Indoor navigation using a diverse set of cheap, wearable sensors

Andrew R. Golding and Neal Lesh.
MERL - A Mitsubishi Electric Research Laboratory
201 Broadway, Cambridge, MA 02139
{golding,lesh}@merl.com

Abstract

We apply machine-learning techniques to the task of context-awareness, or inferring aspects of the user's state given a stream of inputs from sensors worn by the person. We focus on the task of indoor navigation, and show that by integrating information from accelerometers, magnetometers, temperature and light sensors, we can collect enough information to infer the user's location. However, our navigation algorithm performs very poorly, with almost 50% error, if we use only the raw sensor signals. Instead, we introduce a "data cooking" module that computes appropriate high-level features from the raw sensor data. By introducing these high-level features, we are able to reduce the error rate to 2% in our example environment.

1 Introduction

Context-awareness — the ability to detect aspects of the user's internal or external state — is essential for many applications of wearable computing (Pascoe, 1998). Applications include automated tour guides that augment the user's view of whichever attraction she is currently looking at (Feiner et al., 1997; Long et al., 1996); and systems that use different modes of communication depending on whether the user is engaged in a conversation (Clarkson and Pentland, 1998). In this paper, we apply machine-learning techniques to the task of inferring aspects of the user's state given a stream of inputs from sensors worn by the person. We are especially interested in integrating information from the diverse set of cheap, wearable sensors that are now available, including accelerometers, temperature sensors, and photodiodes.

We have focused on indoor navigation, the task of interactively guiding the user to a desired indoor destination. For example, the user may be looking for a certain conference room in a convention center, or a train in a large underground train station. This task requires, minimally, that the computer be aware of the person's location. Global Positioning Systems (GPS) cannot provide this information indoors or in crowded urban areas. One might, instead, use active badges (Want and Hopper, 1998; Lamming and Flynn, 1994) or beacon architectures (Long et al., 1996; Schilit, 1995), but installing and maintaining such systems involves substantial effort and expense.

Instead, we have explored the other extreme of not modifying the environment and using machine-learning techniques to infer a person's location from naturally-occurring signals in the environment. These include characteristic magnetic fields from steel beams in the walls, fixed arrangements of fluorescent lights, and temperature gradients across rooms. Additionally, the user herself provides distinctive acceleration patterns by walking up or down staircases, or riding an escalator or elevator. Figure 1 shows our initial research prototype. The user wears a "utility belt" which holds several sensor boards and a battery. The outputs from the sensors plug into a laptop, which performs the actual navigation function.

A central challenge that arises in this application, and is likely to arise in related ones, is that the raw sensor signals are unsuitable for use as direct inputs to a machine-learning algorithm. The reason is that there is too great a distance between the raw signals and the high-level inference that we wish to make. To address this, we introduce a "data cooking" module that computes high-level features from the raw sensor data. These high-level features do not add new knowledge to the system; they simply reformulate existing information into a form that the machine-learning algorithm can use more effectively. Introducing these high-level features improved performance on the indoor-navigation task dramatically. Currently, we handcraft a set of appropriate high-level features; however, we are interested in algorithms for discovering useful features through search.

The main contributions of this paper are to show that (1) by integrating information from diverse sensors, we can collect enough information to perform context-aware tasks such as indoor navigation, and (2) "cooking" the low level sensors is necessary in or-

Figure 1: Initial prototype of wearable navigation system. User wears a "utility belt" which holds the sensor boards and a battery. The outputs of the sensors feed into a laptop on which the navigation program resides.

der to make the information they contain accessible to machine-learning algorithms.

The rest of this paper is organized as follows. In Section 2, we describe the hardware platform and sensors we have been using. In Section 3, we describe our navigation algorithm. In Section 4, we describe our experiments. Section 5 discusses related work, and Section 6 concludes.

2 Hardware

The guiding principle in choosing sensors was that they be small, lightweight, low-power, cheap, and preferably non-directional — all properties intended to increase practicality for a wearable application. We currently have four types of sensors:

- 3D accelerometer: Detect the user's acceleration in three dimensions. Implemented using two 2D accelerometer boards (ADXL 202 EB from Analog Devices, Inc).

