Learning Significant Locations and Predicting User Movement with GPS

Daniel Ashbrook and Thad Starner
College Of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280 USA
{anjiro, thad}@cc.gatech.edu

Abstract

Wearable computers have the potential to act as intelligent agents in everyday life and assist the user in a variety of tasks depending on the context. Location is the most common form of context used by these agents to determine the user's task. However, another potential use is the creation of a predictive model of the user's future movements. We present a system that automatically clusters GPS data taken over an extended period of time into meaningful locations at multiple scales. These locations are then incorporated into a Markov model that can be consulted for use with a variety of applications in both single–user and collaborative scenarios.

1 Introduction

For any user–assisting technology to be truly useful and not merely irritating, it must have some knowledge of the user to be assisted: it must understand—or at least predict—what the user will do, when and where she will do it, and, ideally, the reason for her actions. User modeling is a necessary step toward gaining this understanding.

Csinger defines user modeling as "...the acquisition or exploitation of explicit, consultable models of either the human users of systems or the computational agents which constitute the systems" [3]. This definition, however, raises the question of "what constitutes a model?" For the purposes of our research, we consider a model to be a collection of data on some particular aspect of a human user's behavior that, when associated with a limited set of contextual clues, yields predictions on what behavior the human will engage in next.

In this paper, we describe research investigating one facet of user modeling, that of location. Location is one of the most commonly used forms of context: it is usually easy to collect location data, and other pieces of context may be inferred from location, such as presence of other people. In our research, we used off–the–shelf GPS hardware to collect location data in a simple and reliable manner.

1.1 Previous Work

In his Master's thesis [9], Jon Orwant describes Doppelgänger, a "user modeling shell" that learns to predict a user's likes and dislikes. Orwant uses active badges, Unix logins, and schedule files to guess where in a building a particular user is likely to go. The locations in the Doppelgänger system were, in a sense, hard–coded, since a user was detected by the infrastructure.

Location prediction systems have become of interest in the cellular network community in recent years. The United States government wants to be able to locate people who place emergency 911 calls from cell phones, and various location–based contextual services are being discussed. Bhattacharya and Das describe a cellular–user tracking system for call delivery that uses transitions between wireless cells as input to a Markov model [1]. As users move between cells, or stay in a cell for a long period of time, the model is updated and the network has to try fewer cells to successfully deliver a call.

Unlike those using fixed sensors, systems using GPS to detect location must have some method to determine which locations are significant, and which may be ignored. In their investigations of automatic travel diaries [13], Wolf et. al. used stopping time to mark the starting and ending points of trips. In their work on the *comMotion* system [8], Marmasse and Schmandt used loss of GPS signals to detect buildings.

2 Applications

Potential applications for a location–modeling system fall into two main categories: single–user, or non–collaborative, and multi–user, or collaborative. Single–user applications are those that can be applied to one person with only her own location model. Collaborative applications, on the other hand, are useful only with two or more location models, and may be used to promote cooperation and collaboration between individuals.

2.1 Single–User Applications

In their paper on the *comMotion* system [8], Marmasse and Schmandt explore the idea of an agent that learns frequented locations. The user may associate a to–do list with each location, in the form of text or audio. When the user reaches a location, the applicable to–do list is displayed. One to–do example Marmasse provides is that of reminding the user of her shopping list as she nears a grocery store. If the user was driving, however, reminding her that she needed to visit the store as she passes it could be frustrating and distracting. However, reminding the user several miles in advance, or even as she enters her car, would be more productive.

Many other early–reminder applications may easily be imagined. For example, suppose a user has a library book she needs to return. If her location model predicts she'll be going by the library later in the day, she can be reminded to take the book on the way out of the house. Reminders, however, are not the only possible use for the single user. Wearable computer systems issues may be addressed as well.

Wireless networks are often inaccessible due to lack of infrastructure, radio shadows due to buildings, power requirements and other problems. In some cases, however, this lack of connectivity may be hidden from the user by caching [7]. For example, if a user composes an e–mail while riding the subway, the wearable may add the message to its outgoing queue and wait to send it until the network is available. On the other hand, if the message is urgent, this behavior may not be appropriate. If the user is predicted to be out of range of the network for some time, she could be alerted of possible alternate travel paths that will allow her message to be sent.

