# Learning and Inferring Transportation Routines

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

*Department of Computer Science and Engineering*
*University Of Washington, Seattle WA 98195, USA*

## 1 Introduction

The mass commercialization of GPS (global positioning system) technology has led to low-cost consumer-grade positioning devices and a subsequent demand for commercial applications that take advantage of this information. But localization based on immediate sensor data is only one small part of a user's spatial context. In this paper we describe a system that creates a probabilistic model of a user's daily movement *patterns* using unsupervised learning from raw GPS data. The model allows one to:

- Robustly track and predict a user's location even with loss of GPS signals or in the presence of other sources of noise;
- Infer a user's mode of transportation (*i.e.*, traveling by foot, car, or bus) and predict when and where she will change modes;
- Infer the locations of transportation destinations, such as a home or workplace;
- Predict her future movements, both in the short term (Will the user turn left at the next street corner?) and in terms of distant goals (Is she going to work?);
- Infer when a user has deviated from her ordinary routine in a way that may indicate that she has made an error, such as failing to get off her bus at her usual stop on the way home;
- And in some cases explicitly infer that a user has made an error, such as boarding the wrong bus to get to the doctor.

We are motivated in this work by the development of personal guidance systems to help cognitively-impaired individuals move safely and independently

*Email address:* {liaolin,djp3,fox,kautz}@cs.washington.edu (Lin Liao, Donald J. Patterson, Dieter Fox, Henry Kautz).

through their community. This technology also supports many other applications, such as *customized* "just in time" information services for presenting relevant bus schedules and traffic information based on routes.

Our approach is based on a hierarchical Markov model trained to model a user with data collected by a small portable GPS unit. The model is compactly represented by a dynamic Bayesian network, and inference is efficiently performed using Rao-Blackwellised particle filtering both for the low level sensor integration and for the higher levels of the hierarchical model.

The main research contribution in this paper is a demonstration of efficient inference and the development of unsupervised learning algorithms for the hierarchical predictive models of transportation routines. Previous work has described inference in hierarchical models [1] and learning non-hierarchical transportation models [2], but our work is the first to combine these techniques. A second research contribution is in unifying this model with previously described techniques for detecting *novel* and *erroneous* behavior [3].

This paper is organized as follows. In the next section, we provide an overview of the hierarchical activity model, followed by a description of inference and learning algorithms. Then we present experimental results and an end-to-end implementation of these techniques called "Opportunity Knocks." Finally, we conclude with a discussion of related work.

## 2 Hierarchical Activity Model

We estimate a person's activities using the three level dynamic Bayesian network model shown in Fig. 1. The individual nodes in such a temporal graphical model represent different parts of the state space and the arcs indicate dependencies between the nodes [4,5]. Temporal dependencies are represented by arcs connecting the two time slices $k-1$ and $k$. The highest level of the model, the novelty mode, indicates whether the user is currently behaving normally, doing something novel, or making a transportation error. The second level of the model represents two concepts: the person's next goal (*e.g.* her work place) and the user's current trip segment. A trip segment is a route with a single transportation mode, which can be concatenated with other trip segments to form a plan for transiting between goals. The person's instantaneous transportation mode, location, and velocity are estimated from the GPS sensor measurements at the lowest level of the model.

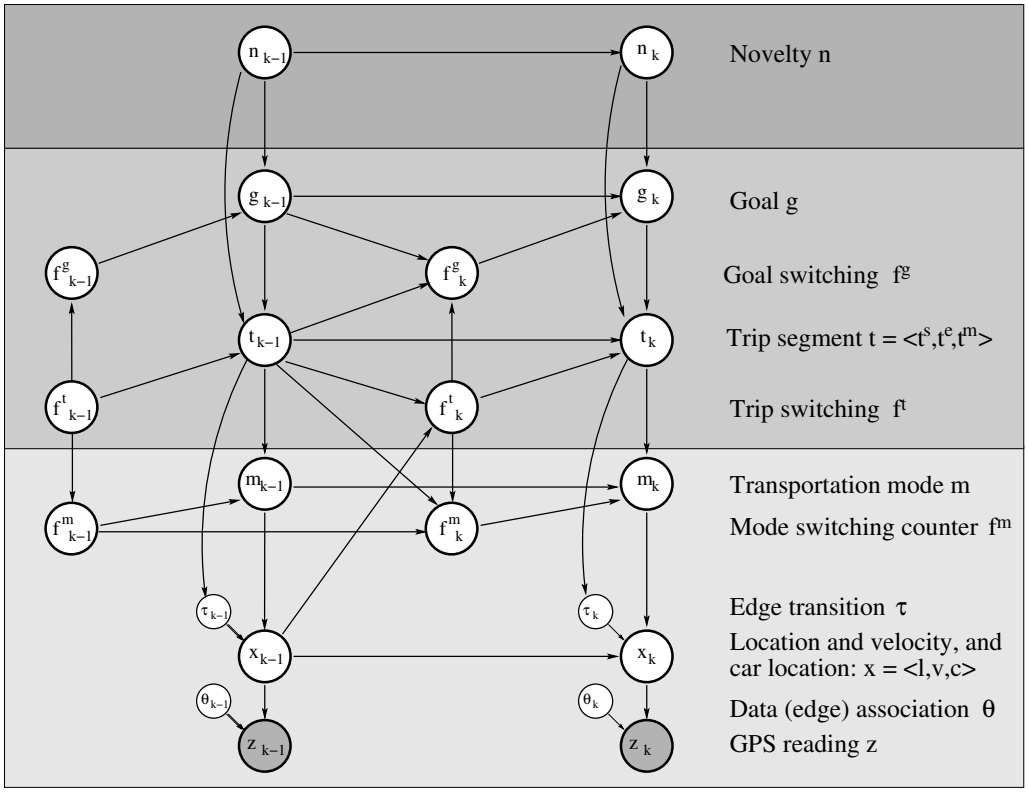



Fig. 1. Hierarchical activity model representing a person's outdoor movements during everyday activities. The upper level is used for novelty detection, and the middle layer estimates the user's goal and the trip segments he or she is taking to get there. The lowest layer represents the user's mode of transportation, speed, and location. Two time slices, $k$ and $k-1$, are shown.

### *2.1 Locations and transportation modes*

We denote by $x_k = \langle l_k, v_k, c_k \rangle$ the location ($l_k$) and velocity ($v_k$) of the person, and the location of the person's car ($c_k$) (subscripts $k$ indicate discrete time). As we will describe in the next section, locations and velocities are estimated on a graph structure representing a street map. GPS sensor measurements, $z_k$, are generated by the person carrying a GPS sensor. Since measurements are given in continuous $xy$-coordinates, they have to be "snapped" onto an edge in the graph structure. The edge to which a specific measurement is "snapped" to is estimated by the association variable $\theta_k$. The location of the person at time $k$ depends on his previous location, $l_{k-1}$, the velocity, $v_k$, and the transition, $\tau_k$. Vertex transitions $\tau$ model the decision a person makes when moving over a vertex in the graph, corresponding to an intersection on the street map; for example, whether the user will turn right, left, or not at all when transiting a street intersection.

The transportation mode $m_k$ can take on four different values $BUS$, $FOOT$, $CAR$, and $BUILDING$. Similar to [2], these modes influence the velocity,

which is picked from a Gaussian mixture model. For example, the $FOOT$ mode draws velocities only from the Gaussian representing walking speeds, while $BUS$ and $CAR$ modes draw velocities from Gaussian distributions representing both high and low velocities. $BUILDING$ is a special mode that occurs only when the GPS signal is lost for significantly long time, corresponding to the assumption that the person is indoors and not moving. Finally, the location of the car only changes when the person is in the car and thus in $CAR$ mode, in which case the car location is set to the person's location. If the person is not in $CAR$ mode, the car's location affects whether a person can switch into the $CAR$ mode. (For these experiments we assume that the user only rides in her own car. The approach could be generalized to distinguish "car travel as driver" and "car travel as passenger".)

### *2.2 Trip segments*

A trip segment is defined by its start location, $t_k^s.\texttt{start}$, end location, $t_k.\texttt{end}$, and the mode of transportation, $t_k^m$, the person uses during the segment. For example, a trip segment models information such as "the user boards the bus at location $t_k^s.\texttt{start}$, rides the bus to location $t_k.\texttt{end}$, and then debarks". In addition to transportation mode, a trip segment predicts the route on which the person gets from $t_k^s.\texttt{start}$ to $t_k.\texttt{end}$. This route is not specified through a deterministic sequence of edges on the graph, but rather through transition probabilities on the graph. These probabilities provide a prediction of the person's change in direction when reaching an intersection, or equivalently, when crossing a vertex in the graph. This dependency is indicated by the arc from $t_k$ to $\tau_k$.

