

**SmartMoveX on a Graph – An
Inexpensive Active Badge Tracker**

John Krumm
Lyndsay Williams
Greg Smith

June 14, 2002

Technical Report
MSR-TR-2002-70

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Abstract. Measuring the locations of people and things in a building is an important part of ubiquitous computing. We present new hardware and software for this purpose. The hardware, called SmartMoveX, is an active badge system in which a small radio transmitter is attached to the person or thing being tracked. Receivers placed in the building's existing offices, connected to existing PCs, transmit signal strength readings to a central PC using the building's existing computer network. Combined with the low cost of the hardware, using the existing network makes this active badge system much less expensive than many others. To compute locations based on signal strength, we gathered signal strength readings from predefined location nodes in the building. We defined a graph on these nodes, which allowed us to enforce constraints on computed movements between nodes (*e.g.* cannot pass through walls) and to probabilistically enforce our expectations on transitions between connected nodes. Modeling the paths with a hidden Markov model, we used the Viterbi algorithm to compute optimal paths based on signal strengths over the node graph. The average location error was 3.05 meters, which compared favorably to a simple nearest neighbor algorithm's average location error of 4.57 meters.

1 Introduction

Knowing the indoor location of people and things is widely considered to be a key enabler of ubiquitous computing applications. These applications include location-sensitive messaging, location-based reminders, and activity inferencing. While the location of static things such as furniture, big displays, and desktop computers can reasonably be measured manually, the locations of things that move, including people, demand automatic measurement. The Global Positioning System works well outdoors, but there is not yet a comparably ubiquitous and inexpensive technology for measuring location indoors.

This paper presents a new active badge system for measuring the location of people and things indoors. The two hardware components of our system are shown in Figure 1. The active badge is the small device, slightly larger than a normal automobile remote key. The badge is attached to objects to be measured, and it transmits radio frequency (RF) signals to several receivers placed around the building. The receiver is the larger device in the figure. The receivers measure the RF signal strength from the badge transmissions. The signal strengths of the receptions are combined to compute the location of the badge in a tracking algorithm based on a hidden Markov model (HMM) defined on a graph of discrete locations in the building.

One of the system's features is its low cost, partly due to its low parts cost. The parts for a badge transmitter cost about US\$ 6, for a receiver about US\$ 16. The major cost saving, however, comes from reusing the building's existing computer network instead of requiring a custom network as many active badge systems do. Each of the receivers is connected to a normal PC via an RS-232 serial cable. When a receiver receives a badge transmission, it sends data to its host PC, which in turn forwards the data over the building's existing network to a central PC for storage and analysis. In a real implementation, the receivers would be placed in occupied offices and connected



Fig. 1. The badge transmitter is on the left, with buttons for on, off, one-time transmit and periodic transmit. The receiver, on the right, is powered externally and connects to a PC via RS-232. The receiver measures the signal strength of transmissions from the badge. The pen is just a pen.

to existing PCs. These PCs would each run our very light-weight data-forwarding program in the background. By using the building's existing network, the system avoids the high cost of a custom network devoted to the badge system. In our implementation, we used four receivers to cover 350 square meters. Not counting the cost of the badges nor any PCs, and considering the existing network infrastructure as free, this works out to a cost of about US\$ 0.18/square meter.

The location of the badge is measured based on the signal strength of the RF receptions at all the receivers. In our case, each transmission results in a vector of four signal strength readings, one element from each of the four receivers. We match the live signal strength vector with a set of calibration signal strength vectors taken from a set of known, discrete positions ("nodes") in the building, shown in Figure 3. We implemented a simple nearest-neighbor algorithm that matched the live measurement vector with the closest training vector. This gave a mean location error of 4.57 meters. We implemented another algorithm that confined the badge's path to the connections between the nodes shown in Figure 3. We formalized this algorithm as a hidden Markov

model on a graph of nodes using the same training data as for the nearest neighbor algorithm. The graph of connected nodes helps tracking by constraining the path to physically possible transitions between nodes, *e.g.* paths cannot pass through walls. The graph also encodes transition probabilities between nodes which help enforce our *a priori* expectations of people's movement between nodes. The HMM algorithm reduced the average error of our location measurements to 3.05 meters. While we show this algorithm working on signal strengths from our own badge hardware, it would apply to many other forms of sensor-based tracking in a building, including other active badge systems.

3 SmartMoveX Active Badge and Network Data Logger

The hardware for our active badge system is called SmartMoveX and was invented at Microsoft Research in Cambridge, UK. SmartMoveX consists of a small radio transmitter that transmits 433 MHz FM to multiple receivers as shown in Figure 1. Each transmission packet contains an ID number of the transmitter, a measured physical activity level, and an incrementing transmission counter to help detect missed transmissions. The transmitter uses a PIC microcontroller to read and control the functions of a tilt switch, tilt angle sensor, the transmitter itself, and four outside buttons. The tilt switch is used to shut off the transmitter after a minute of inactivity and to turn it on again when it moves, thus saving battery power. The physical activity of the transmitter is measured by how many times the tilt angle sensor measures an angle beyond a preset threshold. Although we don't use this information in our tracking application, it could be used to infer users' activities like running, walking, and sitting. The four outside buttons are used to turn on the transmitter, turn off the transmitter, put the transmitter in a periodic transmit mode, and trigger a one-time transmission. The default transmission mode is to transmit a data packet whenever the activity level exceeds a certain threshold. In our experiments, we used the periodic transmit mode which gave a new transmission every one second. When the transmissions are either periodic or triggered by the one-time transmit button, the activity level in the transmitted message is overwritten with a special value so we can tell that the transmission was triggered by a button and not activity. The cost estimate of US\$ 6 per transmitter is based on a transmitter without the tilt switch and tilt sensor, as they are not integral to the location-tracking problem we address in this paper.

