Inferring High-Level Behavior from Low-Level Sensors

Donald J. Patterson, Lin Liao, Dieter Fox, and Henry Kautz

University Of Washington,
Department of Computer Science and Engineering,
Seattle, WA, USA
{djp3,liaolin,fox,kautz}@cs.washington.edu

Abstract. We present a method of learning a Bayesian model of a traveler moving through an urban environment. This technique is novel in that it simultaneously learns a unified model of the traveler's current mode of transportation as well as his most likely route, in an unsupervised manner. The model is implemented using particle filters and learned using Expectation-Maximization. The training data is drawn from a GPS sensor stream that was collected by the authors over a period of three months. We demonstrate that by adding more external knowledge about bus routes and bus stops, accuracy is improved.

1 Introduction

A central theme in ubiquitous computing is building rich predictive models of human behavior from low-level sensor data. One strand of such work concerns tracking and predicting a person's movements in outdoor settings using GPS [1–4]. But location is only one small part of a person's state. Ideally we would want to recognize and predict the high-level intentions and complex behaviors that cause particular physical movements through space. Such higher-order models would both enable the creation of new computing services that autonomously respond to a person's unspoken needs, and support much more accurate predictions about future behavior at all levels of abstraction.

This paper presents an approach to learning how a person uses different kinds of transportation in the community. We use GPS data to infer and predict a user's transportation *mode*, such as walking, driving, or taking a bus. The learned model can predict mode transitions, such as boarding a bus at one location and disembarking at another. We show that the use of such a higher-level transportation model can also increase the accuracy of location prediction, which is important in order to handle GPS signal loss or preparing for future delivery of services.

A key to inferring high-level behavior is fusing a user's historic sensor data with general commonsense knowledge of real-world constraints. Real-world constraints include, for example, that buses only take passengers on or off at bus stops, that cars are left in parking lots, and that cars and buses can only travel on streets, *etc.*. We present a unified probabilistic framework that accounts for both sensor error (in the case of GPS, loss of signal, triangulation error, or multi-path propagation error) and commonsense rules.

Although this work has broad applications to ubiquitous computing systems, our motivating application is one we call the Activity Compass, a device which helps guide a cognitively impaired person safely through the community [5]. The system notes when the user departs from a familiar routine (for example, gets on the wrong bus) and provides proactive alerts or calls for assistance. The Activity Compass is part of a larger project on building cognitive assistants that use probabilistic models of human behavior [6].

Our approach is built on recent successes in particle filters, a variant of Bayes filters for estimating the state of a dynamic system [7]. In particular we show how the notion of graph-constrained particle filtering introduced in [8] can be used to integrate information from street maps. Extensions to this technique include richer user transportation state models and multiple kinds of commonsense background knowledge. We introduce a three-part model in which a low-level filter continuously corrects systematic sensor error, a particle filter uses a switching state-space model for different transportation modes (and further for different velocity bands within a transportation mode), and a street map guides the particles through the high-level transition model of the graph structure. We additionally show how to apply Expectation-Maximization (EM) to learn typical motion patterns of humans in a completely unsupervised manner. The transition probabilities learned from real data significantly increase the model's predictive quality and robustness to loss of GPS signal.

This paper is organized as follows. In the next section, we summarize the derivation of graph-based tracking starting from the general Bayes filter, and show how it can be extended to handle transportation mode tracking. Then, in Sect. 3, we show how to learn the parameters of the tracking model using EM. Before concluding in Sect. 5, we present experimental results that show we can learn effective predictive models of transportation use behavior.

2 Tracking on a Graph

Our approach tracks a person's location and mode of transportation using street maps such as the ones being used for route planning and GPS-based car tracking. More specifically, our model of the world is a graph $G = (V, E)$ which has a set V of vertices and a set E of directed edges. Edges correspond to straight sections of roads and foot paths, and vertices are placed in the graph to represent either an intersection, or to accurately model a curved road as a set of short straight edges. To estimate the location and transportation mode of a person we apply Bayes filters, a probabilistic approach for estimating the state of a dynamic system from noisy sensor data. We will now briefly describe Bayes filters in the general case, show how to project the different quantities of the Bayes filter onto the structure represented in a graph, and then discuss our extensions to the state space model.

2.1 Bayesian Filtering on a Graph

Bayes filters address the problem of estimating the state x_t of a dynamical system from sensor measurements. Uncertainty is handled by representing all quantities involved in the estimation process using random variables. The key idea of Bayes filters is to recursively estimate the posterior probability density over the state space conditioned on the data collected so far. The data consists of a sequence of observations $z_{1:t}$ and the posterior over the state x_t at time t is computed from the previous state x_{t-1} using the following update rule (see [7, 9] for details):

$$p(x_t \,|\, z_{1:t}) \propto p(z_t \,|\, x_t) \int p(x_t \,|\, x_{t-1})\; p(x_{t-1} \,|\, z_{1:t-1}) dx_{t-1} \qquad (1)$$

The term $p(x_t \mid x_{t-1})$ is a probabilistic model of the object dynamics, and $p(z_t \mid x_t)$ describes the likelihood of making observation z_t given the location x_t.

In the context of location estimation, the state, x_t, typically describes the position and velocity of the object in 2D-space. When applying Bayesian filtering to a graph, the state of an object becomes a triple $x_t = \langle e, d, v \rangle$, where $e \in E$ denotes on which edge the object resides, d indicates the distance of the object from the start vertex of edge e, and v indicates the velocity along the edge [8]. The motion model $p(x_t \,|\, x_{t-1})$ considers that the objects are constrained to motion on the graph and may either travel along an edge, or, at the endpoint of the edge, switch to a neighboring edge. To compute the probability of motion from one edge to another, the graph is annotated with transition probabilities $p(e_j \mid e_i)$, which describe the probability that the object transits to edge e_j given that the previous edge was e_i and an edge transition took place. Without other knowledge, this probability is a uniform distribution over all neighboring edges of e_i.