The transfer between modes and trip segments is handled by the switching nodes $f_k^m$ and $f_k^t$, respectively. The Boolean trip switching node $f_k^t$ is set to true whenever the person reaches the end location $t_k.\texttt{end}$ of the current trip segment. In this case, the trip segment is allowed to switch with the constraint that the start location of the next segment is identical to the end location of the current segment. The next trip segment is chosen according to the segment transition conditioned on the current goal $g_k$. Once the next trip segment is active, the person still has to change mode of transportation. This does not happen instantaneously, since, for example, a person has to wait for the bus even though he already reached the bus stop (and thus entered the bus trip segment). This semi-Markov property of delayed mode switching is modeled by the node $f_k^m$, which is a counter that measures the time steps until the next transportation mode is entered. The counter is initialized by the next trip segment, then decremented until it reaches a value of zero, which triggers the mode switch. ⋆

⋆ This semi-Markov mode switch may be handled more efficiently by using fixed

### *2.3 Goals*

A goal represents the current target location of the person. Goals include locations such as the person's home, work place, the grocery store, and locations of friend's homes. The goal of the person can only change when the person reaches the end of a trip segment. This is facilitated by a goal switching node $f_k^g$ which is true only when the trip switching node $f_k^t$ is true and the end of the current trip segment $t_k$ is identical to the goal $g_k$. If the goal switches, the next goal is chosen according to a learned goal-to-goal transition model.

### *2.4 Novelty*

At the highest level is a boolean variable $n_k$, indicating whether a user's behavior is consistent with historical patterns. Different values of $n_k$ instantiate different parameters for the lower part of the model.

When a user is behaving normally, $n_k = False$, the hierarchical model functions as described up to this point and the parameters are trained using historical data. When a user's behavior is novel, $n_k = True$, the goal and the trip segment are set to a distinguished value "UNKNOWN" and as a consequence the parameters of the lowest layer of the model (*i.e.*, transportation mode transitions and edge transitions) are switched to their untrained values: An unknownValue goal and trip segment does not provide any information about the direction that a user will go when she gets to an intersection, or ways in which she will change modes of transportation. The model that is instantiated in this way is referred to as a *flat* model because there is no influence of high level model elements on the inference at the street level. A flat model respects basic intuitions about changing transportation modes and moving on a street map, but is not influenced by long-term goals such as where and how the user is getting home from work. Instead the flat model utilizes straightforward Markov transition probabilities in the lowest level of Fig. 1 analogous to [2].

## 3 Inference

After we have defined the structure of the model and assumed values for the transition parameters (parameter estimation will be discussed in the next section), we must develop efficient algorithms to infer the distribution of hidden

lag smoothing and observing when significant velocity changes are made.

variables at real time given a sequence of GPS readings. In this section, we first focus on the estimation of locations and modes of transportation. For simplicity, we present a version of this algorithm that does not apply switching nodes and counters in order to model waiting times. These concepts will be discussed in the context of the hierarchical model, which additionally estimates a person's high level goals, trip segments, and novelty.

### *3.1 Flat Model*

First we will explain the estimation of locations and transportation modes using a *flat* model. This is the model shown in Fig. 1 with the top two levels removed. A key point of our location tracking is that we constrain all locations to be on a street map. Thus, locations can be represented using a directed graph $G = (V, E)$ whose edges correspond to streets or footpaths and whose vertices correspond to intersections. A location is determined by an edge and distance pair. The edge corresponds to a street and the distance is the distance from the start vertex of the edge. Because street maps have a complex structure and the estimation problem contains a variety of continuous (locations, velocities) and discrete states (edge transition, transportation mode, *etc.*), exact inference is intractable. Instead, we rely on Rao-Blackwellised particle filters for inference in the flat model [6,7]. In our case, the Rao-Blackwellised particle filter combines particle filters with Kalman filters, which we will in turn.

#### *3.1.1 Preliminaries*

*Particle Filters*

Particle filters represent posterior distributions over the state space with temporal sets, $S_k$, of $n$ weighted samples:

$$S_k \;=\; \{s_k^{(i)}, w_k^{(i)} \mid 1 \leq i \leq N\}$$

Here $s_k^{(i)}$ is a sample (or state), and $w_k^{(i)}$ is a non-negative numerical factor called an *importance weight*. 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 [8,9]).

- **Sampling:** Draw $n$ samples $s_{k-1}^{(i)}$ from the previous set and generate $n$ new samples $s_k^{(i)}$ using the *proposal distribution* $q(s_k \mid s_{k-1}^{(i)}, z_k)$
- **Importance sampling:** Assign each sample an importance weight $w_k^{(i)}$ as

$$w_k^{(i)} \;\propto\; w_{k-1}^{(i)} \; \frac{p(z_k \mid s_k^{(i)}) p(s_k^{(i)} \mid s_{k-1}^{(i)})}{q(s_k^{(i)} \mid s_{k-1}^{(i)}, z_k)} \tag{1}$$

- **Re-sampling:** Multiply / discard samples by drawing samples with replacement according to the distribution defined through the importance weights $w_k^{(i)}$.

It is often convenient to choose the proposal distribution to be the prediction as

$$q(s_k \mid s_{k-1}^{(i)}, z_k) \;=\; p(s_k \mid s_{k-1}^{(i)}) \tag{2}$$

By substituting (2) into (1), we get

$$w_k^{(i)} \;\propto\; w_{k-1}^{(i)} \; p(z_k \mid s_k^{(i)}) \tag{3}$$

*Kalman Filters*

Kalman filtering [10,9] is another widely-used technique for state tracking. It approximates beliefs by a unimodal Gaussian representation, $\mathcal{N}(s_k; \mu_k, \Sigma_k)$, where $\mu_k$ and $\Sigma_k$ are the mean and the variance, respectively. At each time step, Kalman filters update these Gaussian beliefs by two steps: a *prediction step* followed by a *correction step*.

In the prediction step, Kalman filters predict the state distribution based on the previous belief $\mathcal{N}(s_{k-1}; \mu_{k-1}, \Sigma_{k-1})$ and the system dynamics $p(s_k \mid s_{k-1})$. Under the assumption that the state at time $k$ is a linear function of the previous state with additive Gaussian noise, the system dynamics can be represented by another Gaussian $\mathcal{N}(s_k; A_t s_{k-1}, R_t)$, where $A_t$ is the dynamics matrix and $R_t$ is the covariance of the Gaussian noise. The prediction is a convolution of these two Gaussians and thus the result is also a Gaussian over the state space:

$$\mathcal{N}(s_k; \widehat{\mu}_k, \widehat{\Sigma}_k) = \int \mathcal{N}(s_k; A_t s_{k-1}, R_t)\, \mathcal{N}(s_{k-1}; \mu_{k-1}, \Sigma_{k-1})\, \mathrm{d}s_{k-1} \tag{4}$$

Here, $\langle \widehat{\mu}_k, \widehat{\Sigma}_k \rangle$ denote the parameters defining the *predictive belief* at time $k$, that is, the belief before the observation $z_k$ is considered. The parameters can be computed in closed form as

$$\begin{aligned} \widehat{\mu}_k &= A_t \mu_{k-1} \\ \widehat{\Sigma}_k &= A_t \Sigma_{k-1} A_t^T + R_t. \end{aligned} \tag{5}$$

In the correction step, the most recent measurement $z_k$ is used to adjust the prediction $\mathcal{N}(s_k; \widehat{\mu}_k, \widehat{\Sigma}_k)$. If we assume that sensor measurements $z_k$ are linear functions of the state $s_k$, with added Gaussian noise, then the sensor model $p(z_k \mid s_k)$ is the Gaussian $\mathcal{N}(z_k; C_k s_k, Q_k)$. Here the matrix $C_k$ maps states to observations and $Q_t$ is the covariance of the observation noise. The posterior belief is again a Gaussian

$$\mathcal{N}(s_k; \mu_k, \Sigma_k) \propto \mathcal{N}(z_k; C_t s_k, Q_k)\, \mathcal{N}(s_k; \widehat{\mu}_k, \widehat{\Sigma}_k) \tag{6}$$

with

$$\begin{aligned} \mu_k &= \widehat{\mu}_k + K_k\, (z_k - C_t\, \widehat{\mu}_k) \\ \Sigma_k &= (1 - K_k C_k)\, \widehat{\Sigma}_k \end{aligned} \tag{7}$$

where $K_k = \widehat{\Sigma}_k C_k^T / (C_k \widehat{\Sigma}_k C_k^T + Q_k)$ is the so-called *Kalman gain*.

### *3.1.2 Rao-Blackwellized Particle Filter for Estimation in the Flat Model*

The main advantage of Kalman filters is their computational efficiency. This efficiency, however, comes at the cost of restricted representation power because Kalman filters only apply to (approximately) linear systems that can be described by unimodal distributions. While particle filters can be applied to arbitrary, non-linear systems, they are less efficient than Kalman filters. The key idea of Rao-Blackwellized particle filters (RBPF) is to combine both representations, thereby leveraging the efficiency of Kalman filters and the representational power of particle filters. RBPFs have been applied with great success to various state estimation problems, including robot mapping [11,12], tracking [13–15], and system monitoring [16,17].