The SmartMoveX receiver is a small box with connections for DC power and RS-232. It has a 15cm antenna with which to receive transmissions. Each transmission is demodulated by a receiver chip which also generates a digital radio signal strength indicator (RSSI). The transmission data (ID, physical activity level, transmission counter) and RSSI are sent to a serial communications chip which forwards the data out the RS-232 port to the host PC. With the addition of RSSI, each transmission results in the PC receiving a record consisting of the transmitter ID, activity level, transmission count, and RSSI.

On each host PC we have a logging program listening to the receiver, as shown in Figure 2. Upon receipt of a transmission record from the receiver, this program ap-

pend the name of the PC and the name of the RS-232 port to the data record and for-

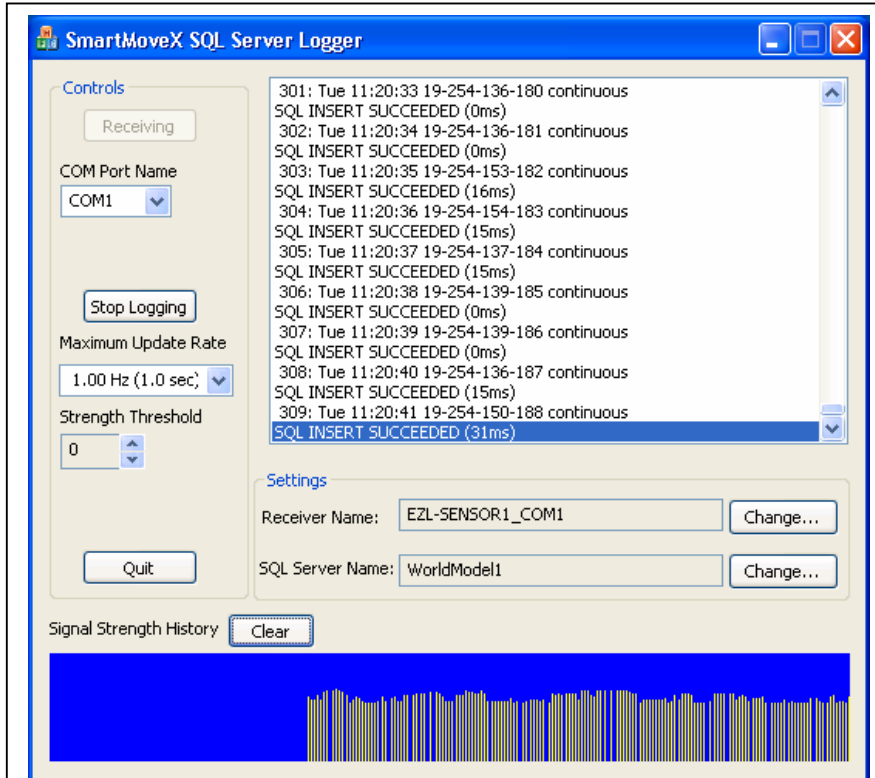


Fig. 2. This is a screen shot of our data logging program. Each receiver is connected to a host PC running an instance of this program. The program retransmits data records to a central SQL Server database. In a real system, this program would run in the background of regular office PCs that were hosting a receiver.

wards it to a SQL Server database on a central PC. This PC collects such records from all receiver PCs. To avoid clock synchronization issues on the PCs, the central PC is responsible for appending a time stamp to each incoming record. Gathering all the records in a database allows us to run queries and computations on the transmission data and allows us to run offline experiments for tracking such as was done for this paper.

3 Spatial Representation and Calibration

Each RF transmission from a mobile transmitter is heard by all the receivers in the area, resulting in a column vector of signal strength readings, s . In our case we used

four receivers, so each signal strength vector had four scalar elements. In open space we might expect the signal strength to fall off with the square of the distance between the transmitter and receiver. Unfortunately, even this simple relationship does not necessarily hold, as shown by an experiment in [1] using hardware similar to ours. Although this experiment was performed in a large, presumably open, room, the authors found a significant deviation from the inverse square law which they attributed to reflection and interference of the signal. In a continuation of this work, [2] went on to use the more sophisticated path loss model of Seidel and Rappoport[3], which was found to fit the data better. This model was also used by [4] to account for attenuation due to walls. However, it is still difficult to predict the effect of furniture, devices, and people on signal strength, particularly if the locations of these things are unknown. In fact [4] found that their analytical model of signal strengths worked significantly worse than their pure empirical model for measuring locations within a multi-room building.