Our work builds on graph-based Bayesian tracking by hierarchically extending the state model. We add a higher level of abstraction which contains the transportation information and a lower level sensor error variable. The resulting state x_t consists of the variables shown in Fig. 1. The presence of a bus stop near the person is given by the binary variable b_t, and the presence of a parking lot is modeled by c_t. The mode of transportation, denoted m_t, can take on one of three different values

$$m_t \in \{BUS, FOOT, CAR\}.$$

v_t denotes the motion velocity, and the location of the person at time t is represented by $l_t = \langle e, d \rangle$. o_t denotes the expected sensor error, which in our current model compensates for systematic GPS offsets. Finally, at the lowest level of the model, raw GPS sensor measurements are represented by gps_t.

Tracking such a combined state space can be computationally demanding. Fortunately, Bayes filters can make use of the independences between the different parts of the tracking problem. Such independences are typically displayed in a graphical model like Fig. 1. A dynamic Bayes net [10, 11], such as this one, consists of a set of variables for each time point t, where an arc from one variable to another indicates a causal influence. Although all of the links are equivalent in their causality, Fig. 1 represents causality through time with dashed arrows. In an abstract sense the network can be as large as the maximum value of t (perhaps infinite), but under the assumption that the dependencies between variables do not change over time, and that the state space conforms to the first-order Markov independence assumption, it is only necessary to represent and reason about two time slices at a time. In the figure the slices are numbered $t-1$ and t. The variables labeled gps are directly observable, and represent the position and velocity readings from the GPS sensor (where a possible value for the reading includes "loss of signal"). All of the other variables — sensor error, velocity,

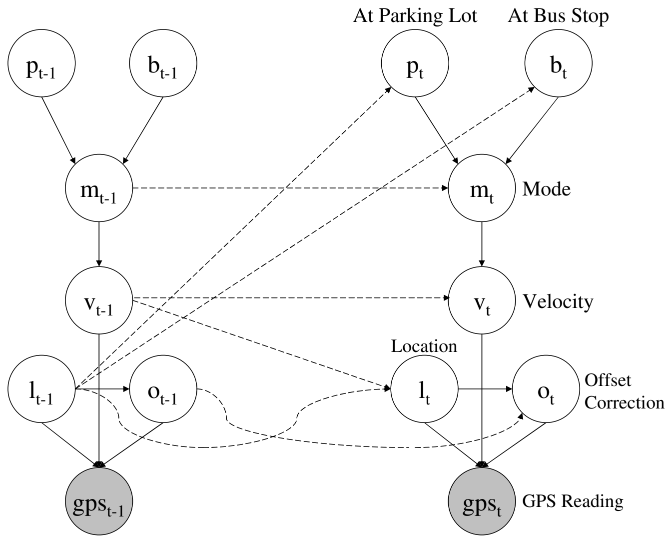

Fig. 1. Two-slice dynamic Bayes net model of the transportation domain, showing dependencies between the observed and hidden variables. Observed variables are shaded. Intra-temporal causal links are solid, inter-temporal links are dashed.

user location, mode, and the presence of a parking lot or bus stop location — are hidden variables whose values must be inferred from the raw GPS readings.

The dependencies between the nodes in Fig. 1 can be quite complex. The GPS reading at each time point is influenced by the local sensor error and the user's actual velocity and location. The location at time t only depends on the person's previous location and the motion velocity. Note that GPS is explicitly not considered to provide the true user location; urban interference, map reference point errors, GPS error and sensor failure all cause the true location to be a hidden variable. The sensor offset correction node o_t is used to reason about errors in the GPS readings which are systematic over time and location. This node maintains a probability distribution over corrections to the GPS signal that are caused by multi-path propagation error and/or dynamic satellite geometry. The node updates its belief state by comparing GPS readings to the street map to gradually adjust to local variations in signal offset.

A more complex relationship governs how the mode of transportation influences the instantaneous velocity. The influence of mode on velocity is complicated by the fact that the range of possible instantaneous velocities for each mode overlap. For example, movement at 7 km/hr may be a brisk walk or a slowly moving car or bus. To simplify the relationship between mode and velocity we model the continuous velocities using the Gaussian mixture shown in Fig. 2. A separate unsupervised Expectation-Maximization (EM) process determined the parameters of these probability densities using real velocity data. Our model assumes that velocities are drawn randomly from these Gaussians, where the probability of drawing from a particular Gaussian depends on the mode. For example, the walking mode draws a speed from the left-most cluster with probability one. In the bus mode, the person has a $1/3$ chance of being in each of the three slow-

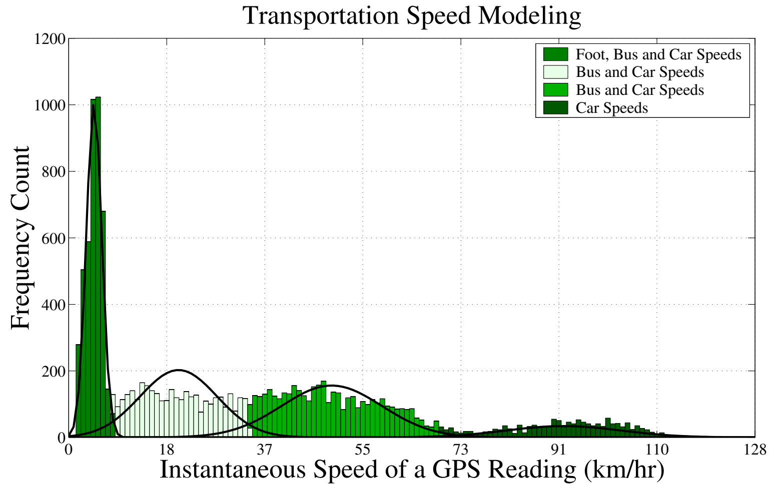

Fig. 2. Gaussian mixture model for the dependency of transportation mode on velocities. The Gaussians were learned using EM based on previously collected velocity data. The frequencies of the raw velocity values are indicated by the bins. Different transportation modes are modeled by sampling with different probability from the four Gaussians.

est velocity clusters. In our current approach, the probabilities for the Gaussians in the different transportation modes were set manually based on external knowledge. Learning the weights of the mixture components depending on the transportation mode (and eventually location) is left for future research.

