points $\underline{x}_j^{(c)}$, $j = 1 \mathrm{K}\, N_c$, to produce the interpolant $d(\underline{x})$ for the chosen point $\underline{x}$:

$$d(\underline{x}) = \sum_{j=1}^{N_c} \left(\frac{\beta_j K\left(\left\| \underline{x} - \underline{x}_j^{(c)} \right\|\right)}{\sum_{l=1}^{N_c} K\left(\left\| \underline{x} - \underline{x}_l^{(c)} \right\|\right)} \right) \qquad (8)$$

For the kernel function $K(r)$ we chose $K(r) = \exp\left(-r^2/\sigma^2\right)$ each centered on a calibration point $\underline{x}^{(c)}$. After some experimentation, we settled on $\sigma = 1.0$ as a parameter that produced good results. The weights β_j were computed with standard least squares fitting to the calibration points. We computed a separate set of weights for each bin of each probability distribution.

The interpolated values formed the access point detection probabilities $p\left(AP_i \middle| \underline{x}_j^{(l)}\right)$ and signal strength probabilities $p\left(s_k \le s < s_{k+1} \middle| AP_i, \underline{x}_j^{(l)}\right)$ at the location nodes $\underline{x}_j^{(l)}$, $j = 1 \mathrm{K}\, N^{(l)}$, thus going from probabilities at a relatively sparse set of calibration points to estimated probabilities at denser set of location nodes. The normalized rbf is not guaranteed to produce probabilities in the range [0,1] nor probability distributions that integrate to one. In practice it came close, however, requiring only slight clamping and normalizing to restore the

proper range.

After completing this procedure, we had the necessary state-conditional probabilities at each node in the dense set of location nodes interpolated from the sparse set of calibration nodes.

8. Inferring Location using an HMM

Section 3.2 summarized the basic ingredients of an HMM: states, initial state probabilities, transition probabilities, and observation probabilities. The states of the HMM for location are the location nodes $\underline{x}_i^{(l)}, i = 1 \mathsf{K}\, N_l$, which we produced with our drawing program. With no other data about where a device might be located, the initial state probabilities $\pi_i, i = 1 \mathsf{K}\, N_l$ are uniformly distributed over the location nodes, *i.e.* $\pi_i = 1/N_l$. The transition probabilities are described in Section 4, and they are sensitive to the building's layout, expected pedestrian speeds, and our inference on whether or not the device is moving. The observation probabilities come from the interpolated probabilities described in Section 7.

For inferring location at time T the device scans for signal strengths from all access points. One result of this scan is an indicator vector $\underline{I}_T$ with one boolean element for each of the N_{AP} access points indicating whether or not the access point was detected. The other result is a vector of signal strengths $\underline{s}_T$ that gives the signal strength for each detected access point. Corresponding elements in these two vectors correspond to the same access point. If the access point was not detected, then the signal strength value for that access point can be any value, because it is not used. The probability of seeing this scan at location $\underline{x}_i^{(l)}$ is

$$P\left(\underline{I}_T, \underline{s}_T \middle| \underline{x}_i^{(l)}\right) = \prod_{j=1}^{N_{AP}} \begin{cases} p\left(AP_j \middle| \underline{x}_i^{(l)}\right) p\left(s_{Tj} \middle| AP_j, \underline{x}_i^{(l)}\right) & \text{if} \quad I_{Tj} = \text{true} \\ 1 - p\left(AP_j \middle| \underline{x}_i^{(l)}\right) & \text{if} \quad I_{Tj} = \text{false} \end{cases} \tag{9}$$

Here I_{Tj} means the j^{th} element of $\underline{I}_T$, and s_{Tj} means the j^{th} element of $\underline{s}_T$. Each multiplicand in this product represents one AP, implying that the scan result for each access point is independent of the other access points. If the j^{th} access point was seen ($\underline{I}_{Tj} = \text{true}$), then the multiplicand represents the probability of seeing this access point at the observed signal strength s_{Tj}. If the j^{th} access point was not seen ($\underline{I}_{Tj} = \text{false}$), then the multiplicand represents the probability of not seeing this access point.