In our approach, we estimate the posterior over the location of the person and the car, and the complete histories of the other parts of the state space, conditioned on $z_{1:k}$, the sequence of GPS measurements observed so far. As we will see, the estimation of these histories is necessary for the derivation of the filter, but implementations of the RBPF typically keep track of the current state only. The following factorization of the posterior forms the basis for our Rao-Blackwellised particle filter:

$$\begin{aligned} & p(c_k, l_k, m_{1:k}, v_{1:k}, \tau_{1:k}, \theta_{1:k} \mid z_{1:k}) \\ &= p(c_k \mid l_k, m_{1:k}, v_{1:k}, \tau_{1:k}, \theta_{1:k}, z_{1:k}) \\ &\quad p(l_k \mid m_{1:k}, v_{1:k}, \tau_{1:k}, \theta_{1:k}, z_{1:k})\, p(m_{1:k}, v_{1:k}, \tau_{1:k}, \theta_{1:k} \mid z_{1:k}) \end{aligned} \tag{8}$$

This factorization separates the state space of our estimation problem into three parts. From right to left, the first part contains histories over the trans-

portation mode $m_{1:k}$, velocity $v_{1:k}$, edge transition $\tau_{1:k}$, and edge association $\theta_{1:k}$, which are represented by the samples in a particle filter. The second part is the location of the person $l_k$ on the graph, which is estimated using Kalman filters conditioned on the samples. The third part represents the car location which, as we will show, is a function of the person's location.

Each particle of the flat RBPF has the following form:

$$s_k^{(i)} = \left\langle \langle \xi_k^{(i)}, \Xi_k^{(i)} \rangle, \langle \mu_k^{(i)}, \Sigma_k^{(i)} \rangle, m_{1:k}^{(i)}, v_{1:k}^{(i)}, \tau_{1:k}^{(i)}, \theta_{1:k}^{(i)}, c_{1:k}^{(i)} \right\rangle, \tag{9}$$

where $\langle \xi_k^{(i)}, \Xi_k^{(i)} \rangle$ and $\langle \mu_k^{(i)}, \Sigma_k^{(i)} \rangle$ represent the mean and variance of the car location and person location estimates, respectively. Here we present a memory efficient version of the RBPF algorithm that only stores the most recent states. Whenever a new GPS measurement arrives, the RBPF draws a particle $s_{k-1}^{(i)}$ from the previous sample set. The updated particles are then generated in three steps, evaluating (8) from right to left: First, the state histories are expanded by sampling the most recent states. Second, the person's location estimate is updated using a Kalman filter update conditioned on the measurement and the sampled values. Finally, the car location is updated conditioned on these estimates.

*Sampling Step*

The updated histories $m_{1:k}^{(i)}, v_{1:k}^{(i)}, \tau_{1:k}^{(i)}$, and $\theta_{1:k}^{(i)}$ are generated by expanding $s_{k-1}^{(i)}$'s histories via sampling the states at time $k$ and attaching them to the existing histories. First, the transportation mode $m_k^{(i)}$ is sampled from $p(m_k^{(i)} \mid m_{k-1}^{(i)}, \mu_{k-1}^{(i)}, c_{k-1}^{(i)})$ and then attached to the particle's mode history $m_{1:k-1}^{(i)}$ to generate $m_{1:k}^{(i)}$. Mode transitions take information about bus routes and the person's car into account. For instance, whenever the person's location $\mu_{k-1}^{(i)}$ was near a bus stop and the previous mode was `FOOT`, then $m_k^{(i)}$ switches to `BUS` with a small probability. Similarly, the person can only switch into the `CAR` mode if he was near the car location $c_{k-1}^{(i)}$.

Once the transportation mode is sampled, the motion velocity $v_k^{(i)}$ is sampled from a mixture of Gaussians which is conditioned on the mode. The value of the next edge transition variable $\tau_k^{(i)}$ is sampled based on the previous position of the person and a learned transition model. This is used in case the mean of the location estimate reaches an intersection. The edge association variable $\theta_k^{(i)}$ "snaps" the GPS reading to a street in the map. To sample $\theta_k^{(i)}$, we first determine the distance between the measurement, $z_k$, and the different streets in the vicinity. The probability of "snapping" $z_k$ to one of these streets is then computed from this distance. These assignments are crucial for the Kalman filter update described nextd.

*Kalman Filter Step*

After the RBPF generated all the sampled histories of a particle, that is, $s_k^{(i)} = \left\langle \langle \_, \_ \rangle, \langle \langle \_, \_ \rangle, m_{1:k}^{(i)}, v_{1:k}^{(i)}, \tau_{1:k}^{(i)}, \theta_{1:k}^{(i)} \right\rangle$, it then generates the missing location estimate $\langle \mu_k^{(i)}, \Sigma_k^{(i)} \rangle$ by updating the Kalman filter conditioned on the already sampled values. To elaborate, let us rewrite the second term on the right hand side of (8):

$$
\begin{aligned}
& p(l_k \mid m_{1:k}^{(i)}, v_{1:k}^{(i)}, \tau_{1:k}^{(i)}, \theta_{1:k}^{(i)}, z_{1:k}) \\
& = p(l_k \mid \langle \mu_{k-1}^{(i)}, \Sigma_{k-1}^{(i)} \rangle, m_k^{(i)}, v_k^{(i)}, \tau_k^{(i)}, \theta_k^{(i)}, z_k) && (10) \\
& \propto p(z_k \mid l_k, \theta_k^{(i)}) \int p(l_k \mid v_k^{(i)}, \tau_k^{(i)}, l_{k-1}^{(i)}) \, \mathcal{N}(l_{k-1}^{(i)}; \mu_{k-1}^{(i)}, \Sigma_{k-1}^{(i)}) \, \mathrm{d}l_{k-1}^{(i)} && (11)
\end{aligned}
$$

(10) follows from the fact that the Gaussian estimate $\langle \mu_{k-1}^{(i)}, \Sigma_{k-1}^{(i)} \rangle$ of the person's location is a sufficient statistic for all observations up to time $k-1$, whena conditioned on the particle's histories over the other parts of the state space. The justification of this step is the key reason for estimating histories rather than only the most recent states. Equation (11) now follows by applying Bayes rule and the independences in our estimation problem (cf. Fig. 1). It represents a standard recursive Bayes filter update rule; see [10,9] for details. The prior probability is given by the Gaussian of the previous Kalman filter estimate. Conditioned on the already sampled values $\theta_k^{(i)}, v_k^{(i)}$, and $\tau_k^{(i)}$, (11) reduces to a standard Kalman filter update.

In the prediction step, the traveled distance is predicted using the sampled Gaussian velocity component. The prediction, $\langle \widehat{\mu}_k^{(i)}, \widehat{\Sigma}_k^{(i)} \rangle$, results then from shifting and convolving the previous estimate by the predicted motion, thereby implementing the integration in (11) via the Kalman update (5). This prediction step is straightforward if the person stays on the same edge of the graph. If she transits over a vertex of the graph, then the next edge is given by the previously sampled edge transition ${\tau_k}^{(i)}$ (see upper panel in Fig. 2). To simplify computation, only the predicted mean $\widehat{\mu}_k^{(i)}$ is used to determine whether the person switches edges. In our experience this approximation is accurate enough for location tracking.

In the correction step, the predicted estimate $\langle \widehat{\mu}_k^{(i)}, \widehat{\Sigma}_k^{(i)} \rangle$ is corrected based on the most recent GPS measurement $z_k$, using (7). Intuitively, this correction compares the predicted mean $\widehat{\mu}_k^{(i)}$ with the location of $z_k$ and shifts the mean toward the measurement (under consideration of the uncertainties). The correction step for one Gaussian is illustrated in the lower panel of Fig. 2. Because we restrict the location estimates to the graph, we "snap" the GPS measurements onto the graph. The already sampled value of the edge association variable $\theta_k^{(i)}$ uniquely determines to which edge the reading is snapped. After

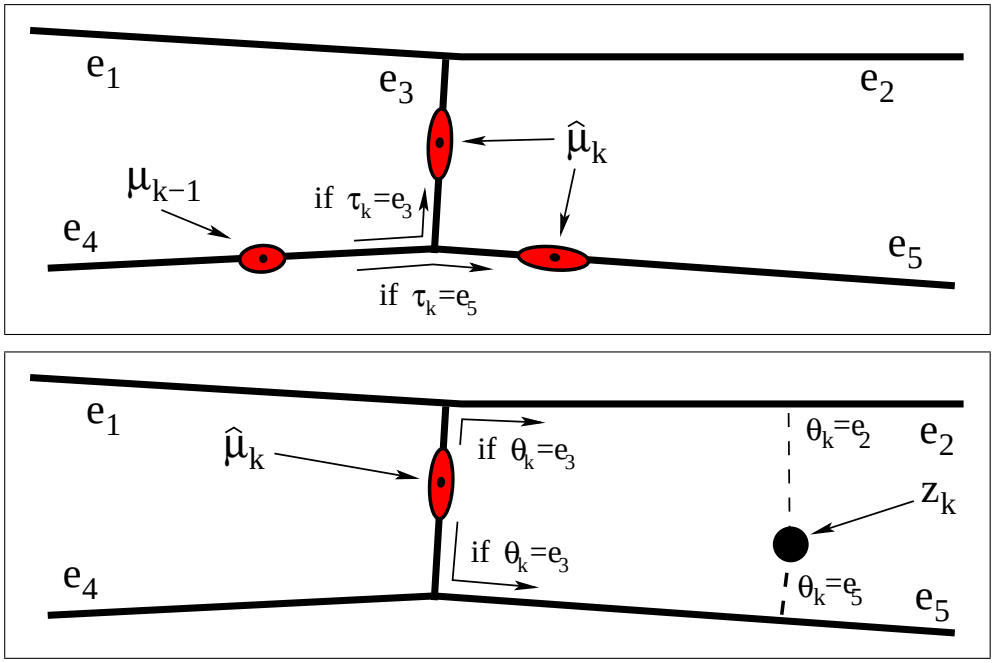



Fig. 2. Kalman filter prediction (upper panel) and correction (lower panel) on a street graph. The previous belief is located on edge $e_4$. When the predicted mean transits over the vertex, then the next location can be either on $e_3$ or $e_5$, depending on the sampled edge transition $\tau_k$. In the correction step (lower panel), the continuous coordinates of the GPS measurement, $z_k$, are between edges $e_2$ and $e_5$. Depending on the value of the edge association, $\theta_k$, the correction step moves the estimate up-wards or down-wards.

a GPS measurement is snapped onto one of the edges, we find the shortest path on the graph between the $\widehat{\mu}_k^{(i)}$ and the snapped measurement using the standard A* search algorithm. Then we can apply a one-dimensional Kalman filtering correction step and get the posterior location estimate $\langle {\mu_k}^{(i)}, {\Sigma_k}^{(i)} \rangle$.