In our model, the motion mode at time t only depends on the previous mode and the presence of a parking lot or bus stop. For example, the person can only get on a bus if the node b_t indicates the presence of a bus stop. The values of the bus stop and parking lot nodes depend on the location of the person, as indicated by the arrows in the model shown in Fig. 1. Learning mode and location transition probabilities is an important aspect of our approach and will be discussed in Sect. 3.

2.2 Particle Filter Based Implementation

Particle filters provide a sample-based implementation of general Bayes filters [7]. They represent posterior distributions over the state space with temporal sets, S_t, of n weighted samples:

$$S_t = \{\langle x_t^{(i)}, w_t^{(i)} \rangle \mid i = 1, \dots, n\}$$

Here each $x_t^{(i)}$ is a sample (or state), and the $w_t^{(i)}$ are non-negative numerical factors called *importance weights*, which sum up to one. Like Kalman filters, particle filters apply the recursive Bayes filter update to estimate posteriors over the state space, but unlike Kalman filters, particle filters are not restricted to unimodal posterior distributions [1]. The basic particle filter updates the posterior according to the following sampling procedure, often referred to as sequential importance sampling with re-sampling (SISR, see also [7]):

- **Sampling:** Draw n samples $x_{t-1}^{(i)}$ from the previous set and generate n new samples $x_t^{(j)}$ using the distribution
$$p(x_t \mid x_{t-1}).$$
The new samples now represent the density given by the product
$$p(x_t \mid x_{t-1})p(x_{t-1} \mid z_{1:t-1})$$
This density is the so-called *proposal distribution* used in the next step.
- **Importance sampling:** Assign each sample $x_t^{(j)}$ an importance weight according to the likelihood of the observation, z_t, given the sample,
$$w_t^{(j)} = p(z_t \mid x_t^{(j)}).$$
- **Re-sampling:** Multiply / discard samples by drawing samples with replacement according to the distribution defined through the importance weights $w_t^{(j)}$.

It can be shown that this procedure in fact approximates the Bayes filter update (1), using a sample-based representation [7, 13].

The application of particle filters to the problem of location and mode estimation using the network shown in Fig. 1 is rather straightforward. Each particle $x_t^{(i)}$ represents an instantiation of the random variables describing the transportation mode m_t, the location l_t, and the velocity v_t. The parking lot and bus stop variables p_t and b_t are extracted from each sample location l_t. Finally, o_t is determined globally for all particles by estimating the offset between GPS readings and the street map. The update steps of the particle filter can be implemented as follows. The temporal sampling step corresponds to advancing each particle according to the motion model: First the transportation mode is chosen according to the previous transportation mode and the presence of bus stops or parking lots. This gives us m_t. Then we randomly pick a velocity from the velocity model for the specific mode m_t. The velocity is used to advance the position of the person on the graph. If the sampled velocity implies a transition to another edge, the next edge e_t is drawn with probability $p(e_t \mid e_{t-1}, m_t)$ (see [8] for more information on edge transitions). After these sampling steps, the resulting states represent the predicted location, velocity, and transportation mode. The importance sampling step is implemented by weighting each sample according to the likelihood of observing the current signal from the GPS sensor given the new location of the sample. The re-sampling step of the particle filter algorithm does not have to be changed.

[1] We consider multi-hypothesis tracking to be a viable alternative to our particle filter based implementation. Multi-hypothesis tracking overcomes the restrictive assumption of the plain Kalman filter by estimating a state using multiple Kalman filters [12]. An implementation of such an approach will be part of our future research.

3 Parameter Learning

One of the advantages of modeling the world with a graph is the ability to record behavioral data about edge transitions. The discrete nature of such transitions facilitates unsupervised learning of hierarchical model parameters. We have an intuitive prior expectation of how state transitions occur between and within edges: edge transitions occur uniformly among the edge's neighbors, and mode transitions vary according to the presence of a bus stop or parking lot.

Learning in this context means adjusting the model parameters to better fit the training data, typically to better model an individual user or the environment. Learning parameters for specific individuals captures idiosyncratic motion patterns — the movements the user commonly makes, as opposed to the logically possible set of movements. Since our model also includes transportation mode, learning also means changing our prior expectations about which edges mode transitions occur on. Bus stops and parking locations are conceptual locations where mode transitions may occur. Our model enables learning of the commonly used subset of these locations, to highlight where a user frequently parks her car, for example. The learned model supports better tracking and prediction than the prior model, and is the foundation upon which high-level understanding of the user's behavior is built.

We now describe how to learn the parameters of our graph model using data collected by a person moving through the community. Our motivating application of the Activity Compass forces us to learn the transportation modes in an *unsupervised manner*. When deployed, Activity Compass users must not be required, for example, to keep a diary for several weeks of their transportation modes in order to create a supervised training set. Hence, the most obvious difficulty is that we have to learn the motion model based solely on a map and a stream of non-continuous and noisy GPS sensor data.

A general approach for solving such learning problems is the well-known Expectation-Maximization (EM) algorithm [14, 15]. In our application, EM is based on the observation that learning the model parameters would be easy *if* we knew the person's true location and transportation mode at any point in time. Unfortunately, location and transportation mode are hidden variables, *i.e.* they cannot be observed directly but have to be inferred from the raw GPS measurements. EM solves this problem by iterating between an Expectation step (E-step) and a Maximization step (M-step). In a nutshell, each E-step estimates expectations (distributions) over the hidden variables using the GPS observations along with the current estimate of the model parameters. Then in the M-step the model parameters are updated using the expectations of the hidden variables obtained in the E-step. The updated model is then used in the next E-step to obtain more accurate estimates of the hidden variables. EM theory tells us that in each iteration the estimation of the parameters will be improved and it will eventually converge to a local optimum. In the following we give a more detailed description of how to apply EM theory in our domain.