- 3D magnetometer: Detect magnetic fields in three dimensions. Our magnetometer (Honeywell HMC2003) uses three permalloy magnetoresistive transducers to measure the strength and direction of a magnetic field.

- Fluorescent light detector: Works by extracting the 60 Hz component of the signal from a photodiode aimed at the ceiling. Responds primarily to fluorescent lights, which flicker at this rate.

- Temperature sensor: Measures ambient room temperature. (We used TMP37 GT9 from Analog Devices, Inc.)

The sensors are mounted on several circuit boards and attached to a utility belt worn by the user. This arrangement (approximately) fixes the orientation of the sensors with respect to the user, circumventing problems of orientation drift (for the accelerometers and magnetometer), and aiming directional sensors as needed (the light sensor is aimed at the ceiling).

The analog outputs of the sensors are fed into an A/D card (PCM-DAS 16/330 from ComputerBoards, Inc) which plugs into a PC/MCIA slot on the laptop. Our navigation program resides on the laptop and samples the digitized signals roughly every 50 msec.

3 Navigation System

Given the view of the world provided by the sensors, the task of the navigation system is to (1) Learn a model of the world at training time, and (2) Use this model to infer the user's location at run time.

The structure of the navigation system is shown in Figure 2. The **data acquisition** module reads a tuple of sensor readings roughly every 50 msec, converting the input voltages into canonical units. The **data cooking** module augments these raw readings with *computed features*; for example, one computed feature is the variance of the user's Z acceleration. The remaining two modules, **data modelling** and **navigation**, perform the two tasks stated above: learning a model of the world at training time, and using this model to infer the user's location at run time.

The following sections describe the modules in greater detail. We first describe the representation of the world that our navigation system uses, and then discuss the data acquisition, data modelling, and navigation modules. Because the data-cooking module is best understood in the context of how the navigation algorithm works, it is presented last.

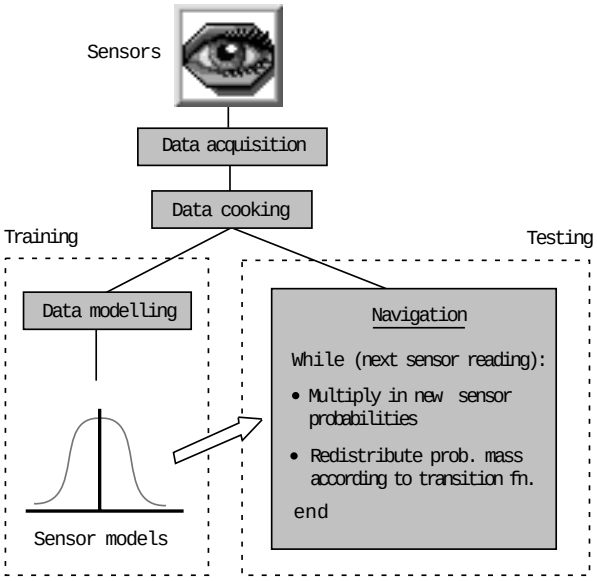

Figure 2: Structure of navigation system. The system samples the sensors roughly every 50 msec. At training time, it uses the sensor readings to learn a model of the world. At testing time, it infers the user's location from the sensor readings.

3.1 Representation

We formulate the navigation problem in terms of the "world" in which the navigation system will operate. Figure 3 shows the environment used for the experiments reported here. It contains five "locations": office, arch, upstairs, downstairs, and ping pong room. We put "locations" in quotes because a location does not correspond to a point in the world, but rather to a *transition* between points. For example, the upstairs "location" corresponds to the transition from the bottom of the stairs to the top (it could more properly be called the "going upstairs" location). This allows us to distinguish between upstairs and downstairs, which occupy the same physical space. A location such as the office corresponds to starting in the office doorway, walking around inside the office, and returning to the doorway.