Based on the difficulty of analytically modeling signal strengths, we adopted an empirical approach to predicting signal strengths similar to the RADAR system of [4]. For this approach we took a series of calibration signal strength readings at predefined node locations in our building, shown in Figure 3. We manually picked the node positions, generally one for each office, one for each office-size rectangle in larger rooms, and one outside each door in the hallways. For each of the $N = 42$ nodes, we call the calibration signal strength readings $\mathbf{s}_i^{(j)}$, where i indexes the node ($i = 0 \dots N - 1$), and j indexes the calibration sample at that node. N_i is the number of readings we took at node i , so for $\mathbf{s}_i^{(j)}$, we have $j = 0 \dots N_i - 1$. Each $\mathbf{s}_i^{(j)}$ is a vector of four signal strengths, one from each receiver. We took a total of 1256 calibration signal strength vectors. The number of calibration vectors taken for each node varied from 12 to 50, with an average of 30.

We took the calibration readings in about 30 minutes by walking around with a laptop PC wirelessly connected to our building's network running the calibration program shown in Figure 3. At each node in the building the walker clicked on the corresponding node on the map and then clicked the "I'm Here Now" button to send a time-stamped and location-stamped marker to the same SQL Server database that was simultaneously collecting transmissions from the walker's transmitter. The walker clicked "Going Away Now" to send a marker indicating the end of calibration for a particular node. This way we produced a database containing signal strength readings bracketed by markers indicating which nodes the transmissions had come from. At each node the walker rotated and moved in an effort to sample the likely positions and orientations of a person wearing a transmitter near that node. For the hallway nodes he merely spun around in place a few times. We note that our sampling of signal strengths at the nodes was not meant to measure the frequency of occurrence of signal strengths. Instead, we just tried to get a reasonable sampling of all the signal strength vectors that we would likely measure at each node. For this reason, it did not make sense to try to summarize the training data with a histogram or probability distribution function.

One interesting question is how the variation of signal strengths at a node compare with the variation of signal strengths throughout the whole building. For measuring location, we would prefer that location be the primary factor in the variation of signal strengths, and that small variations of position and orientation at a node cause rela-

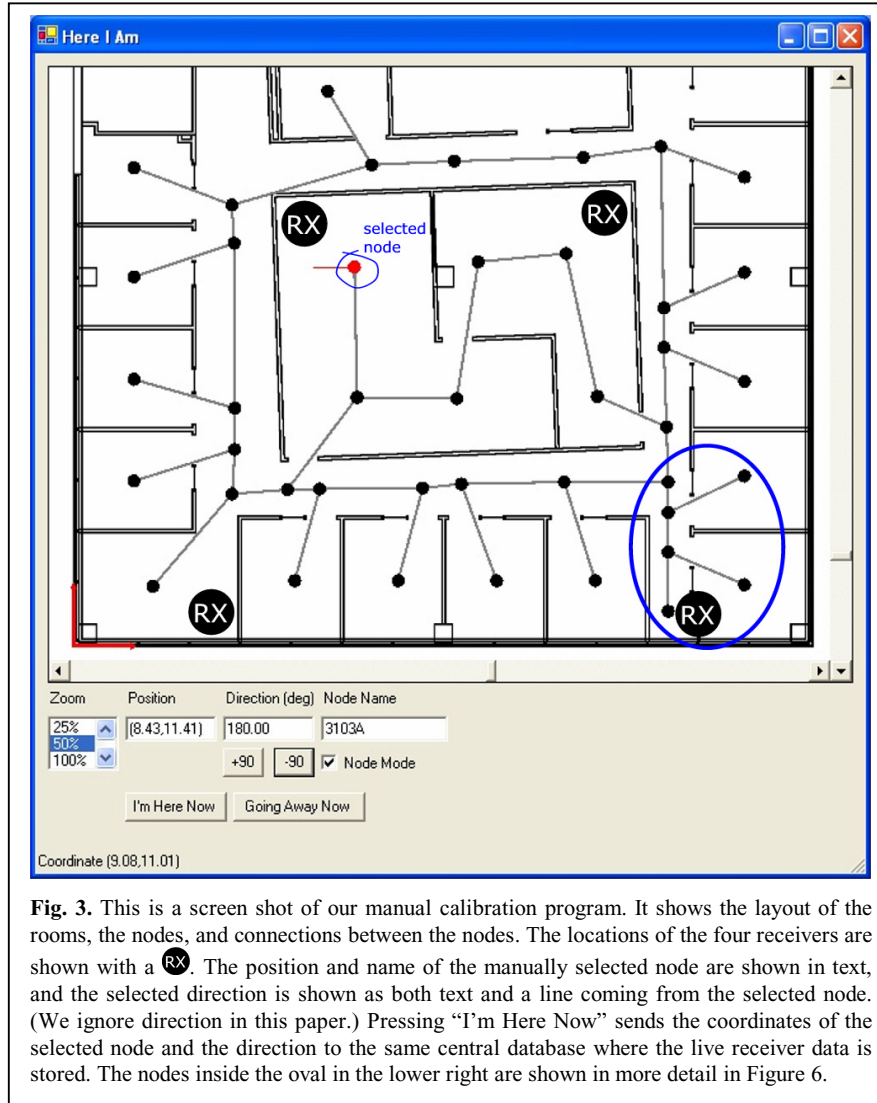
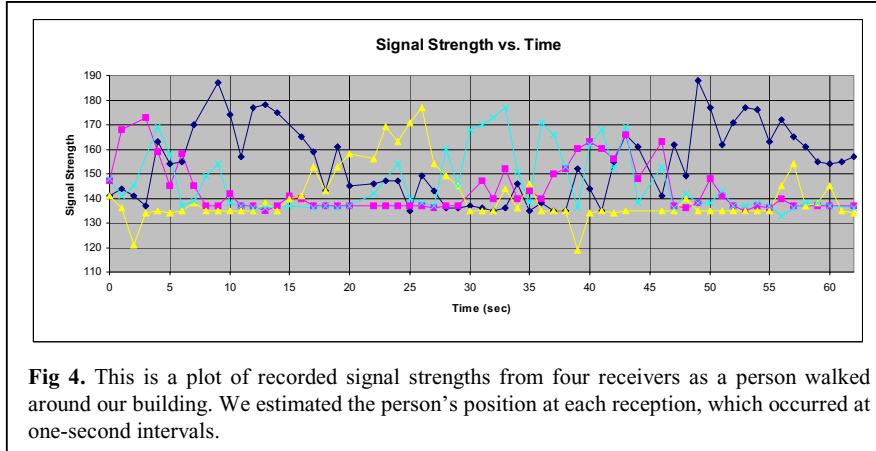