3.1 E-step:

Let Θ denote the parameters of the graph-based model we want to estimate and $\Theta^{(i-1)}$ denote the estimation thereof at the $i-1$-th iteration of the EM algorithm. The model parameters contain all conditional probabilities needed to describe the dynamic system shown in Fig. 1. The E-step estimates

$$p(x_{1:t} \mid z_{1:t}, \Theta^{(i-1)}), \tag{2}$$

i.e. the posterior distribution over the trajectories of the person given the observations and parameters updated in the previous iteration. Here $x_{1:t}$ and $z_{1:t}$ are the states and observations, respectively. Since it is not possible to find a closed-form solution for the posterior over $x_{1:t}$, we have to resort to an approximate approach [16]. Observe that when we do particle filtering using the motion model with parameter $\Theta^{(i-1)}$, the particle distribution at each time t along with the history of particles is an approximation for $p(x_{1:t} \mid z_{1:t}, \Theta^{(i-1)})$. Hence, the desired expectation can be computed using the graph-based particle filter described in Sect. 2.2. Before we give implementation details for the E-step, let us take a closer look at the M-step.

3.2 M-step:

The goal of the M-step is to maximize the expectation of $\log p(z_{1:t}, x_{1:t} \mid \Theta)$ over the distribution of $x_{1:t}$ obtained in the E-step by updating the parameter estimations. Because the distribution of $x_{1:t}$ is represented by the history of particles, the estimation of the parameters at the i-th EM iteration is computed by summing over all trajectories:

$$\begin{aligned}
\Theta^{(i)} &= \underset{\Theta}{\operatorname{argmax}} \sum_{j=1}^{n} \log p(z_{1:t}, x_{1:t}^{(j)} \mid \Theta) \\
&= \underset{\Theta}{\operatorname{argmax}} \sum_{j=1}^{n} (\log p(z_{1:t} \mid x_{1:t}^{(j)}) + \log p(x_{1:t}^{(j)} \mid \Theta)) && (3) \\
&= \underset{\Theta}{\operatorname{argmax}} \sum_{j=1}^{n} \log p(x_{1:t}^{(j)} \mid \Theta) && (4)
\end{aligned}$$

Here, n is the number of particles, $x_{1:t}^{(j)}$ is the state history of the j-th particle, and (3) follows from the independence condition

$$p(z_{1:t} \mid x_{1:t}^{(j)}, \Theta) = p(z_{1:t} \mid x_{1:t}^{(j)}),$$

i.e., observations are independent of model transition parameters if the state trajectory is known. For simplicity, we assume that all the particles have equal weight, *i.e.* after they are resampled. It is straightforward to extend our derivation to the case of different weights.

Our approach is in fact a direct extension of the Monte Carlo EM algorithm [17]. The only difference is that we allow particles to evolve with time. It has been shown that when the number of particles n is large enough, Monte Carlo EM estimation converges to the theoretical EM estimation [16].

3.3 Implementation Details

Even though EM can be used to learn all parameters Θ of the model described in Sect. 2, we are mostly interested in learning those parts of the model that describe the typical motion patterns of a user. All other parameters are fixed beforehand and not adjusted to a specific user. An advantage of this approach is that it requires much less training data than learning all parameters at once.

The motion patterns of a specific user are described by the location transitions on the graph and the mode transitions at the different locations. For the learning process, we have to initialize these probabilities to some reasonable values:

$p(e_t \mid e_{t-1}, m_{t-1})$ is the transition probability on the graph conditioned on the mode of transportation just prior to transitioning to the new edge. This conditional probability is initialized to a uniform distribution across all outgoing edges, with the exception of bus routes which have a strong bias forcing buses to follow the route (bus routes can be obtained from GIS sources such as [18]). With this exception, our model has no preference for a specific path of the person.

$p(m_t \mid m_{t-1}, e_{t-1})$ is the mode transition probability. This probability depends on the previous mode m_{t-1} and the location of the person, described by the edge e_{t-1}. For example, each person has typical locations where she gets on and off the bus. Mode transitions are initialized with commonsense knowledge (*e.g.*, one may not switch from a bus to a car without first being on foot), and with knowledge of bus stops. Parking lots are uniformly distributed across our map with no biases toward actual parking lots.

A straightforward implementation of the E-step given in (2) is to generate the expectation over state trajectories by storing the history of each particle (see [7] for a discussion). To do so, at each re-sampling phase, the history of old samples needs to be copied to the new samples [2]. Then at the last time step, we have a set of samples with their histories. At the M-step, we update the model parameters simply by counting over the particle histories. For example, to get $p(e_j \mid e_i, BUS)$, we count the number of times when a particle in BUS mode transits from edge e_i to e_j and then normalize the counts over all edges following e_i and BUS. This approach, although easy to implement, suffers from two drawbacks. First, it is not efficient. When the data log is fairly long, saving the histories for all the particles needs a large amount of space and history replication becomes slow. Second, and more importantly, since the number of samples is finite, the repetition of the re-sampling will gradually diminish the number of different histories and eventually decrease the accuracy of the particle based approximation [7].

We can overcome these problems by observing that we are only interested in learning the discrete transitions between edges and modes, *e.g.*, the probability of transiting from edge e_i to edge e_j in BUS mode. The discreteness of these transitions allows us to apply the well-known Baum-Welch algorithm [15], an EM algorithm for hidden Markov models (HMM). The Monte Carlo version of the Baum-Welch algorithm [19] performs at each iteration both a forward and a backward (in time) particle filtering

[2] Unnecessary copy operations can be avoided by using tree data structures to manage pointers describing the history of particles.

step. At each forward and backward filtering step, the algorithm counts the number of particles transiting between the different edges and nodes. To obtain probabilities for the different transitions, the counts of the forward and backward pass are normalized and then multiplied at the corresponding time slices.

To show how it works, we define:

$\alpha_t(e_t, m_t)$ is the number of particles on edge e_t and in mode m_t at time t in the *forward* pass of particle filtering.

$\beta_t(e_t, m_t)$ is the number of particles on edge e_t and in mode m_t at time t in the *backward* pass of particle filtering.