Finally, the car location $c_k^{(i)}$ is updated. To see, we rewrite the first term of the factorization in (8) as

$$p(c_k \mid l_k, v_{1:k}, m_{1:k}, \theta_{1:k}, \tau_{1:k}, z_{1:k}) = \begin{cases} \delta(l_k - c_k) & \text{if } m_k = \texttt{CAR} \\ \delta(c_{k-1} - c_k)\; p(c_{k-1} \mid l_{k-1}, v_{1:k-1}, m_{1:k-1}, \theta_{1:k-1}, \tau_{1:k-1}, z_{1:k-1}) & \text{otherwise} \end{cases} \tag{12}$$

where $\delta(x)$ is the Dirac delta function that returns infinity if $x = 0$ and zero otherwise. (12) is based on the assumption that the person is always using the same vehicle and that nobody else moves the vehicle. As a result, if the person is in the car, then the car location is set to the person's location. Otherwise, the car location is the same as in the previous time step.

**RBPF_Flat**

**Inputs:** Sample set $S_{k-1} = \{\langle s_{k-1}^{(i)}, w_{k-1}^{(i)} \rangle \mid 1 \leq i \leq N\}$ and observation $z_k$

**Output:** Current sample set: $S_k = \{\langle s_k^{(i)}, w_k^{(i)} \rangle \mid 1 \leq i \leq N\}$

1. $S_k = \emptyset$ // *Initialize*
2. **for** $i = 1, \ldots, N$ **do** // *Generate samples*
3. Sample $m_k^{(i)} \sim p(m_k^{(i)} \mid m_{k-1}^{(i)}, \mu_{k-1}^{(i)}, c_{k-1}^{(i)})$ // *Sample transportation mode*
4. Sample $v_k^{(i)} \sim p(v_k^{(i)} \mid m_k^{(i)})$ // *Sample velocity*
5. Sample $\tau_k^{(i)} \sim p(\tau_k^{(i)} \mid \mu_{k-1}^{(i)})$ // *Sample decision at next vertex*
6. Sample $\theta_k^{(i)} \sim p(\theta_k^{(i)} \mid z_k)$ // *Sample GPS to edge association*
7. Compute $\langle \mu_k^{(i)}, \Sigma_k^{(i)} \rangle$ conditioned on $\langle \mu_{k-1}^{(i)}, \Sigma_{k-1}^{(i)} \rangle$, $v_k^{(i)}$,$\tau_k^{(i)}$, $\theta_k^{(i)}$, and $z_k$
8. **if** $m_k^{(i)} = \texttt{CAR}$ // *Update car location*
9. **then** $\langle \xi_k^{(i)}, \Xi_k^{(i)} \rangle := \langle \mu_k^{(i)}, \Sigma_k^{(i)} \rangle$ // *Car moves with person*
10. **else** $\langle \xi_k^{(i)}, \Xi_k^{(i)} \rangle := \langle \xi_{k-1}^{(i)}, \Xi_{k-1}^{(i)} \rangle$ // *Car does not move*
11. $w_k^{(i)} = w_{k-1}^{(i)} \, \mathcal{N}(z_k;\, \widehat{\mu}_k^{(i)}, \widehat{\Sigma}_k^{(i)} + Q_k)$ // *Kalman filter likelihood*
12. $S_k = S_k \cup \{\langle s_k^{(i)}, w_k^{(i)} \rangle\}$ // *Insert into sample set*
13. **end for**
14. Multiply / discard samples in $S_k$ based on normalized weights

Table 1. RBPF for flat model

*Importance Weights*

After all components of each particle are generated, the importance weights of the particles need to be updated using Equation (1). In our case, the sampling steps do not consider the most recent observation $z_k$ and the proposal distribution is identical to (2), resulting in importance weights proportional to the observation likelihood:

$$w_k^{(i)} \propto w_{k-1}^{(i)} \, p(z_k \mid s_k^{(i)}) \tag{13}$$

$$= w_{k-1}^{(i)} \, p(z_k \mid \langle \xi_{k-1}^{(i)}, \Xi_{k-1}^{(i)} \rangle, \langle \mu_{k-1}^{(i)}, \Sigma_{k-1}^{(i)} \rangle, m_{1:k}^{(i)}, v_{1:k}^{(i)}, \tau_{1:k}^{(i)}, \theta_{1:k}^{(i)}) \tag{14}$$

$$= w_{k-1}^{(i)} \, \mathcal{N}(z_k;\, \widehat{\mu}_k^{(i)}, \widehat{\Sigma}_k^{(i)} + Q_k) \tag{15}$$

This observation likelihood is computed based on each particle's sampled values, as given in (13). The likelihood can be computed in closed form from the Kalman filter correction step [9]. In the case of GPS readings, the likelihood of a measurement is given by a Gaussian with mean at the predicted location $\widehat{\mu}_k^{(i)}$ and variance given by the predicted location uncertainty $\widehat{\Sigma}_k^{(i)}$ plus measurement noise $Q_k$, as given in (15).

### *3.1.3 RBPF Algorithm for the Flat Model*

The algorithm **RBPF_Flat** is summarized in Table 1. The algorithm accepts as inputs a sample set representing the previous belief and the most recent GPS measurement. For each particle, the algorithm first samples the transportation mode in Step 3. The sampling distribution models that a person can only get on or off a bus when she is near a bus stop, and she can only get into the car when she is near the location where the car is parked. Then, the algorithm samples the velocity conditioned on the mode, the motion decision at the next vertex, and the association of the GPS reading to an edge in the street map (Steps 4-6).

Then, in Step 7, the Gaussian estimate of the person's location is updated using Kalman filtering. Steps 8 through 10 set the car location accordingly. The weight of each particle is determined in Step 11, based on the Kalman filter estimate of the person's location. After inserting each particle into the sample set, the algorithm performs resampling based on the importance weights. Resampling with minimal variance can be implemented efficiently (constant time per sample) using a procedure known under the name deterministic selection [18,19] or stochastic universal sampling [20].

## *3.2 Hierarchical Model*

Up to this point, we have described state estimation in a "flat" model, that is, a model that does not reason about a person's goals, trip segments, and novelty. We will now describe how to extend the RBPF for the flat model so as to estimate the posterior over the complete hierarchical model shown in Fig. 1. To additionally model goals, trip segments, novelty, and improved mode switching, we add the corresponding components to each particle of the RBPF:

$$s_k^{(i)} = \left\langle n_k^{(i)}, \langle g_k^{(i)}, t_k^{(i)} \rangle, f_k^{g(i)}, \langle \xi_k^{(i)}, \Xi_k^{(i)} \rangle, \langle \mu_k^{(i)}, \Sigma_k^{(i)} \rangle, v_k^{(i)}, m_k^{(i)}, f_k^{m(i)}, f_k^{t(i)}, \theta_k^{(i)}, \tau_k^{(i)} \right\rangle \quad (16)$$

Here each $\langle g_k^{(i)}, t_k^{(i)} \rangle$ is a *discrete distribution* over goals and trip segments. These distributions are estimated using exact inference conditioned on the sampled values, just like exact Kalman filtering is performed for the location estimates $\langle \mu_k^{(i)}, \Sigma_k^{(i)} \rangle$ in the flat model. $n_k^{(i)}$ represents whether or not the person currently follows an expected route, where expectation is based on historical data. The values $f_k^{g(i)}$, $f_k^{t(i)}$, and $f_k^{m(i)}$ represent sampled information about switching between goals, trip segments, and transportation modes, respectively.

### *3.2.1 RBPF Algorithm for the Hierarchical Model*

Table 2 summarizes one update step of the algorithm **RBPF_Hierarchical**, which implements Rao-Blackwellised particle filtering for the complete model except novel behavior. We omitt a full derivation of this algorithm, it is similar to the derivation of **RBPF_Flat** and to Rao-Blackwellised inference for abstract hidden Markov models, with which our model shares the high-level structure [21].