For less urgent e–mail and Internet services, it may be desirable to delay transmission even when a wireless connection is available. Energy is one of the most precious resources for mobile devices, and the amount of energy needed to transmit a message may go up with the fourth power of distance in some situations [2]. In addition, the cost of transmitting a message may vary with the time of day and the type of service that is used. Location prediction abilities could allow a wearable computer to optimize its transmissions based on cost and availability of service in various locations and the knowledge of how its user moves throughout the day.

2.2 Multi–User Applications

When multiple people share their location models, either fully or partially, many useful applications become possible. The models could be shared by giving full or partial copies to trusted associates, delegating the coordination of models to a central service, or allowing remote queries from colleagues whenever information is needed. These three options run from more convenient to more accurate: copying models allows instant access to another person's model, but doesn't guarantee that it will be up to date; a central service allows models to be updated whenever one party has connectivity, making the model more likely to be valid; and remote queries would ensure accuracy at the possible cost of high latency.

Regardless of the sharing mechanism, there are several interesting applications that could be implemented. The simplest is also the most privacy–preserving: a user, who we'll call Alice, could ask, "Will I see Bob today?" This type of query gives Alice a useful piece of information—if she needs to bring a thick textbook to Bob, she will only want to take it with her if she's likely to see Bob that day. The query also preserves Bob's privacy, since it is never revealed to Alice when she'll see Bob, where she'll see Bob, or where else Bob has been that day.

One step up from this simple application is the common problem of scheduling a meeting for several people. In his description of the QuickStep platform [10], Jörg Roth described a sample application that facilitated scheduling meetings by showing each user the others' calendars. While a participant would see each user's calendar and when they were unavailable, the labels that showed the reason for the unavailability were removed. We can preserve privacy to an even greater extent by hiding the schedules themselves from the group and deferring the schedule suggestions to some central system. This system could not only find a time when every member is available, but a time when each person is close to the desired meeting location.

Another possibility is encouraging serendipitous meetings between colleagues. Suppose Alice's model indicates that she has lunch at a certain café every

Thursday. If Bob happens to be in that general area on Thursday near lunchtime, he could be notified that Alice might be eating nearby so he can give her a call and meet with her.

Michael Terry's Social Net [12] presents an innovative way to meet people with similar interests. It "searches for patterns of physical proximity between people, *over time*, to infer shared interests between users." Social Net has been implemented using the wireless capabilities of the Cybiko [4] toy to detect proximity. The Cybiko's low (300 foot) range suggests that using location models might be a better way to provide proximity input to Social Net—two people who work in the same building, for example, might be more than 300 feet away from each other almost all of the time. A model that represents places rather than proximity would have a better chance of noticing those people's co–location.

Rather than linking people with similar interests, location modeling could allow otherwise unconnected individuals to exchange favors with each other. For example, suppose Alice needs a book on cryptography for her research, but will not have time to go to the library for several days. She could submit her request to some central arbitration system. The system could look at all of the location models it has for various people, and perhaps discover that Bob will be near the library soon, and not long thereafter near Alice's location. If he is willing, Bob can then pick up the book and deliver it to Alice. One can imagine a sort of reputation system based on favors like these, such as that described in Bruce Stirling's book *Distraction* [11].

The final application for location models we will discuss is that of intelligent interruption. While James Hudson demonstrated that the nature and desirability of interruption is often uncertain [6], there are certain situations in which being interrupted by, say, a ringing cellular phone is definitely *not* acceptable. By allowing one's wearable computer to manage potential interruptions like cell phones, location models can be used to make an intelligent guess about whether the user is interruptible or not.

Using the cell phone as an example, imagine that the user has a class from 4:00 to 5:00 every day. When the user enters the classroom, her wearable automatically turns her cell phone ringer off. If someone calls during the class, her wearable answers for her, perhaps telling the caller that she will be available around 5:00 when her class is over. As the user walks out of class, her wearable turns the ringer on her phone back on and alerts her that someone has called.

3 Implementation

In order to begin investigating the benefits provided by location modeling, we constructed a system to record and model an individual's travel. In its current form, the system performs modeling and prediction on different scales, and allows queries to the model such as "The user is currently at home. What is the most likely place she will go next?" and "How likely is the user to stop at the grocery store on the way home from work?" The answers to these questions for several users can lead to serendipitous meetings: if the response to "Where is the user most likely to go next?" is the same for two colleagues, they can be alerted that they're likely to meet each other.

The system is comprised of two parts: a hardware component and a software component. The hardware handles collection of data, and the software processes the data and, using a Markov model, makes predictions based upon that data.