$\xi_{t-1}(e_t, e_{t-1}, m_{t-1})$ is the probability of transiting from edge e_{t-1} to e_t at time $t-1$ and in mode m_{t-1}.

$\psi_{t-1}(m_t, m_{t-1}, e_{t-1})$ is the probability transiting from mode m_{t-1} to m_t on edge e_{t-1} at time $t-1$.

A short derivation gives us [15, 19],

$$\xi_{t-1}(e_t, e_{t-1}, m_{t-1}) \propto \alpha_{t-1}(e_{t-1}, m_{t-1}) p(e_t \mid e_{t-1}, m_{t-1}) \beta_t(e_t, m_{t-1}) \tag{5}$$

and

$$\psi_{t-1}(m_t, m_{t-1}, e_{t-1}) \propto \alpha_{t-1}(e_{t-1}, m_{t-1}) p(m_t \mid m_{t-1}, e_{t-1}) \beta_t(e_{t-1}, m_t) \tag{6}$$

After we have ξ_{t-1} and ψ_{t-1} for all the t from 2 to T, we update the parameters as: [3]

$$\begin{aligned} p(e_t \mid e_{t-1}, m_{t-1}) &= \frac{\text{expected number of transitions from } e_{t-1} \text{ to } e_t \text{ in mode } m_{t-1}}{\text{expected number of transitions from } e_{t-1} \text{ in mode } m_{t-1}} \\ &= \frac{\sum_{t=2}^{T} \xi_{t-1}(e_t, e_{t-1}, m_{t-1})}{\sum_{t=2}^{T} \sum_{e_t \in Neighbors\ of\ e_{t-1}} \xi_{t-1}(e_t, e_{t-1}, m_{t-1})} \end{aligned} \tag{7}$$

and similarly

$$\begin{aligned} p(m_t \mid m_{t-1}, e_{t-1}) &= \frac{\text{expected number of transitions from } m_{t-1} \text{ to } m_t \text{ on edge } e_{t-1}}{\text{expected number of transitions from } m_{t-1} \text{ on edge } e_{t-1}} \\ &= \frac{\sum_{t=2}^{T} \psi_{t-1}(m_t, m_{t-1}, e_{t-1})}{\sum_{t=2}^{T} \sum_{m_t \in \{BUS, FOOT, CAR\}} \psi_{t-1}(m_t, m_{t-1}, e_{t-1})} \end{aligned} \tag{8}$$

The complete implementation is depicted in Table 1. As the number of particles increases, the approximation converges to the theoretical EM estimation. Fortunately, our approach is very efficient in this regard, since our model parameters are associated with the number of edges and modes in the graph, not the number of particles.

In addition to the user specific parameters our model requires the specification of other parameters, such as motion velocity and the GPS sensor model. The motion velocity is modeled as a mixture of Gaussians from which velocities are drawn at random.

[3] Usually we also need a prior number for each transition. We will not discuss how to set the prior value in this paper.

Table 1. EM-based parameter learning algorithm

Model Initialization: Initialize the model parameters $p(e_t|e_{t-1}, m_{t-1})$ and $p(m_t|m_{t-1}, e_{t-1})$.
E-step:

1. Generate n uniformly distributed samples and set time $t = 1$.
2. Perform forward particle filtering:
   (a) Sampling: generate n new samples from the existing samples using the current parameter estimation $p(e_t|e_{t-1}, m_{t-1})$ and $p(m_t|m_{t-1}, e_{t-1})$.
   (b) Importance sampling: reweight each sample based on observation z_t.
   (c) Re-sampling: multiply / discard samples according to their importance weights.
   (d) Count and save $\alpha_t(e_t, m_t)$
   (e) Set $t = t + 1$ and repeat (2a)-(2d) until $t = T$.
3. Generate n uniformly distributed samples and set $t = T$.
4. Perform backward particle filtering:
   (a) Compute backward parameters $p(e_{t-1}|e_t, m_t)$, $p(m_{t-1}|m_t, e_t)$ from $p(e_t|e_{t-1}, m_{t-1})$ and $p(m_t|m_{t-1}, e_{t-1})$
   (b) Sampling: generate n new samples from the existing samples using the backward parameter estimation.
   (c) Importance sampling: reweight each sample based on observation z_t.
   (d) Re-sampling: multiply / discard samples according to their importance weights.
   (e) Count and save $\beta_t(e_t, m_t)$
   (f) Set $t = t - 1$ and repeat (4b)-(4e) until $t = 1$.

M-step

1. Compute $\xi_{t-1}(e_t, e_{t-1}, m_{t-1})$ and $\psi_{t-1}(m_t, m_{t-1}, e_{t-1})$ using (5) and (6) and then normalize.
2. Update $p(e_t|e_{t-1}, m_{t-1})$ and $p(m_t|m_{t-1}, e_{t-1})$ using (7) and (8).

Loop Repeat E-step and M-step using updated parameters until model converges.

The probabilities of the mixture components depend on the current motion mode and can be learned beforehand using data labeled with the correct mode of motion. We use a standard model to compute the likelihood $p(z_t \mid x_t)$ of a GPS sensor measurement z_t given the location x_t of the person [1].

4 Experiments

Our test data set consists of logs of GPS data collected by one of the authors. The data contains position and velocity information collected at 2-10 second intervals during periods of time in which the author was moving about outdoors. This data was hand labeled with one of three modes of transportation: foot, bus, or car. This labeling was useful for validating the results of our unsupervised learning, but was not used by the EM learning process.