3.2 Data acquisition

The data acquisition module is responsible for continually reading tuples of sensor readings, and converting them into canonical units. A tuple is read roughly every 50 msec. Table 3.2 lists the 9 sensors currently in use and their canonical units of measurement.

Table 1: Sensors read by the data acquisition module.

Sensor	Description	Units
Left X acc.	Acceleration to user's right (measured at user's left hip)	G
Left Y acc.	Acceleration forward (measured at user's left hip)	G
Right X acc.	Acceleration to user's right (measured at user's right hip)	G
Right Z acc.	Acceleration upward (measured at user's right hip)	G
Comp. X	Component of magnetic field pointing to user's right	gauss
Comp. Y	Component of magnetic field pointing forward	gauss
Comp. Z	Component of magnetic field pointing upward	gauss
Temp.	Ambient room temperature	degrees (F)
Lights	Strength of 60 Hz component of overhead lights	volts

3.3 Data modelling

The navigation system needs a model of the world, or, more particularly, a model of what its sensor signals look like throughout the indoor environment.

Training proceeds as follows: the user is given a set itinerary to traverse through the environment. He proceeds along the itinerary, clicking a handheld button once when entering and once when leaving each location. In this way, the system is kept informed of the user's true location in the world. In addition, the system samples the sensors every 50 msec, as usual, and performs its customary conversion to canonical units and augmentation with computed features (described below). The result is a set of training data in the form of a sequence of augmented sensor tuples, each accompanied by a training *label* that specifies the location at which it was collected.

From these labelled tuples, the data-modelling module learns a model of the world. The data-modelling module lumps together all readings for a given sensor at a given location, and constructs a *distribution* of these readings. The gross characteristics of the location show up in the shape of the distribution. For example, the magnetism of the arch will show up in the Compass Z distribution as an elevated mean and standard deviation compared to the Compass Z distribution of other locations. The overall model of the world produced by the data-modelling module takes the form of a set of distributions over the possible values of the sensors — one distribution for each ⟨sensor, location⟩ pair.

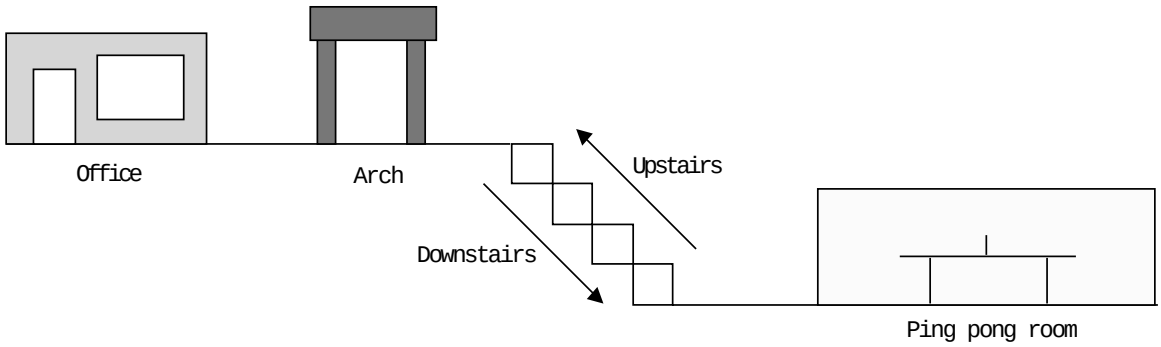

Figure 3: Example indoor environment consisting of five "locations": office, arch, upstairs, downstairs, and ping pong room.