Fig. 3. This is a screen shot of our manual calibration program. It shows the layout of the rooms, the nodes, and connections between the nodes. The locations of the four receivers are shown with a **RX**. The position and name of the manually selected node are shown in text, and the selected direction is shown as both text and a line coming from the selected node. (We ignore direction in this paper.) Pressing “I’m Here Now” sends the coordinates of the selected node and the direction to the same central database where the live receiver data is stored. The nodes inside the oval in the lower right are shown in more detail in Figure 6.

tively small variations in signal strengths. Unfortunately, this was not the case for us. In fact, the maximum *intra-node* range of signal strengths was about 87% of the *inter-node* range for the whole set of nodes. Thus, rotation and small motion at a node can cause fluctuations in signal strength that are as large as fluctuations caused by moving from node to node.

We gathered test data by walking on two prescribed paths through the nodes with a transmitter pinned to the front of the walker’s shirt. The transmitter was set to periodically transmit at a one second interval, which resulted in a sequence of time-stamped

signal strength vectors $\mathbf{s}(i)$. We computed the walker’s speed by summing the distances between the visited nodes and dividing by the elapsed time of the walk. Using this speed and the distances between the nodes, we could estimate the walker’s nearest node for each of the time-stamped test transmissions. In total, our ground truth data consisted of 140 transmissions, each characterized by the closest node and a vector of four signal strengths.



4 Nearest Neighbor Location Measurement

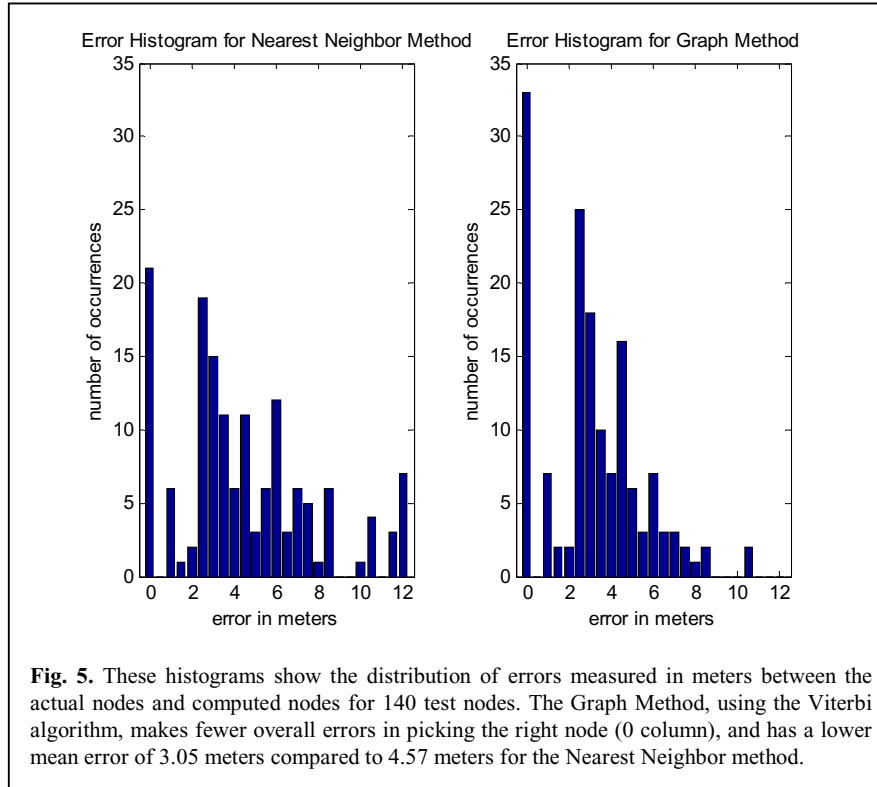
The general procedure for measuring the location of a transmitter is to compare its latest signal strength vector \mathbf{s} against the calibration signal strength vectors described above. We implemented a simple nearest-neighbor location algorithm, similar to RADAR[4], as a baseline against which to test our more sophisticated graph-based algorithm described in the next section. The nearest neighbor algorithm simply finds the $\mathbf{s}_i^{(j)}$ with the minimum Euclidian distance to \mathbf{s} and declares the transmitter to be at the node from which this $\mathbf{s}_i^{(j)}$ came. More formally, given a signal strength reading \mathbf{s} , we compute the corresponding node n as

$$n = \arg \min_{i=0 \dots N-1} \left(\min_{j=0 \dots N_i-1} \left\| \mathbf{s} - \mathbf{s}_i^{(j)} \right\| \right) \quad (1)$$