These HMM elements are combined with the Viterbi algorithm as described in Rabiner's tutorial paper[16] to produce a set of state probabilities over the location nodes, *i.e.* $p_T\left(\underline{x}_i^{(l)}\right)$. For the final location estimate, we take the expected value of the location:

$$\underline{x}_T = \frac{\sum_{i=1}^{N_l} p_T\left(\underline{x}_i^{(l)}\right) \underline{x}_i^{(l)}}{\sum_{i=1}^{N_l} p_T\left(\underline{x}_i^{(l)}\right)} \tag{10}$$

9. Results of Experiments

We tested our system in our home building. Our test data came from 10 short walks through the halls and into some offices in our test area. The 10 walks amounted to 4586 positions and associated 802.11 scans. The average length of each walk was 2 minutes 25 seconds. Determining ground truth location while walking is a problem we solved by walking in straight segments and clicking on our location on the floor map at the end of every segment. Assuming a constant walking speed along segments, we linearly interpolated ground truth positions along each segment. The median location error for all the tests was 1.53 meters. Figure 4 shows a histogram of the error amounts and a cumulative error distribution.

It is difficult to fairly compare different Wi-Fi location techniques without testing them in the same physical environment with the same data. This is because different buildings vary in their placement and density of access points and in their physical construction, which affects radio propagation. Likewise we expect such systems to be sensitive to the amount and exact placement of calibration points. Nevertheless, we can get an inkling of how LOCADIO compares by looking at its performance relative to two other Wi-Fi location techniques based on their reported performance, both in terms of accuracy and calibration effort.

Bahl and Padmanabhan's RADAR system covered the hallway outside about 54 offices with 70 calibration points, about 1.3 points per office. In our LOCADIO experiments, we used 27 calibration points in the hallway outside 44 offices, about 0.6 points per office, a reduction of over 50% with a simultaneous reduction in median error of about 35% (from 2.37 meters to 1.53 meters). (For this analysis we did not count the calibration points we took in offices, as RADAR's testing was only done in hallways.)

Ladd *et al.* spaced their hallway calibration points about 1.5 meters (5 feet) apart, while our hallway points were slightly more than twice as far apart, at 3.1 meters. Their median error was 1 meter compared to ours of 1.53 meters. While the lack of identical test conditions means we cannot make any definitive claims for relative performance, we can say that LOCADIO's performance appears to be competitive with the best existing systems. In addition, LOCADIO gives an indication of whether or not a device is in motion.

10. Conclusion

Computing context from 802.11 is attractive because many spaces are already wired with 802.11 access points, and more and more mobile devices will be equipped with wireless network hardware. LOCADIO computes whether or not a device is in motion by examining the variance of Wi-Fi signal strengths, smoothing the inferences with an HMM and simple prior assumptions on how often people move. The location part of LOCADIO is based on a principled model that accounts for the building's layout, expected pedestrian speeds, our previous inference on the device's state of motion, and probabilistic signal strength signatures. While we applied this framework to 802.11 signals, it could be easily applied to other types of location sensing as well as serving as platform for sensor fusion. Because we carefully model the constraints and dynamics of location, our experiments show a median error of 1.53 meters without onerous calibration effort. Calibration effort is further reduced because we interpolate from a sparse set of calibration nodes to a dense set for doing actual inference.

Our future research in this area will concentrate on reducing calibration effort even further to make systems of this type more attractive. It is worth characterizing the tradeoff between calibration effort and accuracy. We are also interested in trying to better exploit the building's floor plan to predict signal strengths and in trying to find ways to encourage users of the system to contribute to the calibration effort for the benefit of everyone.