From this data set, we chose 29 episodes representing a total of 12 hours of logs. This subset consists of all of portions of the data set which were bounded by GPS signal loss, *i.e.* had no intermediate loss of signal of more than 30 seconds, and which

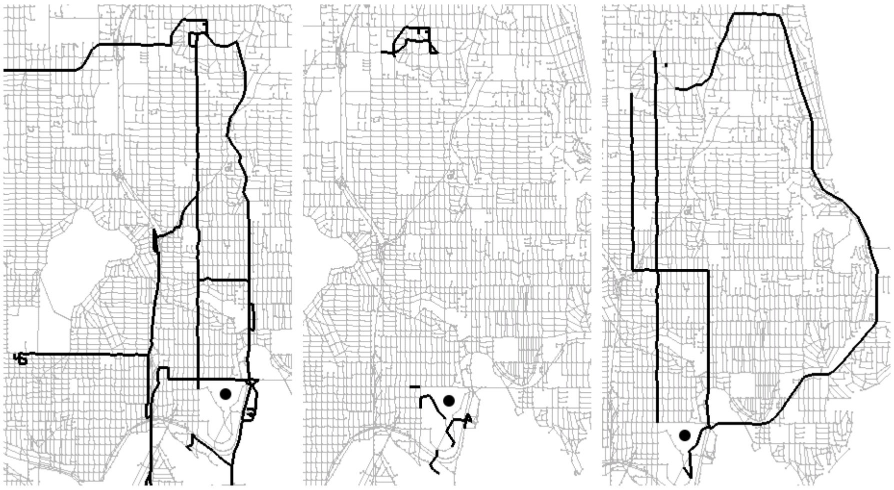

Fig. 3. Car (left), Foot (middle), and Bus (right) training data used for experiments. The black dot is a common map reference point on the University of Washington campus.

contained a change in the mode of transportation at some point in the episode. These episodes were divided chronologically into three groups which formed the sets for three-fold cross-validation for our learning. Fig. 3 shows one of the cross-validation groups used for training. The street map was provided by the US Census Bureau [20] and the locations of the bus stops come from the King County GIS office [18].

4.1 Mode Estimation and Prediction

One of the primary goals of our approach is learning a motion model that predicts transportation routes, conditioned on the mode of transportation. We conducted an experiment to validate our models' ability to correctly learn the mode of transportation at any given instant. For comparison we also trained a decision tree model using supervised learning on the data [21]. We provided the decision tree with two features: the current velocity and the standard deviation of the velocity in the previous sixty seconds. Using the data annotated with the hand-labeled mode of transportation, the task of the decision tree was to output the transportation mode based on the velocity information. We used three-fold cross-validation groups to evaluate all of the learning algorithm. The results are summarized in the first row of Table 2. The first result indicates that 55% of the time the decision tree approach was able to accurately estimate the current mode of transportation on the test data. Next, we used our Bayes filter approach without learning the model parameters, *i.e.* with uniform transition probabilities. Furthermore, this model did not consider the locations of bus stops or bus routes (we never provided parking locations to the algorithm). In contrast to the decision tree, the Bayes filter algorithm integrates information over time, thereby increasing the accuracy to 60%. The benefit of additionally considering bus stops and bus routes becomes obvious in the next row, which shows a mode accuracy of 78%. Finally, using EM to learn the model parameters increases the accuracy to 84% of the time, on test data not used for training. Note that this value is very high given the fact that often a change of transportation mode cannot be detected instantaneously.

Table 2. Mode estimation quality of different algorithms.

Model	Cross-Validation Prediction Accuracy
Decision Tree with Speed and Variance	55%
Prior Graph Model, w/o bus stops and bus routes	60%
Prior Graph Model, w/ bus stops and bus routes	78%
Learned Graph Model	84%

A similar comparison can be done looking at the techniques' ability to predict not just instantaneous modes of transportation, but also *transitions* between transportation modes. Table 3 shows each technique's accuracy in predicting the qualitative change in transportation mode within 60 seconds of the actual transition — for example, correctly predicting that the person got off a bus. Precision is the percentage of time when the algorithm predicts a transition that an actual transition occurred. Recall is the percentage of real transitions that were correctly predicted. Again, the table clearly indicates the superior performance of our learned model. Learning the user's motion patterns significantly increases the precision of mode transitions, *i.e.* the model is much more accurate at predicting transitions that will actually occur.

Table 3. Prediction accuracy of mode transition changes.

Model	Precision	Recall
Decision Tree with Speed and Variance	2%	83%
Prior Graph Model, w/o bus stops and bus routes	6%	63%
Prior Graph Model, w/ bus stops and bus routes	10%	80%
Learned Graph Model	40%	80%

An example of the modes of transportation predicted after training on one cross-validation set is shown in Fig. 4.

4.2 Location Prediction

The location prediction capabilities of our approach are illustrated in Fig. 5 and 6. In Fig. 5, the learned model was used to predict the location of the person into the future. This was done by providing the ground truth location and transportation mode to the algorithm and then predicting the most likely path based on the transition probabilities learned from the training data. The figure shows the percentage of trajectories that were predicted correctly, given different prediction horizons. Prediction length was measured in city blocks. For example, in 50% of the cases, the location of the person