As a reminder, $\mathbf{s}_i^{(j)}$ is the j^{th} calibration vector from node i , N_i is the number of calibration vectors at node i , and N is the number of nodes.. This computation is relatively fast with only 1256 calibration vectors to compare to, certainly fast enough to keep up with the 1 Hz transmission rate of the transmitter.

We quantified the results by computing the Euclidian distances between the computed and actual nodes. The average error for our 140 test nodes was 4.57 meters. The distribution of error distances is shown in Figure 5.

One obvious omission from this algorithm is any enforcement of the fact that people



cannot move through walls and must traverse in-between nodes to get from place to place. We include these constraints in the next method and show correspondingly better results.

5 Tracking on a Graph

The nearest neighbor algorithm ignores any adjacency relationships between the nodes, so it allows instantaneous transitions between nodes that are separated by a wall and/or an arbitrarily large distance. We found we could improve the performance of the system significantly by adding path constraints that only allow physically realizable paths through the nodes. In fact, using this constraint reduced the mean error from 4.57 meters in the nearest neighbor case to 3.05 meters, using the same calibration data.

5.1 Graph and Transition Probabilities

We instantiated the constraints with the manually constructed graph of nodes shown in Figure 3. The connections between the nodes show which paths we allow. This is an easy way of preventing paths from going through walls and of preventing superhuman transitions between distant nodes.

The allowable paths between nodes are hard constraints. We also attached transition probabilities between connected nodes as a soft constraint on the likelihood of moving between nodes or remaining at the current node. We assigned the transition probabilities manually based on our assumptions about people’s behavior. For instance, we assumed that the probability of moving from a node in an office to the node outside the office’s door was 0.05 in the one-second interval between transmissions, with the remaining 0.95 being the probability of remaining at the office node. We define $a_{i \rightarrow j}$ as the probability of transitioning from node i to node j , and we pick the transition probabilities such that $a_{i \rightarrow j} \geq 0$ and $\sum_{j=0}^{N-1} a_{i \rightarrow j} = 1$, where N is the number of nodes.

We note that $a_{i \rightarrow i}$ indicates the probability of staying at node i , and it is never zero for our model. To eliminate the possibility of transitioning between node i and j , we simply set $a_{i \rightarrow j} = 0$. Figure 6 shows some of the transition probabilities we used. These nodes are from the lower right part of the map in Figure 3, where we have drawn an oval around them. The transition probabilities give a principled way of incorporating our prior beliefs about people’s behavior into the tracking algorithm. The transition probabilities are balanced against the sensor measurements in an attempt to compute paths that satisfy both.

Another *a priori* set of probabilities that we need is the probability of starting a path at a given node. We denote the probability of a path starting at node i as π_i . Since we have no prior knowledge of where a path starts, we set $\pi_i = 1/N$, where N is the number of nodes. This means the path can start at any node with equal probability.

Using the initial state probabilities and transition probabilities, the probability of a path $\{n_0, n_1, n_2, \dots, n_{T-1}\}$ is $\pi_{n_0} a_{n_0 \rightarrow n_1} a_{n_1 \rightarrow n_2} \dots a_{n_{T-2} \rightarrow n_{T-1}}$ if we disregard the signal strengths.

5.2 Hidden Markov Model

The graph, initial state probabilities, and transition probabilities represent our *a priori* knowledge of people’s behavior. The remaining element in tracking location is the signal strength data. The standard way to combine uncertain measurement data, discrete states, and transition probabilities between the states is a hidden Markov model (HMM)[5].

The “Markov” part of the HMM is manifested in our transition probabilities. We will say the sequence of nodes in a person’s path up to time $i-1$ is $\{n_0, n_1, n_2, \dots, n_{i-1}\}$. The first-order Markov assumption says that the probability of transitioning to some node n_i at time i is a function only of node n_{i-1} and not any of the other previous

nodes. This probability is what we represent in our transition probabilities, $a_{i \rightarrow j}$.