The remaining question is what *type* of distribution will best model a set of sensor readings at a location. We tried two types: gaussians and histograms. Gaussians are appropriate under the assumption that the observed sensor values were generated by the cumulative effect of numerous random processes. Histograms test this assumption by modelling the data without making any assumptions about the type of distribution; the trick with histograms, of course, is choosing an appropriate bin width. We used a cross-validation approach, which starts by splitting the set of observed sensor values into a training set and a holdout set. The intuition is that an appropriate bin width will have the property that if we form a histogram out of the training set using that bin width, then the histogram will be highly predictive of more data from the same distribution — namely, the holdout set. Thus our approach is to try a variety of bin widths, and to select the bin width w that maximizes the probability of the holdout set, where the probability of the holdout set is calculated using a histogram probability density function constructed from the training set with bin width w.[1] Figure 4 shows examples of both gaussian and histogram distributions for the data for four sensors collected at the arch. Informally, it can be seen that the gaussian approximation is reasonably well-justified in most cases. We have therefore used gaussian models in the work reported here.

[1] More specifically, we used leave-one-out cross-validation; that is, given n sensor values to model, we did n iterations, designating each value in turn as the (singleton) holdout set, and the remaining $n-1$ values as the training set. The probabilities of the holdout set from all iterations were combined by taking the product. We chose this style of cross-validation because we found that it outperformed other styles (e.g., 2-way cross-validation) on synthetic datasets.

3.4 Navigation

The navigation system brings two types of knowledge to bear on the task of inferring the user's location. First, it has knowledge of the distribution of signal values throughout the environment, as learned at training time. For example, if it observes large Compass Z values, it can infer that the user is likely to be at the arch. Second, the system has dead-reckoning knowledge — rough knowledge about the user's change in position from one time step to the next, as inferred from accelerometer and compass readings.

The navigation algorithm incorporates these two types of knowledge in an iterative, Markov-model-like, two-step algorithm, as seen in Figure 2. The algorithm starts with some initial probability assigned to each location. Each time a sensor tuple is read, the algorithm incrementally updates the probabilities.

The first step of the update incorporates the sensor information. This is done using Bayes' rule. Let the augmented tuple of sensor readings be $S = \langle s_1, \ldots, s_n \rangle$. Then for each location ℓ, we calculate the posterior probability that the user is now at that location (given that we have observed the new sensor readings) using Bayes' rule with the conditional independence assumption:

$$P(\ell|\langle s_1, \ldots, s_n \rangle) = \left(\prod_{1 \le i \le n} P(s_i|\ell) \right) \frac{P(\ell)}{P(\langle s_1, \ldots, s_n \rangle)}.$$

The $P(s_i|\ell)$ terms are calculated using the model learned at training time for the ith sensor at location ℓ. The prior probability, $P(\ell)$, is the probability assigned to location ℓ before the update. The $P(\langle s_1, \ldots, s_n \rangle)$ term is omitted; instead, we scale the probabilities for all locations to sum to 1.

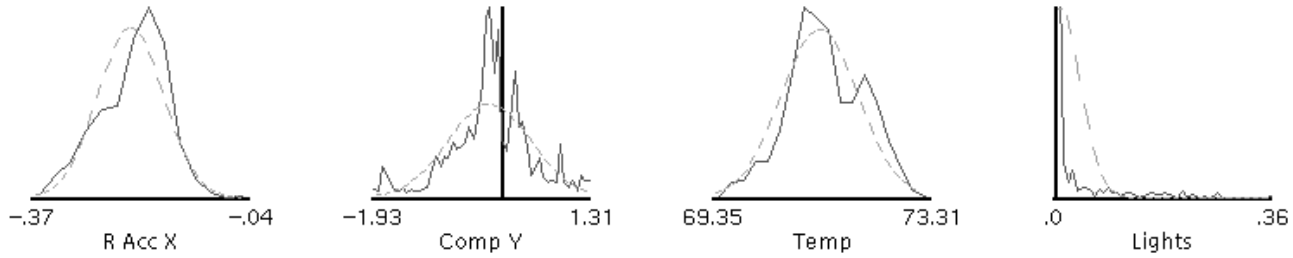

Figure 4: Gaussian (dashed) and histogram (solid) models of the data for four sensors collected at the arch. The bin width was set separately for each histogram using a leave-one-out cross-validation procedure.