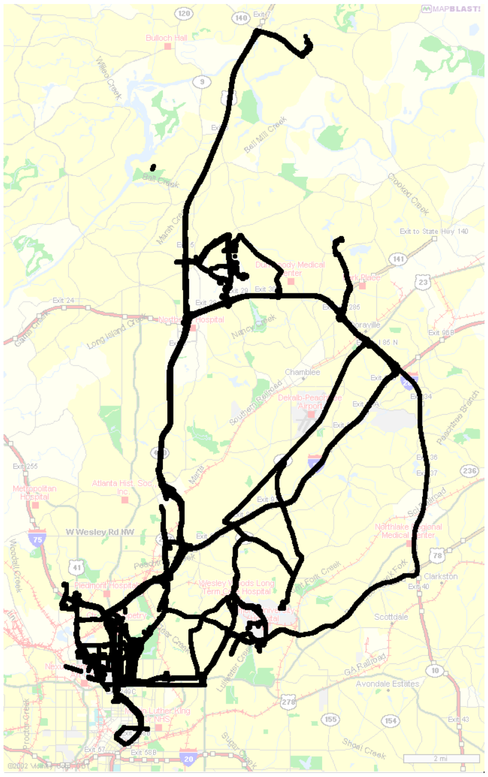

Figure 1: *All GPS data captured during the four month period.*

3.1 Hardware

Using a Garmin 35-LVS wearable GPS receiver and a GPS data logger, both from GeoStats [5], we collected data from one user for a period of four months. During those four months, the user traveled mainly in and around Atlanta, Georgia. Our data logger recorded the output from the GPS receiver at an interval of once a second, but only if the receiver was moving at one mile an hour or greater. Because humans walk at an average of three miles per hour, we capture most forms of transit. Figure 1 shows the data we collected superimposed on a map of Atlanta.

While in many respects GPS is an ideal sensor, there were some drawbacks. Although Selective Availability has been turned off, the accuracy of our GPS receiver was 15 meters; this means that the same physical location will have a different GPS coordinate from day to day. Another limitation we had was with battery life. The GPS receiver draws 500–600 mW, and was connected to six "D"–size batteries. This allowed the receiver to operate for approximately a week's time; ironically, this was worse for continuous data collection than if the battery required replacing once per night. This was because there was no visible indicator as to whether the batteries were still good or not, and the user often missed a day or two of data.

3.2 Software

Although it might seem strange to throw away data points, logging data only while the user is traveling at greater than one mile per hour actually helps to pre–process the data. Since we are largely interested in the locations where the user spends her time, rather than how she gets there, we can look for time gaps in the data that indicate that the user stopped moving. The same time gaps will also occur when the GPS receiver cannot find any GPS satellites, such as when the user enters a building. Whenever a point is found that has more than a certain time t between it and the previous point, we conclude that the point marks a significant location. Collectively, we call these significant locations *places*.

3.2.1 Determining *places*

When analyzing our four months of data, we discovered that t and the number of *places* followed a fairly linear relationship (Figure 2). As the threshold t approaches zero, the number of *places* grows ever more rapidly (at $t =$ one minute, 97,028 *places* were found), but there are few indicators as a good value for t. In the end, we (arbitrarily) decided on ten minutes; after more data has been collected, we may revisit this problem and try to determine what value of t yields the most accurate predictions.

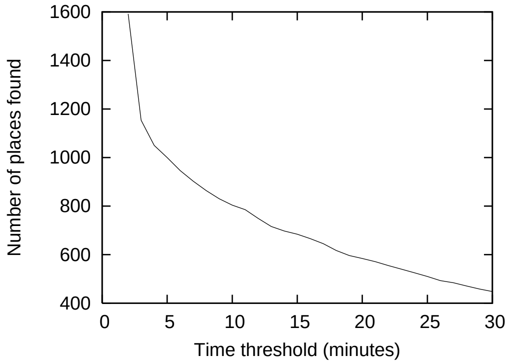

Figure 2: *Number of* places *found for varying values of time threshold t.*

3.2.2 Clustering places into locations

Because multiple GPS measurements taken in the same physical location can vary by as much as 15 meters, the logger will not record exactly the same GPS point for a location even if the user stops for ten minutes at precisely the same point every day. For this reason, we create clusters of *places* using a variant of the k–means clustering algorithm. The basic idea is to take one *place* point and a radius. All the points within this radius are marked, and the mean of these points is found. The mean is then taken as the new center point, and the process is repeated. This continues until the mean stops changing. When the mean no longer moves, all points within its radius are placed in its cluster (or *location*) and removed from consideration. The procedure repeats until no *places* remain and we are left with a collection of *locations*.