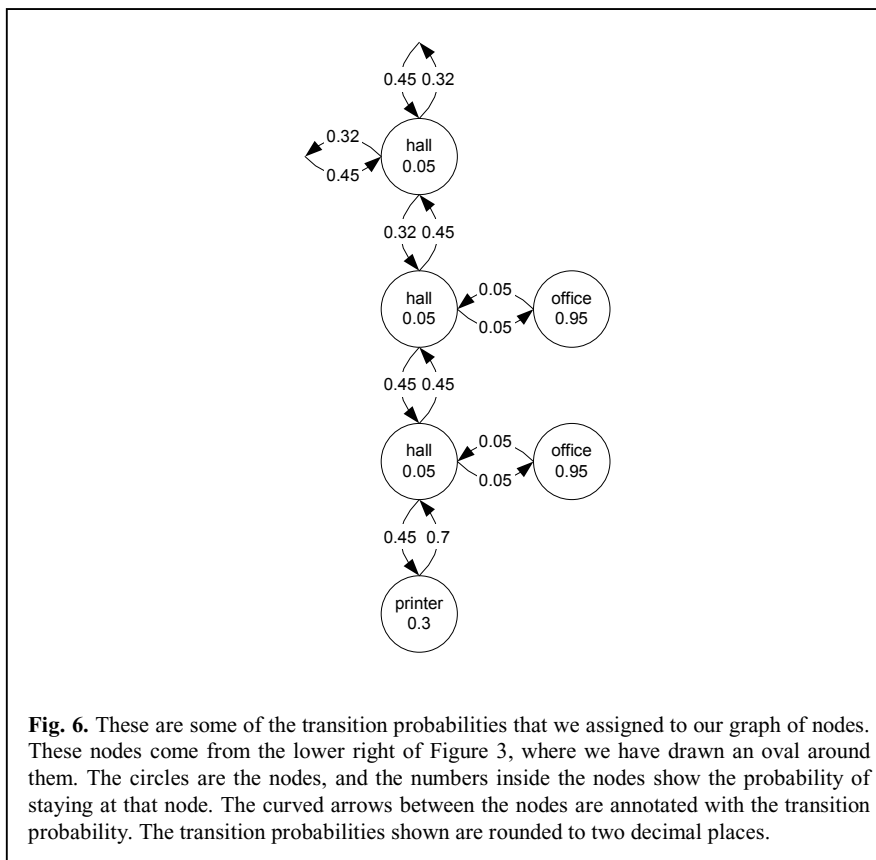


Fig. 6. These are some of the transition probabilities that we assigned to our graph of nodes. These nodes come from the lower right of Figure 3, where we have drawn an oval around them. The circles are the nodes, and the numbers inside the nodes show the probability of staying at that node. The curved arrows between the nodes are annotated with the transition probability. The transition probabilities shown are rounded to two decimal places.

Stated as an equation, the first-order Markov assumption is

$$P(n_i | n_{i-1}, n_{i-2}, \dots, n_0) = P(n_i | n_{i-1}) = a_{n_{i-1} \rightarrow n_i} \quad (2)$$

The “hidden” part of the HMM has to do with the signal strength data. Since we are not measuring the node sequence directly, we say that it is hidden from direct view, and the only things we can see are the signal strength vectors. The signal strength vector \mathbf{s} is probabilistically related to the node n_i via the probability distribution function $P(\mathbf{s} | n_i)$, which is called the observation probability function.

Ideally we would compute the observation probability with a carefully constructed physics-based model including our hardware, radio wave propagation equations, and the RF attenuation effects of objects and people. But this is too hard. The next best approach would be to thoroughly sample signal strengths at all the nodes and empirically construct observation probabilities for each one. This is also too hard, as it would

mean spending considerable time at each node and trying to mimic the behavior of a typical user at that node, including turning, sitting, standing, bending, and moving, all of which have a significant effect on signal strength. Not only would we have to anticipate all these behaviors, we would have to anticipate their frequency of occurrence.

Our approach to computing the observation probability functions was to use the training data we originally gathered for our nearest neighbor algorithm described in Section 4. We gathered this data by stopping at every node, turning and slightly moving, while recording signal strengths. This was intended to capture a series of plausible signal strength vectors, but not intended to capture their frequency of occurrence. To compute $P(\mathbf{s}|n_i)$ from this data, we first find the nearest neighbor calibration vector at the node n_i :

$$\mathbf{s}_{n_i}^* = \min_{j=0 \dots N_{n_i}-1} \|\mathbf{s} - \mathbf{s}_{n_i}^{(j)}\| \quad (3)$$

Where, as a reminder, N_{n_i} is the number of calibration vectors at node n_i , and $\mathbf{s}_{n_i}^{(j)}$ are the calibration vectors at node n_i . The assumption here is that the calibration vector from node n_i that most closely matches \mathbf{s} corresponds to the physical pose of the walker at that node that produced \mathbf{s} . The probabilistic part of the observation probability function comes from the signal noise inherent in a series of signal strength readings from a stationary transmitter, which we model as Gaussian. An experiment with our hardware suggests that the noise standard deviation of signal strength is approximately one unit. Taking $\mathbf{s}_{n_i}^*$ as the mean and assuming statistical independence among the four receivers, the observation probability function is

$$P(\mathbf{s}|n_i) = \frac{1}{(\sqrt{2\pi})^4 |\Sigma|^{0.5}} e^{-\frac{1}{2}(\mathbf{s} - \mathbf{s}_{n_i}^*)^T \Sigma^{-1} (\mathbf{s} - \mathbf{s}_{n_i}^*)} \quad (4)$$

where the covariance matrix Σ is the 4x4 identity matrix scaled by the standard deviation of the signal strength readings, which we measured to be approximately one.