The second step of the update incorporates the dead-reckoning information. This step attempts to transfer probability mass in the same direction and distance that the user moved since the last sensor readings. For example, if we had perfect dead-reckoning information that the user moved east by 5 feet, then we would transfer 100% of the probability mass from each location to the location 5 feet to the east of it. In practice, of course, we have only rather vague dead-reckoning information, and thus we are reduced to spreading out the probability mass in a less focused manner. In particular, our current dead-reckoning information consists simply of footstep detection.[2] When a footstep is detected, we spread out probability mass from a location by retaining some fraction p_{stay} of its probability at the location itself, and distributing the remaining $(1 - p_{\text{stay}})$ fraction of its probability equally among all adjacent locations. The value p_{stay} is the probability that a single footstep will not carry the user outside of her current location, and is estimated from training data.

3.5 Data cooking

The preceding sections describe our basic algorithms for training and running the navigation system. If we apply them to the raw (unaugmented) tuples of sensor readings, however, they perform rather badly. We can, however, improve the performance to high levels (see Section 4) by introducing computed features that add no new information at the knowledge level, but simply reformulate the existing information into a form in which it can be used more effectively. Below we discuss the specific problems that we identified with using the raw sensor readings for navigation, and the computed features that we developed to address these problems.

[2]We detect a footstep iff the the Z acceleration is greater than 0.05 G after being less than 0.05 G for at least 250 msec.

One problem that quickly became apparent was that of overcounting evidence due to excessive independence assumptions in the Bayesian update part of the navigation algorithm. Not only are all features in a tuple assumed to be conditionally independent, but consecutive tuples are assumed independent as well. Thus if the user stands under a light, we find that the algorithm quickly comes to believe she's in whichever room was best-lit during training. The solution is to add a computed feature that triggers only when the lights go from off to on, rather than responding whenever the lights are on.

A second problem concerns absolute versus relative measurements. Sensor readings are often subject to drift for various reasons; for example, an accelerometer may shift around somewhat on the utility belt, thereby changing the perceived value of gravity (which nominally has a Z acceleration of -1 G). Clearly it is bad for the navigation algorithm to depend on the absolute Z accelerations that are read. To address this, we compute a feature which, for each accelerometer, reports the delta between the current value and a running average of the last, say, 10 minutes' worth of readings.

A third problem arose in the context of the Z acceleration sensor: we observe that when a user walks down stairs, the amplitude of the oscillations in the Z acceleration increases. This effect does in fact show up if one looks at the distribution of Z acceleration values for the downstairs location: the distribution has a wider shape than the Z acceleration distributions for other locations. However, this is still not enough of a clue for the navigation algorithm to latch onto easily. If we make the information about wider shape available *directly*, however, by adding a computed feature for the *variance* of the Z acceleration, then the algorithm becomes able to discriminate the downstairs location reliably. Figure 5 displays the raw Z acceler-

ation feature together with the variance of Z acceleration feature. The latter, cooked feature is much more obviously correlated with walking down stairs.

A final problem concerns noisy or unreliable sensor distributions, a difficulty which occurs frequently in the navigation domain. For example, the Compass Z sensor often hovers in an intermediate range indicating a fairly weak magnetic field. Such a value is not particularly indicative of *any* location; yet the value will inevitably turn out to favor one location over another (perhaps by a small amount). Incorporating this kind of evidence into the calculation is never good; it introduces noise, and the noise does not always balance out across locations. There are several ways to clean up such evidence; the approach we have adopted is to establish an *in-bounds range* for each sensor. If a sensor value falls outside this range, it is ignored (i.e., not used to perform Bayesian updates). For example, the Compass Z in-bounds range is above 1.2 gauss or below -1.2 gauss. If a weak Compass Z value, such as 0.3 gauss is read, it is discarded. That is, the algorithm does not use this reading of the Compass Z sensor in the Bayesian update step. Currently the ranges are set by hand, but we are considering methods for setting or adjusting them automatically.