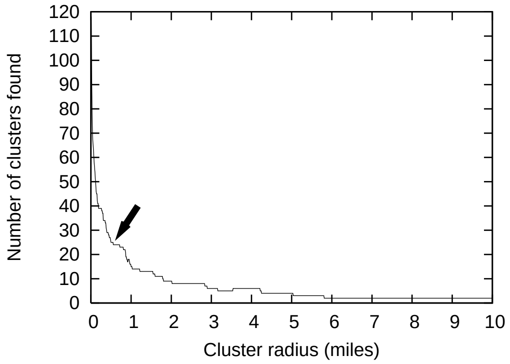

Figure 3: *Number of locations found as cluster radius changes. The arrow denotes a knee in the graph—the radius just before the number of locations begins to converge to the number of* places.

To make our predictions as useful and specific as possible, we want to have *locations* with small radii, thus differentiating between as many distinct locations as possible. However, if we make the radii too small, we will end up with only one point per *location*, and be faced with predictions between all the points shown in Figure 1. To find an optimal radius, we run the above clustering algorithm several times with varying radii. We then plot the results on a graph and look for a "knee" (Figure 3). The knee signifies the radius just before the number of *locations* begins to converge to the number of points.

In order to find the knee in the curve, we start at the right–hand side of the graph, and work our way leftwards. For each point, we find the average of it and the next n points on the right. If the current point minus the average is greater than some threshold, we use it as the knee point. This method is a simple variant of looking for a significant change in the slope of the graph.

Figure 4 shows the *locations* found for a time threshold of ten minutes and a *location* radius of one half mile. Note the vast reduction in points from the full set of data in Figure 1—while the user traveled by car and foot over 1,600 miles, there are only a handful of places that the user actually stopped at for any length of time.

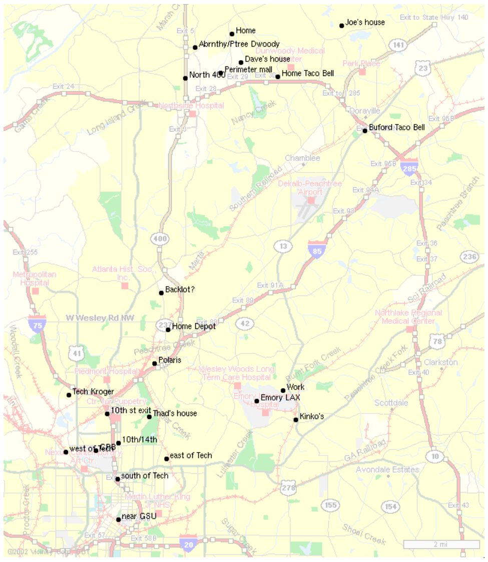

Figure 4: *Significant points in the Atlanta area, as determined with a time threshold of* $t = 10$ *minutes and a location radius of* $r = .5$ *miles.*

3.2.3 Learning *sublocations*

When creating our *locations* with a particular radius, we may subsume smaller–scale paths—for example, if our radius is chosen to make prediction efficient on a city–wide scale, we may obscure prediction opportunities on a campus–wide scale. Choosing a small radius to allow for multiple campus locations, however, will remove the ability to predict broader trips such as "Campus→Home" in favor of things like "Physics building→Home," "Math building→Home," and so forth.

To solve this problem, we introduce the concept of *sublocations*. For every cluster we find in the main list of *places*, we determine if there is a network of sublocations within it that may be exploited. This is accomplished by taking the points within each *location* and running them through the same clustering algorithm described above, including graphing varying radii and looking for the knee in the graph (Figure 5). If the knee exists, that radius is used to form *sublocations*, which can then have the same prediction techniques applied as the main *locations*. If no knee exists, we assume that there are not enough points within the *location* to form *sublocations*.