The algorithm accepts as inputs the sample set representing the previous belief and the most recent measurement. In order to be able to sample the switching nodes conditioned on the high-level nodes, the algorithm first samples a goal / trip segment combination from the previous distribution (Step 3). Then, Step 4 tests whether the previous location reached the end of the trip segment. In our implementation, this test returns true if $\mu_{k-1}^{(i)}$ just entered the edge of $\tilde{t}_{k-1}^{(i)}$.`end`. The goal switching node $f_k^{g(i)}$ is set to true if the end of the trip segment is reached and the trip segment ends in the goal location (Steps 7-9). The time period until the transportation mode on the new trip segment is switched, is sampled in Step 11, and decremented in Step 12. This semi-Markov mode switching enables the RBPF to model non-exponential waiting times between, for example, reaching a bus stop and getting on the bus. We found that this technique is far more robust than a straightforward approach that samples a mode switch at every iteration.

Then, in Step 13, the distribution over goals and trip segments is projected forward conditioned on the sampled switching nodes. Similar to Step 3, a goal / trip segment combination is sampled from the predicted distribution in Step 14. If the transportation mode switching counter reaches zero, then $m_k^{(i)}$ is set to the mode of the sampled trip segment (Step 16). Steps 18 through 24 correspond exactly to Steps 4 through 10 in the flat model, with a key difference in Step 19: While the flat model samples the transition $\tau_k^{(i)}$ at the next intersection solely based on the location $\mu_{k-1}^{(i)}$, the hierarchical model takes the current trip segment $\tilde{t}_k^{(i)}$ into account. Thus, the hierarchical model can have different transition probabilities at an intersection depending on the trip segment the person is following. As we will show in the experimental results, this additional dependency leads to greatly improved predictive capabilities.

The distribution over goals and trip segments is update in Step 25. The sampled transition $\tau_k^{(i)}$ plays an important role in this update, since it indicates in which direction the person is moving. Similarly, the sampled transportation mode $m_k^{(i)}$ indicates transitions between trip segments. After each particle is weighted and inserted into the sample set, the algorithm finishes with the resampling step.

**RBPF_Hierarchical**

**Inputs:** Sample set $S_{k-1} = \{\langle s_{k-1}^{(i)}, w_{k-1}^{(i)} \rangle \mid 1 \leq i \leq N\}$ and observation $z_k$

**Output:** Current sample set: $S_k = \{\langle s_k^{(i)}, w_k^{(i)} \rangle \mid 1 \leq i \leq N\}$

1. $S_k = \emptyset$ // *Initialize*
2. **for** $i = 1, \ldots, N$ **do** // *Generate N samples*
3. Sample $(\tilde{g}_{k-1}^{(i)}, \tilde{t}_{k-1}^{(i)}) \sim p(g_{k-1}^{(i)}, t_{k-1}^{(i)})$ // *Draw goal and trip segment*
4. **if** $\mu_{k-1}^{(i)} = \tilde{t}_{k-1}^{(i)}$.`end` // *Just reached end of trip segment?*
5. **then** $f_k^{t(i)} := true$
6. **else** $f_k^{t(i)} := false$
7. **if** $f_k^{t(i)} = true$ **and** $\tilde{g}_{k-1}^{(i)} = \tilde{t}_{k-1}^{(i)}$.`end` // *Goal reached?*
8. **then** $f_k^{g(i)} := true$
9. **else** $f_k^{g(i)} := false$
10. **if** $f_k^{t(i)} = true$ // *Sample when to switch mode*
11. **then** sample $f_k^{m(i)} \sim$ Uniform$[0,$ max-waiting-time$]$
12. **else** $f_k^{m(i)} := f_{k-1}^{m(i)} - 1$
13. Compute $p(\hat{g}_k^{(i)}, \hat{t}_k^{(i)} \mid f_k^{g(i)}, f_k^{t(i)}, g_{k-1}^{(i)}, t_{k-1}^{(i)})$
14. Sample $(\tilde{g}_k^{(i)}, \tilde{t}_k^{(i)}) \sim p(\hat{g}_k^{(i)}, \hat{t}_k^{(i)})$ // *Draw goal and trip segment*
15. **if** $f_k^{m(i)} = 0$ // *Change transportation mode?*
16. **then** $m_k^{(i)} = \tilde{t}_k^{(i)}$.`mode`
17. **else** $m_k^{(i)} = m_{k-1}^{(i)}$
18. Sample $v_k^{(i)} \sim p(v_k^{(i)} \mid m_k^{(i)})$ // *Sample velocity*
19. Sample $\tau_k^{(i)} \sim p(\tau_k^{(i)} \mid \mu_{k-1}^{(i)}, \tilde{t}_k^{(i)})$ // *Sample decision at next vertex*
20. Sample $\theta_k^{(i)} \sim p(\theta_k^{(i)} \mid z_k)$ // *Sample GPS to edge association*
21. Compute $\langle \mu_k^{(i)}, \Sigma_k^{(i)} \rangle$ conditioned on $\langle \mu_{k-1}^{(i)}, \Sigma_{k-1}^{(i)} \rangle$, $v_k^{(i)}$,$\tau_k^{(i)}$, $\theta_k^{(i)}$, and $z_k$
22. **if** $m_k^{(i)} =$ `CAR` // *Update car location*
23. **then** $\langle \xi_k^{(i)}, \Xi_k^{(i)} \rangle := \langle \mu_k^{(i)}, \Sigma_k^{(i)} \rangle$ // *Car moves with person*
24. **else** $\langle \xi_k^{(i)}, \Xi_k^{(i)} \rangle := \langle \xi_{k-1}^{(i)}, \Xi_{k-1}^{(i)} \rangle$ // *Car does not move*
25. Compute $p(g_k^{(i)}, t_k^{(i)} \mid m_k^{(i)}, \tau_k^{(i)}, \hat{g}_k^{(i)}, \hat{t}_k^{(i)})$
26. $w_k^{(i)} = w_{k-1}^{(i)} \mathcal{N}(z_k;\, \widehat{\mu}_k^{(i)}, \widehat{\Sigma}_k^{(i)} + Q_k)$ // *Update particle weight*
27. $S_k = S_k \cup \{\langle s_k^{(i)}, w_k^{(i)} \rangle\}$ // *Insert into sample set*
28. **end for**
29. Multiply / discard samples in $S_k$ based on normalized weights

Table 2. RBPF for hierarchical model

**RBPF_Novelty**
**Inputs:** Sample set $S_{k-1} = \{\langle s_{k-1}^{(i)}, w_{k-1}^{(i)} \rangle \mid 1 \leq i \leq N\}$ and observation $z_k$
**Output:** Current sample set: $S_k = \{\langle s_k^{(i)}, w_k^{(i)} \rangle \mid 1 \leq i \leq N\}$

1. $S_k = \emptyset$ *// Initialize*
2. **for** $i = 1, \ldots, N$ **do** *// Generate N samples*
3. Sample $n_k^{(i)} \sim p(n_k^{(i)} \mid n_{k-1}^{(i)})$ *// Novel behaviour?*
4. **if** $n_k^{(i)} = true$
5. **then** $\langle s_k^{(i)}, w_k^{(i)} \rangle :=$ **RBPF_Flat**$(\langle s_{k-1}^{(i)}, w_{k-1}^{(i)} \rangle)$
6. **else** $\langle s_k^{(i)}, w_k^{(i)} \rangle :=$ **RBPF_Hierarchical**$(\langle s_{k-1}^{(i)}, w_{k-1}^{(i)} \rangle)$
7. **end for**
8. Multiply / discard samples in $S_k$ based on normalized weights

Table 3. RBPF for novelty detection

### *3.2.2 Detecting Novel Behavior*

In order to distinguish novel from expected behavior, we compare the predictive capabilities of a model that is not trained for a specific user and a model with goals, trip segments, and transition probibilities trained on data collected by the specific user (see Section 4). The idea behind this approach is that if the person behaves as expected based on his history, then the trained model is able to much better predict the person's motion. If the user follows a novel route or commits an error, then the untrained model is more predictive, since it is not biased toward any specific route.

This approach can be implemented naturally within our RBPF technique. To do so, we sample the node $n_k^{(i)}$ and, if $n_k^{(i)} = false$, a particle is associated with a trained model. When $n_k^{(i)} = true$, the particle is updated using an untrained model. As a result, while particles with $n_k^{(i)} = false$ are strongly biased toward following routes and transportation routines the user has visited in the training data, particles with $n_k^{(i)} = true$ have no user-specific preferences for certain motion directions or modes of transportation. The algorithm **RBPF_Novelty**, shown in Table 3, implements novelty detection.