4 Experiments

We were not able to develop a single metric by which to measure the performance of the navigation algorithm. One possible metric is the percentage of time that the navigation algorithm "knows where it is", i.e., that the location it reports as most likely is, in fact, the location it is in. However, not all errors seem equally important. For example, when one first enters an office, it seems reasonable for the navigation algorithm to not immediately report that it is in the office. This error seems less significant than, for example, suddenly reporting that it is going downstairs in the middle of a visit to an office.

We divide each visit to a location into a *header* and a *body*. The *header* is the time from the point the person enters a location until the navigation algorithm first reports that it is in any location other than the one it was reporting at the very beginning of the visit.[3] The *body* is the remaining time spent in that location. We measure three aspects of the performance of the navigation algorithm: (1) the average length of the header, (2) the number of errors, i.e., the number of times the algorithm changes the location it is reporting to a wrong answer, and (3) the error rate, i.e. the fraction of time (within the body) that the algorithm's prevailing guess is wrong.

[3]However, if the navigation algorithm reports the same location throughout a visit or is already reporting the correct location at the beginning of a visit, then the header size is 0.

Table 2: Performance results.

high-level features	restricted ranges	time until first guess	errors per visit	error rate
yes	yes	4.8 secs	0.3	2%
yes	no	2.0 secs	2.6	33%
no	yes	3.4 secs	10.1	52%
no	no	2.1 secs	10.5	49%

We trained the navigation algorithm a total of 28 visits to the locations in our test environment and tested the algorithm on a set consisting of 19 visits. A typical visit to a location lasted about 20–30 seconds. We handcrafted the set of high-level features and ranges used by our navigation algorithm on some of the training data, prior to collecting the test data.

As shown in Table 2, our navigation algorithm had very low error rates when we used both the cooked features and the restricted ranges on the features, although it took an average of almost 5 seconds to recognize a location once the person entered it. The algorithm's performance degraded to unacceptably high error rates if we turned off the computed features or the restricted ranges, or both. These results show that the machine-learning algorithm could not use the raw signals effectively, unless the information they contain was reformulated in terms of the computed features.

The performance of our system degrades even further if we do not make use of the history of sensor readings, but try to guess the location of the user only based on the current sensor readings. Although not shown in the table, without the extra features or the ranges, the navigation algorithm using only the current sensor readings has a 55% error rate and makes 52 errors, on average, per visit. With the extra features and the ranges, but only using the current sensor readings to guess where it is, the algorithm has a 49% error rate, and an average of 5.5 errors per visit.

5 Related work

While we have focused on the task of inferring state information about the user by combining information from the available sensors, much previous work has investigated the *desirability* of obtaining this information. Pascoe (1998), for example, generally considers the value of context-aware computing, and describes a case study of incorporating GPS information into a system that assists a field worker who is observing