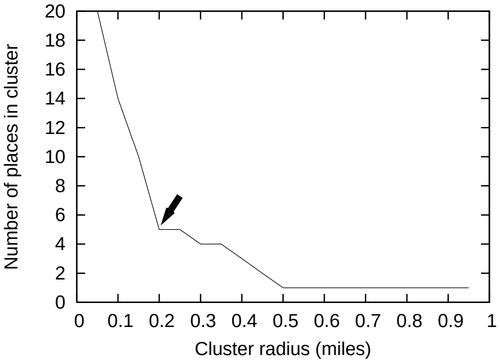

Figure 5: *Number of* places *found in a cluster as the cluster's radius changes. The arrow indicates the knee in the graph.*

3.2.4 Prediction

Once we have formed *locations* from all of the data, we assign each *location* a unique ID. Then, going back to the original chronological *place* list, we substitute for each *place* the ID of the *location* it belongs to. This gives us a list of *locations* the user visited, in the order that they were visited. Each *location* may be given a name by the user, if so desired (e.g., "home," "work," etc. . .).

Next, a Markov model is created for each *location*, with transitions to every other *location*. If the user never traveled between two *locations*, that transition probability is set to zero. Figure 6 shows three partial Markov models—those for “Home”, “CRB”, and “VA”. For clarity, only transitions between those three *locations* are shown. The labels on the lines between locations show the relative probabilities of each transition; for example, seventy–seven trips were made from “CRB” to other *locations*, and of those trips, sixteen were made to “Home”. Of three trips made from “VA” to other places, one was made to “Home” and one was made to “CRB”.

Note that the number of trips made from “VA” are relatively few as compared to those from “Home” and “CRB.” It is possible that “VA” is a new *location*, or one that is seldom traveled to by the user. Since there are so few trips, this node should not be used for prediction; however, the number of trips (in both directions) between “Home” and “CRB” *are* significant and *can* be used for prediction. A simple test on whether a path has sufficient evidence for prediction is to compare the path's relative frequency to the probability that the path was taken by chance.

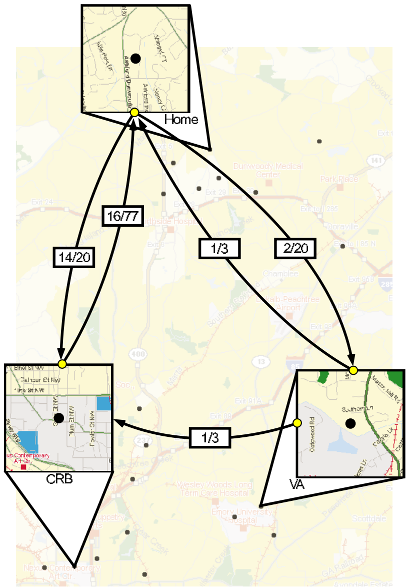

Figure 6: *Partial Markov model of trips made between home, Centennial Research Building (CRB), and Dept. of Veterans Affairs (VA). Because some paths are not shown, the ratios do not sum to 1.*

Figure 6 shows a first order Markov model; that is, one in which the probability of the next state is dependent only upon the current state. We also have the ability to create n^{th} order models, in which the probability of the next state is dependent on the current state and the previous $n-1$ states. In many cases, doing this yields more certain predictions; for example, Table 3.2.4 shows A→B with a probability of 70%, but in the second order, **B**→A→B is 81%.

Transition	Relative Frequency	Probability
$A \rightarrow B$	14/20	0.700000
$A \rightarrow B \rightarrow A$	3/14	0.214286
$A \rightarrow B \rightarrow C$	2/14	0.142857
$A \rightarrow B \rightarrow D$	3/14	0.214286
$A \rightarrow B \rightarrow E$	1/14	0.071429
$A \rightarrow B \rightarrow F$	1/14	0.071429
$A \rightarrow B \rightarrow G$	1/14	0.071429
$A \rightarrow B \rightarrow H$	1/14	0.071429
$A \rightarrow B \rightarrow I$	1/14	0.071429
$B \rightarrow A$	16/77	0.207792
$B \rightarrow A \rightarrow B$	13/16	0.812500
$B \rightarrow A \rightarrow J$	3/16	0.187500
$B \rightarrow C$	10/77	0.129870
$B \rightarrow C \rightarrow A$	6/10	0.600000
$B \rightarrow C \rightarrow K$	4/10	0.400000
$D \rightarrow B$	5/7	0.714286
$D \rightarrow B \rightarrow A$	2/5	0.400000
$D \rightarrow B \rightarrow L$	2/5	0.400000
$D \rightarrow B \rightarrow M$	1/5	0.200000

Table 1: *Probabilities for transitions in first and second order Markov models. A = “Home,” B = “CRB,” and D = “south of Tech.”*

The ability to use n^{th} order Markov models raises the question of what the appropriate order model is to use for prediction. Bhattacharya and Das have examined this question from an information theoretic standpoint [1]. In practice, a natural limitation is the quantity of data available for analysis; as shown in Table 3.2.4, even with four months of data, the number of second order transitions is relatively small. Thus, we have limited ourselves to a second order model.