Step 3 samples a particle's novelty. Switching from $n_{k-1}^{(i)} = false$ to $n_k^{(i)} = true$ indicates that the person just left the routes expected by the learned model. Switching from $true$ to $false$ models the situation when the person gets back onto a known route. The probability for these two cases is significantly lower than the probability of remaining in the previous novelty mode. If the sampled novelty is $true$, then the particle is udpated using the untrained, flat RBPF algorithm in Step 5. Here, we choose the flat model since it supports unbiased motion more naturally and efficiently than a hierarchical model with uniform transitions. If the novelty is $false$, then the particle is updated using

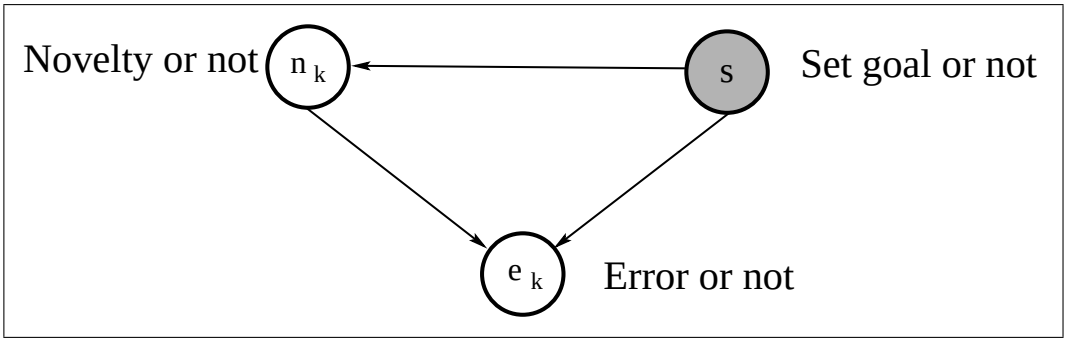



Fig. 3. Bayesian network to estimate the error probability. All three variables are boolean.

a hierarchical model that is trained on historical data collected by the user. Finally, if the novelty mode switches from $n_{k-1}^{(i)} = true$ to $n_k^{(i)} = false$, we re-initialize the distribution over goals and trip segments to uniform.

The resulting technique is able to detect when a user deviates from a known route and when she returns to a previously used route. However, it does not distinguish errors from deliberate novel behavior. In some cases, we may want to explicitly estimate the probability of an error. This can be done using the model shown in Fig. 3. In this model, the variable $e_k$ represents whether a user has made an error by time $k$. The value of $e_k$ depends on the novelty $n_k$ and the *observed* variable $s$ that indicates whether the *true* goal is known.

When the true goal is unknown (as we have assumed so far), *i.e.*, $s = False$, we can estimate the error probability as

$$\begin{aligned} P(e_k = True \mid s = False) &= \sum_{n_k} P(n_k \mid s = False) P(e_k = True \mid n_k, s = False) \\ &= P(n_k = True \mid s = False) P(e_k = True \mid n_k = True, s = False) \quad (17) \end{aligned}$$

where $P(n_k \mid s = False)$ is estimated by sampling as we have discussed, and (17) follows from setting $P(e_k = True \mid n_k = False, s = False)$ to 0 (*i.e.*, errors only happen in novel situations). Note $P(e_k = True \mid n_k = True, s = False)$ is a user-dependent parameter that we set manually: for people who seldom make mistakes, we could choose its value as 0, while for people with cognitive disabilities, the value should be set much higher.

When the user's *true* intention is known, we can integrate the knowledge into our inference and predict errors more accurately. It is possible for the system to know where the user is going, for example, if the user asks for directions to a destination, if a care-giver or job coach indicates the "correct" destination, or if the system has access to a location enabled date-book. In these cases, $s = True$, and the error probability is computed as

$$P(e_k = True \mid s = True) = \sum_{n_k} P(n_k \mid s = True) P(e_k = True \mid n_k, s = True)$$
$$= P(n_k = True \mid s = True) P(e_k = True \mid n_k = True, s = True) \quad (18)$$

where we assume $P(e_k = True \mid n_k = False, s = True) = 0$. $P(n_k \mid s = True)$ can be estimated similarly to the previous case; the difference is that we now *clamp* the supplied value of the goal and/or trip segment in the learned model. $P(e_k = True \mid n_k = True, s = True)$ is usually set close to 1, *i.e.*, if a user does novel activities while the goal is clamped, he probably makes an error.

## 4 Learning

Learning the hierarchical model includes two procedures: first, learning variable domains and then, estimating transition parameters. Both procedures are completely unsupervised and generate unique results for an individual. Calculating the variable domains correspond to learning mode transfer locations, trip segments, and goals. These values comprise the range of the variables $g_k$ and $t_k$ in the hierarchical model shown in Fig. 1. After the variable domains are determined, we are able to estimate the transition matrices at all levels of the hierarchical model.

### *4.1 Finding mode transfer locations*

Trip segments begin and end at goals and/or locations where a user often changes transportation modes (*e.g.*, her usual bus stops and parking lots). In order to find those mode transfer locations, we estimate the mode transition probabilities for each street, using the expectation maximization (EM) algorithm [22] and the flat activity model described previously. Even before learning, knowledge about the bus stops and the fact that the car is either parked or moves with the person already provides important constraints on mode transitions.

The EM algorithm starts with a prior estimate of the mode transition probabilities, then it improves the estimate iteratively based on the GPS training data. During learning, each iteration in the EM includes two steps: an E-step and an M-step. In the E-step, both a forward and a backward filtering pass are performed, and the transition counts of the two passes are combined. Then, in the M-step, the parameters are updated based on the counts. After EM, the mode transfer locations for a user are taken to be those locations at which the mode switching probability exceeds a certain threshold. Typically, this will identify the user's favorite bus stops, and a set of edges where the user

typically parks her car.

### *4.2 Finding goals*

In [23], significant locations are extracted from user traces by detecting places where GPS signals are lost. The disadvantage of such an approach is that it can only detect goals associated with moving indoors. In contrast, we consider goal locations to be those locations where a person typically spends extended periods of time whether indoors or outdoors. Since we model loss of GPS signal by transitioning to "BUILDING" mode, our model detects both kinds of goals. In order to do this, we keep track of how long a person stays on every edge in the graph, which can be easily collected during the EM smoothing steps in the flat model.

Once significant edges are detected, they are clustered by combining edges that are connected or very close. Note that goal extraction and clustering rely on the underlying street graph, which simplifies the process significantly. Durations are then thresholded to determine a subset of edges that are considered possible goals during reasoning with the full hierarchical model.

### *4.3 Estimating transition matrices*

Once goals and trip segments are determined, we can extend the flat model by inserting those significant locations into the higher levels of the activity model. Then, we use another round of EM to estimate the transition matrices between the goals, between the trip segments given the goal, and between the adjacent streets given the trip segment (used in Steps in the hierarchical RBPF algorithm shown in Table 2).

Similar to EM for estimating mode transitions, during the E-steps, smoothing is performed by tracking the states both forward and backward in time. Then the M-steps update the model parameters using the frequency counts generated in the E-steps. All transition parameters are smoothed using Dirichlet priors. Our system does not currently learn the parameters associated with novelty dectection. This would entail learning the likelihood of a user making an error versus doing something novel, and was beyond the scope of this work.

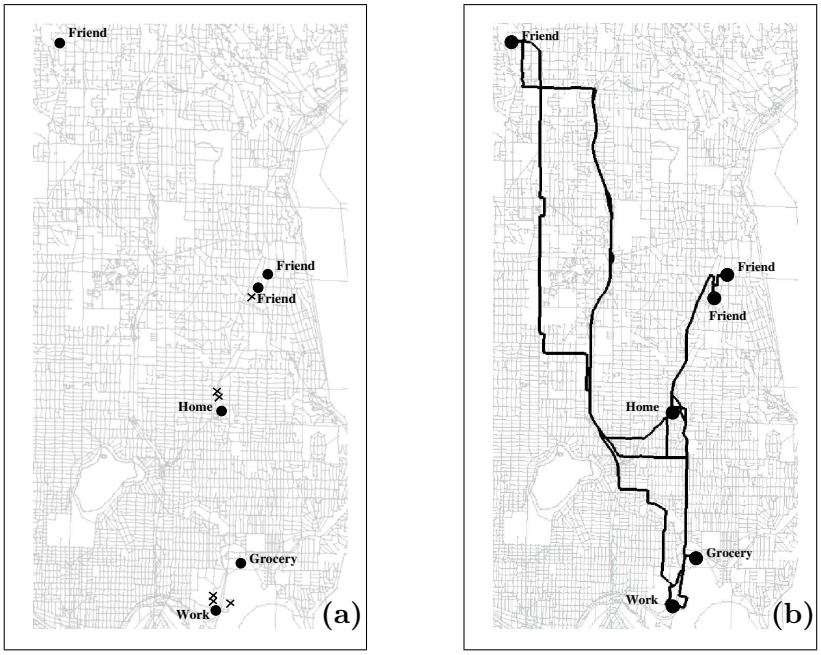



Fig. 4. (a) Street map along with goals (dots) learned from 30 days of data. Learned trip switching locations are indicated by cross marks. (b) Most likely trajectories between goals obtained from the learned transition matrices.

## 5 Experimental Results

To validate our model, we collected 60 days of GPS data from one person wearing a small GPS unit. We used the first 30 days for learning and the second 30 days for testing.

### *5.1 Activity model learning*

The learning was done without manual labeling. The system precisely identifies the subject's six most common transportation goals and all frequently used bus stops and parking lots, as shown in Fig. 4 (a).
`DDD which threshold, how sensitive?` After recognizing the goals and transfer locations, parameter learning estimates the transition matrices at all levels of the model. Using those transition matrices, we can extract the most likely trajectories on the street map between pairs of the goals, as shown in Fig. 4 (b).

Fig. 5 shows a close up display of the area near the work place. The learned results clearly show the common routes using different modes of transportation, as well as the usual bus stops and parking lots.