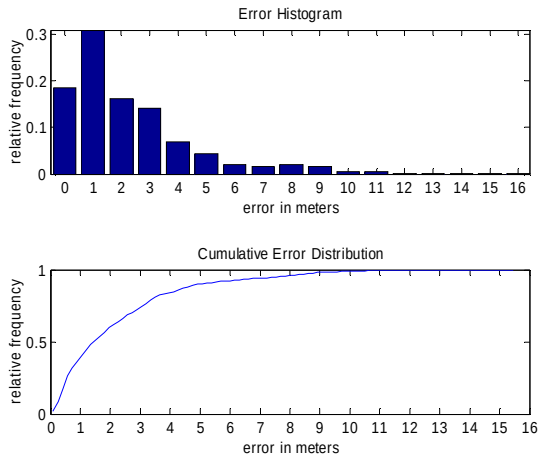


Figure 4: These plots show the histogram of error distances and cumulative error distribution for our test of 4586 different signal strength scans.

11. References

1. Intille, S.S., et al. Tools for Studying Behavior and Technology in Natural Settings. in UbiComp 2003: Ubiquitous Computing. 2003. Seattle, WA, USA: Springer-Verlag.
2. Horvitz, E., et al. Coordinate: Probabilistic Forecasting of Presence and Availability. in 18th Annual Conference on Uncertainty in AI (UAI). 2002.
3. Sparacino, F. Sto(ry)chastics: A Bayesian Network Architecture for User Modeling and Compuational Storytelling for Interactive Spaces. in UbiComp 2003: Ubiquitous Computing. 2003. Seattle, WA, USA: Springer-Verlag.
4. Patterson, D.J., et al. Inferring High-Level Behavior from Low-Level Sensors. in UbiComp 2003: Ubiquitous Computing. 2003. Seattle, WA, USA: Springer-Verlag.
5. Krumm, J., L. Williams, and G. Smith. SmartMoveX on a Graph -- An Inexpensive Active Badge Tracker. in UbiComp 2002: Ubiquitous Computing. 2002. Goteborg, Sweden: Springer-Verlag.
6. Hightower, J. and G. Borriello, Location Systems for Ubiquitous Computing. Computer, 2001. **34**(8): p. 57-66.
7. Hazas, M. and A. Ward. A Novel Broadband Ultrasonic Location System. in UbiComp 2002: Ubiquitous Computing. 2002. Goteborg, Sweden: Springer-Verlag.
8. Cox, D., V. Kindratenko, and D. Pointer. IntelliBadge: Towards Providing Location-Aware Value-Added Services at Academic Conferences. in UbiComp 2003: Ubiquitous Computing. 2003. Seattle, WA, USA: Springer-Verlag.
9. Bahl, P. and V.N. Padmanabhan. RADAR: An In-Building RF-Based Location and Tracking System. in

IEEE INFOCOM 2000. 2000. Tel-Aviv, Israel.
10. Bahl, P., V.N. Padmanabhan, and A. Balachandran, Enhancements to the RADAR User Location and Tracking System. 2000, Microsoft Research: Redmond, WA.
11. Ladd, A.M., et al. Robotics-Based Location Sensing using Wireless Ethernet. in Eighth International Conference on Mobile Computing and Networking. 2002. Atlanta, GA, USA.
12. Small, J., A. Smailagic, and D.P. Siewiorek, Determining User Location For Context Aware Computing Through the Use of a Wireless LAN Infrastructure. 2000: Carnegie Mellon University.
13. Castro, P., et al. A Probabilistic Room Location Service for Wireless Networked Environments. in Ubicomp 2001. 2001. Atlanta, GA, USA: Springer.
14. Griswold, W.G., et al., ActiveCampus - Sustaining Educational Communities through Mobile Technology. 2002, University of California, San Diego: La Jolla. p. 19.
15. Koile, K., et al. Activity Zones for Context-Aware Computing. in UbiComp 2003: Ubiquitous Computing. 2003. Seattle, WA, USA: Springer-Verlag.
16. Rabiner, L.R., A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 1989. **77**(2): p. 257-285.
17. Thrun, S., et al., Robust Monte Carlo Localization for Mobile Robots. Artificial Intelligence, 2001. **128**(1-2): p. 99-141.
18. Fruin, J.J., Pedestrian Planning and Design. 1971, New York: Metropolitan Association of Urban Designers and Environmental Planners.