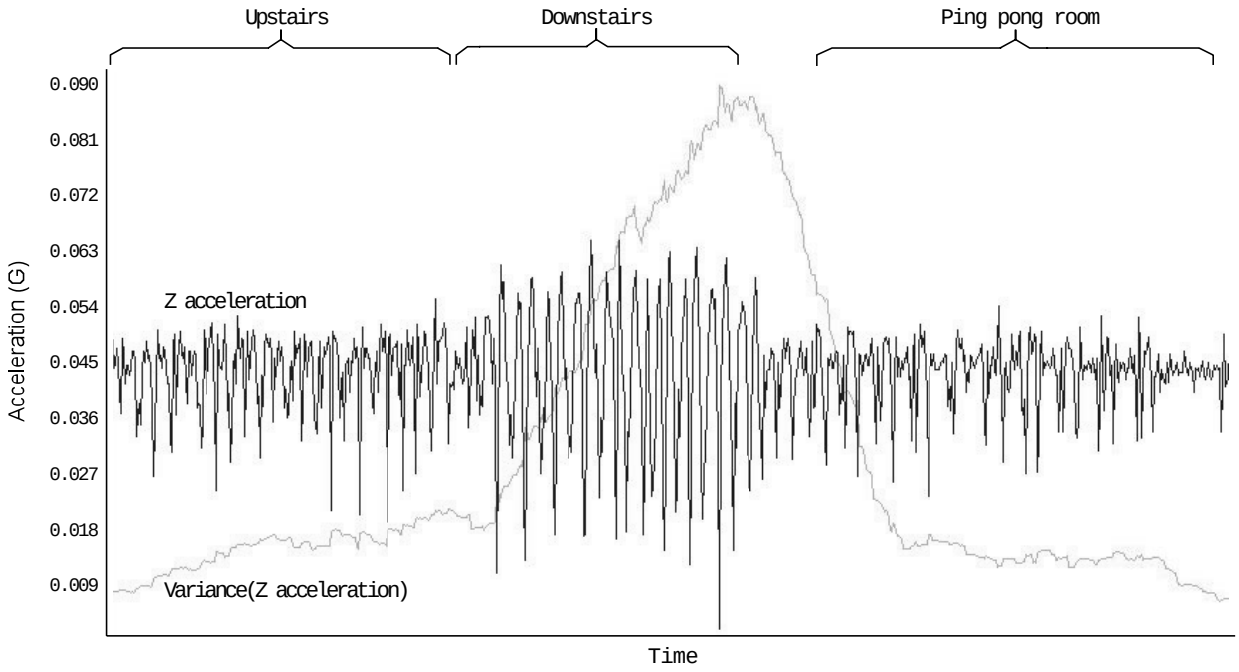

Figure 5: The Z acceleration feature (black) versus the Variance(Z acceleration) feature (gray). The features are shown over time as the user walks upstairs, downstairs, and into the ping pong room. Variance is computed over the last 100 readings.

giraffes. Feiner et al. (1997) and Long et al. (1996) describe automated tour guides that augment a person's view of the attractions that they are viewing. Ertan et al. (1998) and Ram and Sharf (1998) describe systems that provide navigational assistance for visually impaired people.

Our work is close in spirit and objectives to work being done by the Technology for Enabling Awareness group (http://www.omega.it/research/tea) which is developing hardware and software to enable portable devices, such as loptops, to have a higher degree of awareness of their user's context such as location and activity. Additionally, previous work has made use of the diverse variety of sensors that are available, using them to detect a variety of aspects of the user's internal and external state. For example, Picard and Healey (1997) describe how to use wearable sensors to detect the wearer's affective patterns such as expressions of joy or anger. Similarly, Myrtek and Brugner (1996) investigate the correlations between physiological parameters (heart rate, physical activity) and psychological parameters (excitement, enjoyment). Additionally, van Someren et al. (1998) use actigraphy (processed accelerometer readings) for detailed and long-term measurement of hand tremors to help Parkinson patients manage their symptoms. Blood et al. (1997) compare three wearable sensors for the task of detecting when a person is sleeping. Teicher (1995) shows correlations between levels of activity and mood disorders and attention-deficit hyperactivity disorder.

Finally, previous work has used video and audio sensors to provide context-awareness for wearable computers. Starner, Schiele, and Pentland (1998) tackle the problem of indoor navigation using the output of two cameras worn on the heads of MIT students playing a game called Patrol. They use Hidden Markov models (HMMs) with a small number of states trained on a feature vector derived by sampling various regions from the images returned by the cameras. They obtained encouraging results (82% on their scoring system, which is not directly comparable to ours), and discuss possible improvements such as incorporating optical flow or inertial sensors in order to decide when to process frames. Similarly, Clarkson and Pentland (1998) train an HMM on an audio signal to detect

whether or not a person is engaged in a conversation so as to decide how to notify them of an incoming message. In contrast, we have focused on combining evidence from different sensors rather than optimizing one source of evidence. Additionally, our algorithms model the world directly (i.e., one state for every location) and thus produce a probability distribution over all possible locations. Of course, the work we have described in this paper is complementary with a video- or audio-based one. For example, we could use video or audio as additional sensors. We chose not to in our initial design because we suspected the video images would vary wildly based on the orientation of the person in the rooms.