4 Future Work

One shortcoming of our work thus far is lack of data on more than one user. In order to investigate multi–user applications, as well to validate our prediction methods for individuals, we need location information from several people. We intend to collect this data in June of 2002, when several members of our research group will travel to Zürich, Switzerland, for a seven–month research program. We will collect data throughout the seven months, and when we have enough data, we will begin implementing some of the multi–user applications described in Section 2.2.

While we have location prediction fully functioning, we have not yet implemented time prediction—that is, we can predict where someone will go next, but not when. Our next task will be to extend the Markov model to support time prediction; at the same time, we will investigate how variance in arrival and departure times can indicate the importance of events. For instance, if the user always arrives at a certain location within a fifteen–minute time period, that location may be more important than one with a one–hour variance.

One limitation of our approach to the Markov models is that changes in schedule may take a long time to be reflected in the model. For example, a college student might have a model that learned the locations of her classes for an entire semester (sixteen weeks). When the next semester started, she may have an entirely different schedule; because in our model each transition is given equal weight, it might take the entire semester for the model to be updated to correctly reflect the new information. One way this could be solved is by weighting updates to the model more heavily; we must be careful, however, to avoid unduly weighting one–time trips.

Currently, our system does not update the user models in real–time; this will become more necessary as we add more users to our system. We plan on not only allowing instant integration of location data, but allowing the users to view their models and give feedback; if the user knows her schedule has changed but that the model has not yet detected this, she can update the model as appropriate.

5 Conclusion

We have demonstrated how locations of significance can be automatically learned from GPS data at multiple scales. We have also shown a system that can incorporate these locations into a predictive model of the user's movements. In addition, we have described several potential applications of such models, including both single– and multi–user scenarios. Potentially such methodologies might be extended to other sources of context as well. Hopefully, one day such predictive models might become an integral part of intelligent wearable agents.

6 Acknowledgments

Thanks to Graham Coleman for writing visualization tools and to MapBlast (http://www.mapblast.com) for having freely available maps. Funding for this project has been provided in part by NSF career grant number 0093291.

References

[1] Amiya Bhattacharya and Sajal K. Das. Lezi-update: An information-theoretic approach to track mobile users in PCS networks. In *Mobile Computing and Networking*, pages 1–12, 1999.

[2] Jae-Hwan Chang and Leandros Tassiulas. Energy conserving routing in wireless ad–hoc networks. In *INFOCOM (1)*, pages 22–31, 2000.

[3] Andrew Csinger. *User Models for Intent–based Authoring*. PhD thesis, The University of British Columbia, Vancouver, B.C., 1995.

[4] Cybiko, Inc. *http://www.cybiko.com.*

[5] GeoStats, Inc. *http://www.geostats.com.*

[6] James M. Hudson, Jim Christensen, Wendy A. Kellogg, and Thomas Erickson. 'I'd be overwhelmed, but it's just one more thing to do:' Availability and interruption in research management. In *Proceedings of Human Factors in Computing Systems (CHI 2002)*, Minneapolis, MN, 2002.

[7] J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. *ACM Trans. on Computer Systems*, 10(1), February 1992.

[8] Natalia Marmasse and Chris Schmandt. Location–aware information delivery with ComMotion. In *HUC*, pages 157–171, 2000.

[9] Jon Orwant. Doppelgänger goes to school: Machine learning for user modeling. Master's thesis, MIT Media Laboratory, September 1993.

[10] Jörg Roth and Claus Unger. Using handheld devices in synchronous collaborative scenarios. In *HUC*, pages 187–199, 2000.

[11] Bruce Stirling. *Distraction*. Spectra, 1998.

[12] Michael Terry, Elizabeth D. Mynatt, Kathy Ryall, and Darren Leigh. Social net: Using patterns of physical proximity over time to infer shared interests. In *Proceedings of Human Factors in Computing Systems (CHI 2002)*, Minneapolis, MN, 2002.

[13] Jean Wolf, Randall Guensler, and William Bachman. Elimination of the travel diary: An experiment to derive trip purpose from GPS travel data.