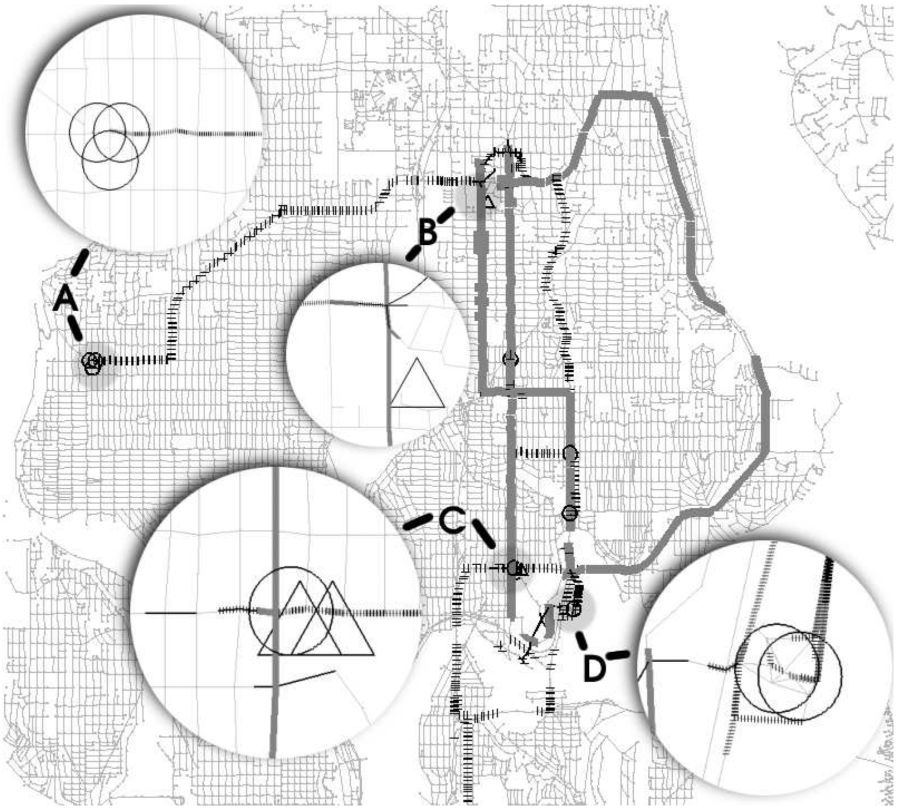

Fig. 4. This map shows the learned transportation behavior based on one cross-validation set containing nineteen episodes. Shown are only those edges and mode transitions which the learned model predicts with high probabilities. Thick gray lines indicate learned bus routes, thin black lines indicate learned walking routes, and cross-hatches indicate learned driving routes. Circles indicate parking spots, and the triangles show the subset of bus stops for which the model learned a high probability transition on or off the bus. There are four call-outs to show detail. **(A)** shows a frequently traveled road ending in three distinct parking spaces. This route and the parking spots indicate the correctly learned car trips between the author's home and church. **(B)** shows a frequently traveled foot route which enters from the northeast, ending at one of the frequently used bus stops of the author. The main road running east-west is an arterial road providing access to the highway for the author. **(C)** shows an intersection at the northwest of the University of Washington campus. There are two learned bus stops. The author frequently takes the bus north and south from this location. This is also a frequent car drop off point for the author, hence the parking spot indication. Walking routes extend west to a shopping area and east to the campus. **(D)** shows a major university parking lot. Foot traffic walks west toward campus.

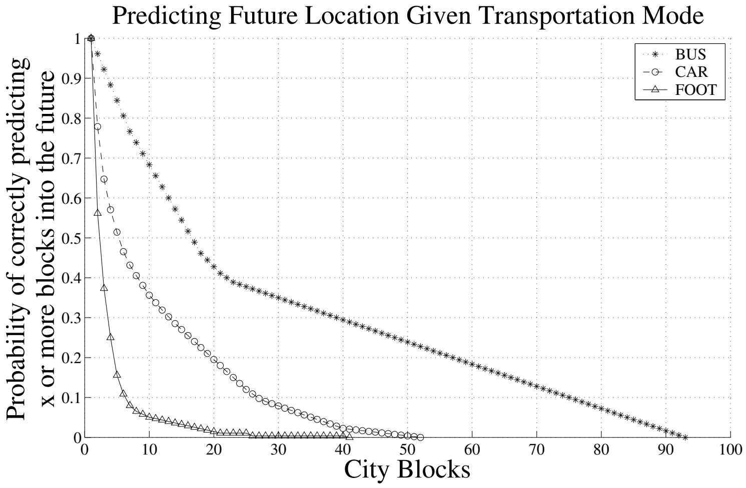

Fig. 5. Location prediction capabilities of the learned model.

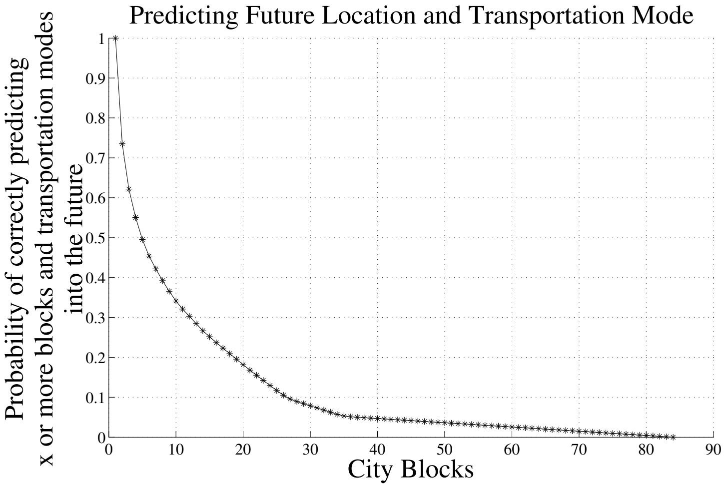

Fig. 6. Location and mode prediction capabilities of the learned model.

was predicted correctly for 17 blocks when the person was on the bus. In 30% of the cases, the prediction was correct for 37 blocks, and 75 blocks were predicted correctly in 10% of the cases. Note that the linear drop in bus route prediction probability is due to the fact that the data contained several correctly predicted episodes of a 92 block long bus trip. Obviously, long term distance prediction is much less accurate when a person walks. This is due to the higher variability of walking patterns and the fact that people typically do not walk for many city blocks, thereby making a long term prediction impossible.

In Fig. 6, the learned model was used to predict both the location and the transportation mode of the person into the future. This was done by providing the ground truth location to the algorithm and then predicting the most likely path and sequence of transportation mode switches based on the transition probabilities learned from the training data. The graph shows that in 50% of the cases, the model is able to correctly predict the motion and transportation mode of the person for five city blocks. This result is extremely promising given that the model was trained and tested on subsets of 29 episodes.

5 Conclusions and Future Work

The work presented in this paper helps lay the foundation for reasoning about high-level descriptions of human behavior using sensor data. We showed how complex behaviors such as boarding a bus at a particular bus stop, traveling, and disembarking can be recognized using GPS data and general commonsense knowledge, without requiring additional sensors to be installed throughout the environment. We demonstrated that good predictive user-specific models can be learned in an unsupervised fashion.

The key idea of our approach is to apply a graph-based Bayes filter to track a person's location and transportation mode on a street map annotated with bus route information. The location and transportation mode of the person is estimated using a particle filter. We showed how the EM algorithm along with frequency counts from the particle filter can be used to learn a motion model of the user. A main advantage of this unsupervised learning algorithm is the fact that it can be applied to raw GPS sensor data.