With the initial state probabilities, transition probabilities, and observation probabilities, we can compute the probability of a given path through the nodes given the signal strength readings. The relevant parts are:

Transmission times	$\{t_0, t_1, t_2, \dots, t_{T-1}\}$
Nodes along path	$\mathbf{N} = \{n_0, n_1, n_2, \dots, n_{T-1}\}$
Measured signal strengths	$\mathbf{S} = \{\mathbf{s}(t_0), \mathbf{s}(t_1), \mathbf{s}(t_2), \dots, \mathbf{s}(t_{T-1})\}$
Data likelihood	$P(\mathbf{S} \mathbf{N}) = \pi_{n_0} a_{n_0 \rightarrow n_1} a_{n_1 \rightarrow n_2} \dots a_{n_{T-2} \rightarrow n_{T-1}} \prod_{i=0}^{T-1} P(\mathbf{s}(t_i) n_i)$

Given a set of signal strengths \mathbf{S} , we would like to find the path \mathbf{N}^* that maximizes the data probability $P(\mathbf{S}|\mathbf{N})$. This path will conform to our *a priori* expectations about the path as well as to the signal strength measurements. One inefficient way to do this

is to check all possible paths. The Viterbi algorithm, described next, gives a much more efficient way of finding \mathbf{N}^* .

5.3 Viterbi Algorithm

The Viterbi algorithm is described nicely in [5], so we omit a complete description here. The result of the Viterbi algorithm is the state sequence \mathbf{N}^* that maximizes $P(\mathbf{S}|\mathbf{N})$. It is based on a quantity Rabiner calls $\delta_i(j)$, where $i = 0 \dots T-1$ is the time index and $j = 0 \dots N-1$ is the node index. For time $i = 0$:

$$\delta_0(j) = \pi_j P(\mathbf{s}(t_0)|j) \quad (5)$$

This is just the product of the probability of starting at node j (which is π_j) and the probability of measuring $\mathbf{s}(t_0)$ at node j (which is $P(\mathbf{s}(t_0)|j)$). The estimate of the first node in the sequence is simply the maximum of $\delta_0(j)$ over the node index j , *i.e.*

$$n_0^* = \arg \max_{j=0 \dots N-1} (\delta_0(j)) \quad (6)$$

Subsequent values of $\delta_i(j)$ are computed recursively as

$$\delta_i(j) = \max_{k=0 \dots N-1} (\delta_{i-1}(k) a_{k \rightarrow j}) P(\mathbf{s}(t_i)|j) \quad (7)$$

Where the time index ranges over $i = 1 \dots T-1$. At each time i the node estimate is

$$n_i^* = \arg \max_{j=0 \dots N-1} (\delta_i(j)) \quad (8)$$

Thus the computations are simple, and each new signal strength reading gives a new node estimate.

Using the Viterbi algorithm along with the transition probabilities and observation probability function, we achieved an average location error of 3.05 meters compared to an average error of 4.57 meters for the nearest neighbor algorithm described above. The distribution of errors is shown in Figure 5. We attribute this improvement to three factors:

1. The graph limits paths to only those that are physically possible.
2. The transition probabilities encourage paths that conform to our *a priori* expectations of people's behavior.
3. The Viterbi algorithm uses *all* the data up to and including the current time to compute the most likely current node. The nearest neighbor algorithm using only the current signal strength vector.

The third point is not obvious from Equation (8), but the Viterbi algorithm implicitly reoptimizes the path at every time step. The algorithm described in [5] shows how to maintain auxiliary variables that can be used to compute the complete path at every time step. We don't use this part of the algorithm, because we are only interested in the current best node. But it is the case that the best node computed at each time step by Equation (8) may not end up being on the best path at subsequent time steps. At the expense of some delay, we can slightly improve our performance by waiting to compute the best node until a few more measurements have been recorded. We compared the accuracy of computing the nodes one at a time with Equation (8) to computing the single best path at the end of the walk. The average error for the single best path was 2.81 meters, slightly better than 3.05 meters reported above.

A potentially useful byproduct of the Viterbi algorithm is $\delta_i(j)$, which is the probability of being at node j at time i given the data up to and include time i . In equation form

$$\delta_i(j) = \max_{n_0, n_1, n_2, \dots, n_{i-1}} P(n_0, n_1, n_2, \dots, n_{i-1}, n_i = j, \mathbf{s}(t_0), \mathbf{s}(t_1), \mathbf{s}(t_2), \dots, \mathbf{s}(t_i)) \quad (9)$$

Instead of reporting the single best estimate of the current node to a location-aware application, $\delta_i(j)$ could be used for reporting the probability of being at each node. If the uncertainty were too high, an application could defer taking certain actions or it could consult other sources of location information.

6 Comparisons to Similar Systems

Active badge location systems date back to the work of Olivetti Research Laboratory in 1989[6], which used diffuse infrared to measure proximity. Other location measurement systems since then have used ultrasound, RF signal strength, and RF time of flight. An excellent taxonomy and survey of these systems can be found in [7].

We will compare our system against technologies with similar hardware or software. The most similar existing hardware is from the company RFIDeas Inc. Their "AirID" product allows users approaching a PC to be automatically logged on by virtue of their wearing a badge transmitting RF[8]. The receiver connects to the PC via RS-232, just like ours. Although this system was not designed to measure location in a building, Hightower *et al.*[1] investigated its use for location measuring in a room. They measured 3D location based on multiple receivers and an empirically derived function giving signal strength as a function of distance to the receiver. The hardware limited signal strength measurements to two bits, which in turn limited the system's resolution to a cube of three meters on a side. In addition, it took 10 to 20 seconds to gather readings from all the receivers into the central database for the computation of a single 3D location.