6 Conclusion

The work reported here addresses the problem of enabling a wearable computer to be context-aware — that is, to take into account aspects of the user's internal or external state. We focus on the task of indoor navigation, where the particular aspect of the user's state that is of interest is the user's physical location. By affixing a diverse set of cheap, wearable sensors to the user, including accelerometers, magnetometers, and temperature sensors, and applying machine-learning techniques, we find that we are able to infer the user's location quite accurately in a simplified office environment. However, this high accuracy was obtained only when we augmented the representation with "cooked" versions of the original sensor readings; the raw sensor readings were too low-level to be used directly to perform the high-level inference. Two conclusions emerge: (1) By obtaining high accuracy with our cooked representation, we have demonstrated that our approach of integrating information from a diverse set of low-level sensors is adequate to obtain enough raw knowledge to perform context-aware tasks such as indoor navigation; and (2) The raw sensor values must be "cooked" appropriately to put the knowledge in a form that is accessible to machine-learning algorithms. Currently, we have handcrafted the definitions of our cooked features; a direction for future work is to develop methods for discovering such feature definitions automatically or semi-automatically.

Acknowledgements

This project would not have been possible without the help of Darren Leigh, who designed and built the hardware sensors, as well as being a good all-around source of ideas. We would also like to thank Bill Yerazunis and Baback Moghaddam for their helpful thoughts and comments on the project.

References

Blood, ML., RL. Sack, DC. Percy, and JC. Pen. 1997. A comparison of sleep detection by wrist actigraphy, behavioral response, and polysomnography. *Sleep*, 20(6):388–95, June.

Clarkson, B. and A. Pentland. 1998. Extracting context from environmental audio. In *Proc. of the second international symposium on wearable computers*, pages 154–155, October.

Ertan, S., L. Clare, A. Willets, H. Tan, and A. Pentland. 1998. A wearable haptic navigation guidance system. In *Proc. of the second international symposium on wearable computers*, pages 164–165, October.

Feiner, S., B. MacIntyre, T. Hollerer, and A. Webster. 1997. A touring machine: Prototyping 3D mobile augmented relatiy systems for exploring the urban environment. In *Proc. of the first international symposium on wearable computers*, October.

Lamming, M. and M. Flynn. 1994. Forget-me-not: Intimate computing in support of human memory. In *FRIEND21: international symposium on next generation Human interface*, pages 125–128.

Long, S., R. Kooper, G.D. Abowd, and C. G. Atkeson. 1996. Rapid prototyping of mobile context-aware applications: the cyberguide case study. In *Proc. of the second ACM international conference on mobile computing and networking*, November.

Myrtek, M. and G. Brugner. 1996. Perception of emotions in everyday life: studies with patients and normals. *Biological Psychology*, 42(1-2):147=–64, January.

Pascoe, J. 1998. Adding generic contextual capabilities to wearables computers. In *Proc. of the second international symposium on wearable computers*, pages 92–99, October.

Picard, R.W. and J. Healey. 1997. Affective wearables. *Personal Technologies*, 1:231–240.

Ram, S. and J. Sharf. 1998. The people sensor: A mobility aid for the visually impaired. In *Proc. of the second international symposium on wearable computers*, pages 166–167, October.

Schilit, W. 1995. *System architecture for context-aware mobile computing.* Ph.D. thesis, Columbia University.

Starner, T., B. Schiele, and A. Pentland. 1998. Visual contextual awareness in wearable computing. In *Proc. of the second international symposium on wearable computers*, pages 50–57, October.

Teicher, MH. 1995. Actigraphy and motion analysis: new tools for psychiatry. *Harvard Review of Psychiatry.*, 3(1):18–35.

van Someren, EJ., BF. Vonk, WA. Thijssen, JD. Speelman, PR. Schuurman, M. Mirmiran, and DF. Swaab. 1998. A new actigraph for long-term registration of the duration and intensity of tremor and movement. *IEEE Transactions on Biomedical Engineering*, 45(3):286–95.

Want, R. and A. Hopper. 1998. Active badges and personal interactive computing objects. *IEEE Transactions on Consumer Electronics*, 38(1):10–20.