The combination of general knowledge and unsupervised learning enables a broad range of "self-customizing" applications, such as the Activity Compass mentioned in Sect. 1. Furthermore, it is straightforward to adopt this approach for "life long" learning: the user never needs to explicitly instruct the device, yet the longer the user carries the device the more accurate its user model becomes.

Our current and future research extends the work described in this paper in a number of directions, including the following:

1. *Making positive use of negative information.* Loss of GPS signal during tracking causes the probability mass to spread out as governed by the transition model. We have seen that learning significantly reduces the rate of spread. In some cases, however, loss of signal can actually be used to tighten the estimation of the user's location. In particular, most buildings and certain outdoor regions are GPS dead-zones. If signal is lost when entering such an area, and then remains lost for a significant period of time while the GPS device is active, then one can strengthen the probability that the user has not left the dead-zone area.
2. *Learning daily and weekly patterns.* Our current model makes no use of absolute temporal information, such as the time of day or the day of the week. Including such variables in our model will improve tracking and prediction of many kinds of common life patterns, such as the fact that the user travels towards his place of work on weekday mornings.
3. *Modeling trip destination and purpose.* The work described in this paper segments movement in terms of transitions at intersections and between modes of transportation. At a higher level of abstraction, however, movement can be segmented in

terms of trips that progress from a location where some set of activities take place (such as home) to a location where a different class of activities take place (such as the office). A single trip between activity centers can involve several shifts between modes of transportation. By learning trip models we expect to be able to increase the accuracy of predictions. More significantly, trip models provide a way to integrate other sources of high-level knowledge, such as a user's appointment calendar.
4. *Using relational models to make predictions about novel events.* A significant limitation of our current approach is that useful predictions cannot be made when the user is in a location where she has never been before. However, recent work on relational probabilistic models [22, 23] develops a promising approach where predictions can be made in novel states by smoothing statistics from semantically similar states. For example, such a model might predict that the user has a significant chance of entering a nearby restaurant at noon even if there is no history of the user patronizing that particular restaurant.

Acknowledgment

This work has partly been supported by the NSF under grant numbers IIS-0225774 and IIS-0093406, by AFRL contract F30602–01–C–0219 (DARPA's MICA program), and the Intel corporation.

References

1. Hightower, J., Borriello, G.: Location systems for ubiquitous computing. Computer **34** (2001) IEEE Computer Society Press.
2. Bonnifait, P., Bouron, P., Crubillé, P., Meizel, D.: Data fusion of four ABS sensors and GPS for an enhanced localization of car-like vehicles. In: Proc. of the IEEE International Conference on Robotics & Automation. (2001)
3. Cui, Y., Ge, S.: Autonomous vehicle positioning with GPS in urban canyon environments. In: Proc. of the IEEE International Conference on Robotics & Automation. (2001)
4. Ashbrook, D., Starner, T.: Learning significant locations and predicting user movement with gps. In: International Symposium on Wearable Computing, Seattle, WA (2002)
5. Patterson, D., Etzioni, O., Fox, D., Kautz, H.: The Activity Compass. In: Proceedings of UBICOG '02:First International Workshop on Ubiquitous Computing for Cognitive Aids. (2002)
6. Kautz, H., Arnstein, L., Borriello, G., Etzioni, O., Fox, D.: The Assisted Cognition Project. In: Proceedings of UbiCog '02: First International Workshop on Ubiquitous Computing for Cognitive Aids, Gothenberg, Sweden (2002)
7. Doucet, A., de Freitas, N., Gordon, N., eds.: Sequential Monte Carlo in Practice. Springer-Verlag, New York (2001)
8. Liao, L., Fox, D., Hightower, J., Kautz, H., Schulz, D.: Voronoi tracking: Location estimation using sparse and noisy sensor data. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems. (2003)
9. Bar-Shalom, Y., Li, X.R., Kirubarajan, T.: Estimation with Applications to Tracking and Navigation. John Wiley (2001)
10. Dean, T., Kanazawa, K.: Probabilistic temporal reasoning. In: Proc. of the National Conference on Artificial Intelligence. (1988)

11. Murphy, K.: Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, UC Berkeley, Computer Science Division (2002)
12. Bar-Shalom, Y., Li, X.R.: Multitarget-Multisensor Tracking: Principles and Techniques. Yaakov Bar-Shalom (1995)
13. Del Moral, P., Miclo, L.: Branching and interacting particle systems approximations of Feynman-Kac formulae with applications to non linear filtering. In: Seminaire de Probabilites XXXIV. Number 1729 in Lecture Notes in Mathematics. Springer-Verlag (2000)
14. Bilmes, J.: A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report ICSI-TR-97-021, University of Berkeley (1998)
15. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. In: Proceedings of the IEEE, IEEE (1989) IEEE Log Number 8825949.
16. Levine, R., Casella, G.: Implementations of the Monte Carlo EM algorithm. Journal of Computational and Graphical Statistics **10** (2001)
17. Wei, G., Tanner, M.: A Monte Carlo implementation of the EM algorithm and the poor mans data augmentation algorithms. Journal of the American Statistical Association **85** (1990)
18. County, K.: Gis (graphical information system). http://www.metrokc.gov/gis/mission.htm (2003)
19. Thrun, S., Langford, J., Fox, D.: Monte Carlo hidden Markov models: Learning nonparametric models of partially observable stochastic processes. In: Proc. of the International Conference on Machine Learning. (1999)
20. Bureau, U.C.: Census 2000 tiger/line data. http://www.esri.com/data/download/census2000-tigerline/ (2000)
21. Mitchell, T.: Machine Learning. McGraw-Hill (1997)
22. Anderson, C., Domingos, P., Weld, D.: Relational markov models and their application to adaptive web navigation. In: Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining, ACM Press (2002) 143–152 Edmonton, Canada.
23. Sanghai, S., Domingos, P., Weld, D.: Dynamic probabilistic relational models. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann (2003) Acapulco, Mexico.