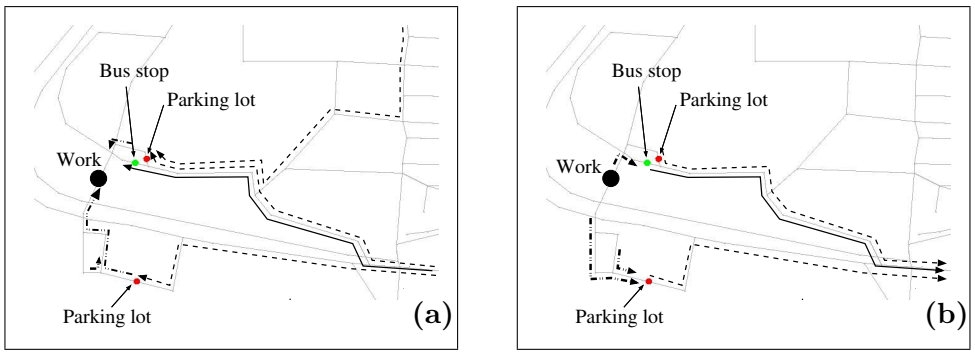



Fig. 5. Close up of the area around the work place. (a) Shown are edges that are connected by likely transitions (probability above 0.75), given that the goal is the work place (dashed lines indicate car mode, solid lines bus, and dashed-dotted lines foot). (b) Learned transitions in the same area conditioned on home being the goal.

### *5.2 Empirical comparison to other models*

The hierarchical model is very expressive and able to answer many useful queries. For example, many applications need to query the probability of a given goal. In this section we present results comparing the goal prediction performance of our hierarchical model with a flat model [2] and a second-order Markov model (2MM) trained on sequences of goals [23].

The flat model is basically a first-order Markov model over the street blocks. Thus, in order to calculate the probability of a goal, one must calculate the sum over all possible paths to the goal, which is intractable if the goal is far away. A reasonable approximation is to compute the probability of the most likely path to the goal. Fig. 6(a) compares the result of such a query on the probability of the goal being the work place during an episode of traveling from home to work. As one can see, the hierarchical model quickly assigns a high probability to the true goal, while the estimate from the flat model is meaningless until the user is near the goal. In [2], the flat model is also used to predict the street blocks and transportation modes in the future. As shown in Fig. 6(b), the prediction capability of the hierarchical model is much better than that of the flat model. For instance, in 50% of the cases, the flat model is able to correctly predict the motion and transportation mode of the person for 5 city blocks, while the hierarchical model can predict correctly for 43 blocks.

The 2MM model indroduced in [23] is a second-order Markov model that only reasons about transitions between goal locations. Since this model ignores GPS measurements collected during transit between goals, it cannot refine the goal prediction as a person moves to a goal. To show the difference in preformance, we labeled the 30 days of test data with the true goals and computed the prediction accuracy using the 2MM and our hierarchical model, which was learned using the same training data. The average prediction accuracies at

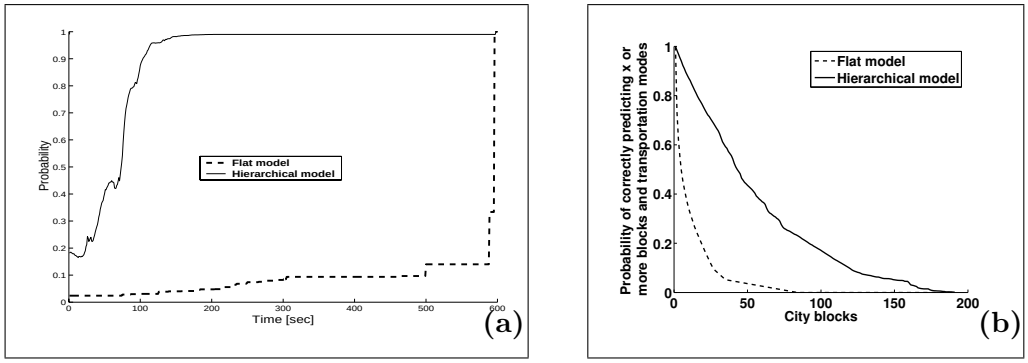



Fig. 6. Comparison of flat model and hierarchical model. (a) Probability of the true goal (work place) during an episode from home to work, estimated using the flat and the hierarchical model. (b) Location and transportation mode prediction capabilities of the learned model.

| Model | Avg. accuracy at given time | | | |
|---|---|---|---|---|
| | beginning | 25% | 50% | 75% |
| 2MM | 0.69 | 0.69 | 0.69 | 0.69 |
| Hierarchical model | 0.66 | 0.75 | 0.82 | 0.98 |

Table 4. Comparison of goal predictions using 2MM and hierarchical model

the beginning of each trip and after 25%, 50%, and 75% of each trip was completed are listed in Table 4. At the beginning, our model predicts the next goal using first-order transition matrices; it performs only slighty worse than the 2MM. However, by integrating real time measurements, our predictions become much more accurate while 2MM's estimates remain the same.

### *5.3 Error detection*

Another important feature of our model is the capability to differentiate normal, erroneous, and deliberate novel activities.

Whenever a true destination is known, the system can *clamp* it as the user's goal, $g_k$, in the hierarchical model, estimate the novelty probability and then compute the error probability using Equation (18). To evaluate the effect of clamping on our model's ability to detect errors we conducted two experiments. In each experiment, we calculated the probabilities of normal behavior and user errors over time (see Equations (17) and (18)) and compared the results (see Fig. 7).

In the first experiment, the user had notified the system that the true desti-

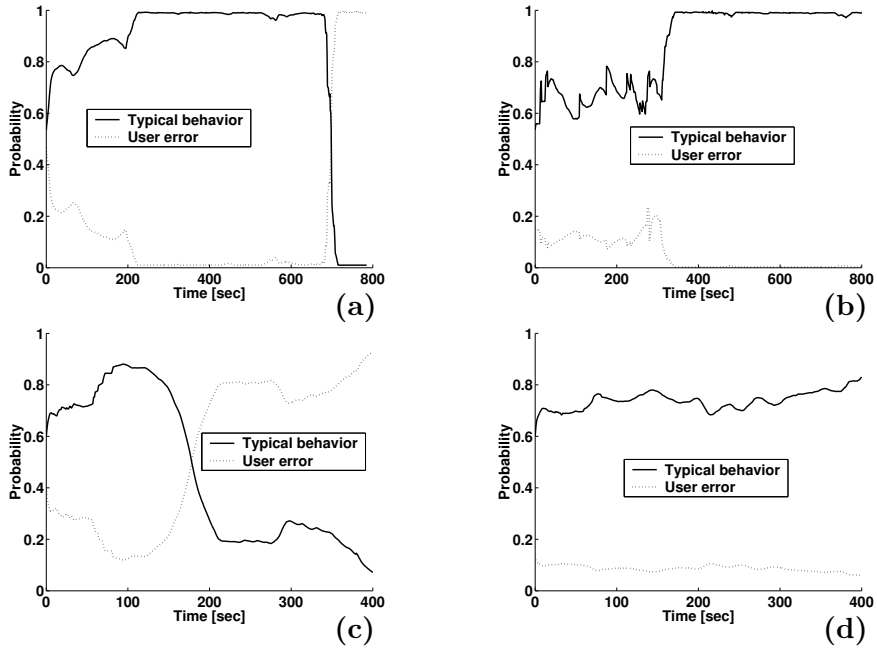



Fig. 7. The probabilities of typical behavior and user errors in two experiments (when goal is unclamped, the prior ratio of typical behavior, user error and deliberate novel behavior is 3:1:2; when goal is clamped, the probabilities of deliberate novel behavior are always zero): (a) Bus experiment with goal clamped; (b) Bus experiment with goal unclamped; (c) Foot experiment with goal clamped; (d) Foot experiment with goal unclamped.

nation was going home; however, the user took an incorrect bus toward one of his friend's houses. Both the correct bus route and the incorrect bus route had been learned during training. For the first 700 seconds, the wrong bus route coincided with the correct one and both the clamped and unclamped inference engines believed that the user was in normal mode, *i.e.*, $n_k = False$ (see Fig. 7(a and b)). But when the incorrect bus took a turn at time 700 that the user had never taken *to get home*, the probability of errors in the model with the goal clamped to home dramatically jumped (see Fig. 7(a)). In contrast, the unclamped model could not conclude that the user was making an error because the user, while on the wrong bus route to get home, was on a bus route consistent with going to other familiar goals (see Fig. 7(b)).

In the second experiment, the user left his office and then proceeded to walk in a direction away from his normal parking spot. When the destination was not specified (see Fig. 7(d)), the tracker had a fairly steady level of confidence in the user's path (there were many previously observed paths from his office consistent with the observed data). However, when the destination was specified (see Fig. 7(c)), the system initially inferred that the behavior was consistent with walking toward the parking lot, and then, as the user began to turn away at time 125, the tracker started doubting the success of the user's intentions. The tracker's confidence in the user's success correspondingly dropped.

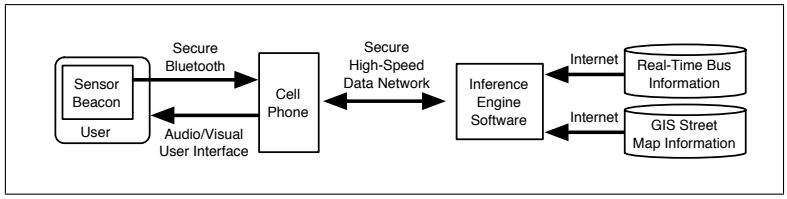



Fig. 8. The client-server architecture of Opportunity Knocks

## 6 Application: Opportunity Knocks

Our motivation for developing these techniques was grounded in the observation that for many individuals, mobility in the community means using public transportation. However, public transportation can be daunting for anyone who is born with below average cognitive abilities or whose cognitive abilities have begun to decline. If impaired individuals had effective compensatory cognitive aids to help them use public transportation, their independence and safety would improve, they would have new opportunities for socialization and employment, and stress on their families and care givers would be reduced.