One of our system's advantages is its low infrastructure cost, which in our test consists of four inexpensive, statically mounted RF receivers connected to existing PCs via RS-232. There are systems with theoretically even less expensive infrastructure

costs, such as the SpotOn hardware by Hightower *et al.* [1] and the “Positioning by Diffusion” idea from Spratt[9]. Both these systems can theoretically operate with no fixed base stations, although real world test results have not been published yet. MIT’s Cricket[10] location-support system uses non-networked, ceiling-mounted ultrasonic transmitters whose cost per unit is about the same as our RF receivers, making its infrastructure cost similar to ours. Randell and Muller[11] describe a similar system with much higher spatial resolution and low cost. Both the infrared Active Badge and ultrasonic Active BAT[12] from AT&T Cambridge require their own dedicated network to connect statically mounted base stations, which is expensive.

Another closely related system, both in terms of hardware and software, is RADAR[4], which comes from our colleagues at Microsoft Research. RADAR’s infrastructure cost is essentially zero, since it uses the building’s existing 802.11 wireless network to locate mobile wireless devices. RADAR used signal strength in a nearest neighbor sense much as we do, and even used a rudimentary Viterbi-like algorithm[13] to help smooth trajectories. Our infrastructure could be used to complement RADAR where wireless access is not available.

The Nibble[14] system also uses 802.11 signals to compute locations in a building. Interestingly, it uses the measured signal-to-noise ratio instead of absolute signal strength. Nibble is based on a Bayesian network to compute the probability of being at any of a set of discrete locations in the building, much like our system. Nibble’s Bayesian network also supports the inclusion of transition probabilities between nodes.

7 Conclusion

The SmartMoveX transmitter and receiver hardware make an inexpensive basis for an in-building location-tracking system. Besides being cheap to build, the hardware uses existing office PCs and a building’s existing network, thus saving the cost of a network dedicated to active badges. On top of SmartMoveX we have built a tracking algorithm using signal strengths from four receivers placed around our building. Instead of trying to analytically model signal strength as a function of location, we gathered signal strength readings from a set of discrete location nodes in the building. We confined the badge-wearer’s computed location to one of the nodes, and we further confined his path to feasible connections between these nodes. In addition, transition probabilities between the nodes give soft constraints on a person’s path. We combined the graph of nodes and signal strength data in an HMM and used the Viterbi algorithm to compute the most likely path, giving an average location error of 3.05 meters. This compares favorably to a naïve nearest neighbor algorithm using the same data, which gave an average error of 4.57 meters. Thus, by constraining the possible path with the node graph, we were able to significantly improve performance. Our algorithm would apply to other forms of in-building location tracking as well.

8 References

1. Hightower, J., G. Borriello, and R. Want, SpotOn: An Indoor 3D Location Sensing Technology Based on RF Signal Strength, UW-CSE 2000-02-02, University of Washington, 2000.
2. Hightower, J., et al., Design and Calibration of the SpotOn Ad-Hoc Location Sensing System, <http://www.cs.washington.edu/homes/jeffro/pubs/hightower2001design/hightower2001design.pdf>.
3. Seidel, S.Y. and T.S. Rappoport, *914 MHz Path Loss Prediction Model for Indoor Wireless Communications in Multifloored Buildings*. IEEE Transactions on Antennas and Propagation, 1992. **40**(2): p. 207-217.
4. Bahl, P. and V.N. Padmanabhan. *RADAR: An In-Building RF-Based Location and Tracking System*. in *IEEE INFOCOM 2000*. 2000. Tel-Aviv, Israel.
5. 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.
6. Want, R., et al., *The Active Badge Location System*. ACM Transactions on Information Systems, 1992. **10**(1): p. 91-102.
7. Hightower, J. and G. Borriello, *Location Systems for Ubiquitous Computing*. Computer, 2001. **34**(8): p. 57-66.
8. Inc., R., Proximity Activated Identification, <http://www.pcprox.com/Support/AIRID/airid.html>.
9. Spratt, M., An Overview of Positioning by Diffusion, HPL-2001-207, HP Laboratories Bristol, 2001.
10. Priyantha, N.B., A. Chakraborty, and H. Balakrishnan. *The Cricket Location-Support System*. in *6th ACM International Conference on Mobile Computing and Networking*. 2000. Boston: ACM Press.
11. Randell, C. and H. Muller. *Low Cost Indoor Positioning System*. in *UbiComp 2001*. 2001. Atlanta, GA, USA: Springer.
12. Harter, A., et al. *The Anatomy of a Context-Aware Application*. in *5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. 1999. Seattle, WA: ACM Press.
13. Bahl, P., V.N. Padmanabhan, and A. Balachandran, Enhancements to the RADAR User Location and Tracking System, MSR-TR-2000-12, Microsoft Research, 2000.
14. Castro, P., et al. *A Probabilistic Room Location Service for Wireless Networked Environments*. in *UbiComp 2001*. 2001. Atlanta, GA, USA: Springer.