Based on the techniques we have discussed, we developed a ubiquitous computing system, called "Opportunity Knocks" (OK) [3], in order to explore the feasibility of just such a cognitive aid. This system targets mentally retarded individuals and individuals with traumatic brain injury, who are generally high functioning but unable to use public transportation due to short-term confusion or memory lapses. The name of our system is derived from the desire to provide our users with a source of computer generated opportunities from which they can learn more efficient transportation routes, and correct simple errors before they become dangerous errors. When the system has determined that an especially important opportunity has made itself available, it plays a sound like a door knocking to get the user's attention.

Our system has a client-server architecture, as shown in Fig. 8. The client side consists of a cell phone and a GPS sensor beacon which communicates position information to the phone via Bluetooth technology. The cell phone transmits GPS readings and user queries to a server through a wireless (GPRS) data network. On the server side, the learning and inference engine integrates the user information with map and bus information, and sends inference results or suggestions back to the client side.

In the experiment, explained in Fig. 9 and Fig. 10, a user with our device boarded a bus to get home after carrying the system with them for 1 month (a similar scenario to Fig. 7(a, b), but with different data). Unfortunately, the user boarded the wrong bus, which shared the first part of the bus route in common with the correct bus. OK detected the mistake and guided the user back on track. The top of each panel in the figures display a representation

of the reasoning process that the inference engine is undertaking. The center portion of each panel displays what the user interface displayed at each stage of the experiment, and the bottom portion holds a text description of the frame.

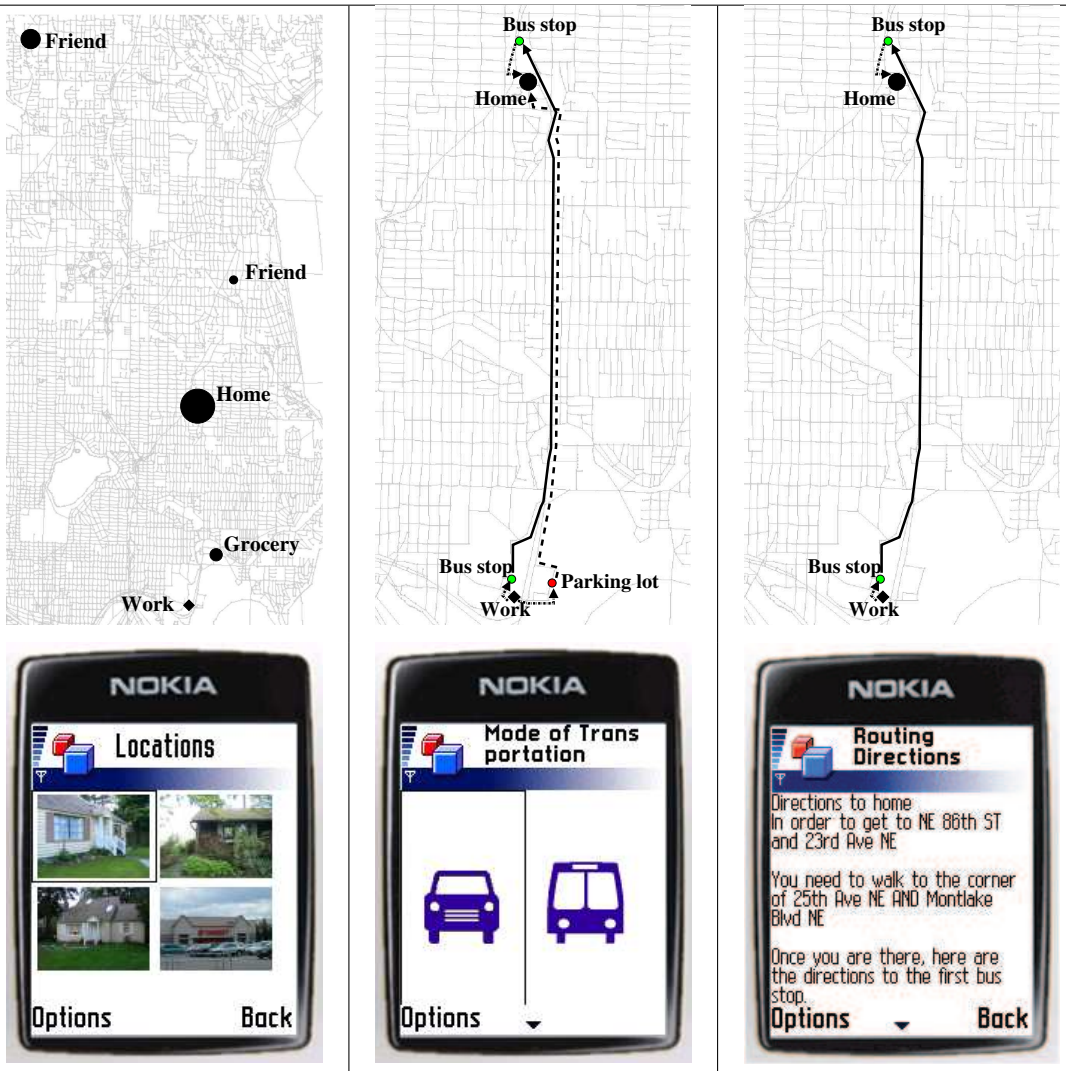
## 7 Related work

Over the last few years, estimating a person's activities has gained increased interest in the AI, robotics, and ubiquitous computing communities. Ashbrook and Starner [23] learn significant locations from logs of GPS measurements based on the time a person spends at a certain location (see also [24,25] for similar ideas). For these locations, they use frequency counting to estimate the transition parameters of a second-order Markov model. Their approach then predicts the next goal based on the current and the previous goals. In contrast to our approach, their model is not able to refine the goal estimates using GPS information observed when the user is between significant locations. Furthermore, such a coarse representation does not allow the detection of potential user errors.

In our previous work [2], we estimate a person's location and mode of transportation from GPS measurements using a non-hierarchical model. Since the model has no notion of significant locations, it is not able to predict the high-level goal of a person. By conditioning on goals and segments of a trip, our hierarchical model is able to learn more specific motion patterns of a person, which also enables us to detect user errors. As we showed in the experiments, predictions can also be accurately made about user locations much further into the future.

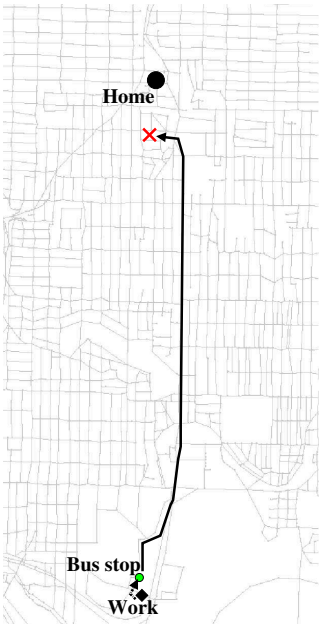
The system in this paper can extract significant goals from GPS data, but cannot automatically determine the *type* of place, *e.g.*, "home", "work", "friend's house", *etc.*. Complementary to this work, in [26] we presented a discriminatively trained model to automatically classify significant places and activities based on the framework of relational probabilistic models [27,28]. In [29,30], we showed how to use hierarchical conditional random fields to simultaneously detect and label a person's significant places. While these more recent techniques are able to incorporate a wide range of information in order to distinguish types of goals, they are not designed to predict a user's goal or to detect user errors. In contrast, these capabilities are inherent to the generative technique described in this paper.

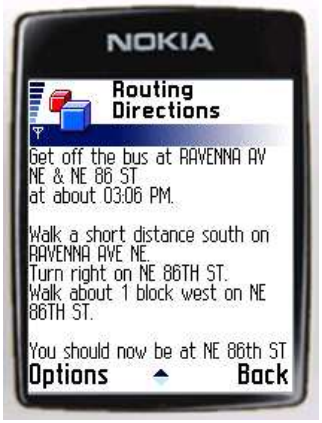
In the context of probabilistic plan recognition, Bui and colleagues introduced the abstract hidden Markov model, which uses hierarchical representations to efficiently infer a person's goal in an indoor environment from camera infor-




| | | |
|---|---|---|
| This frame shows the user at work when the system identified the most likely destinations. Circle areas correspond to relative likelihood. The phone displayed images of the most likely destinations left to right, top to bottom: home, friend 1, friend 2, grocery store. | After the user indicated that he'd like to go home, the system identified two routes that he usually takes, a bus route shown in solid lines and a car route shown in dashed lines. The system asked the user which way he would like to proceed. | The user selected the bus route and *OK* presented a text description of the learned route. The user proceeded to the bus stop, and boarded a bus. The bus that the user boarded however was going to his friend's house, a familiar, but incorrect route considering the expressed intention to go home. |
| **1** | **2** | **3** |

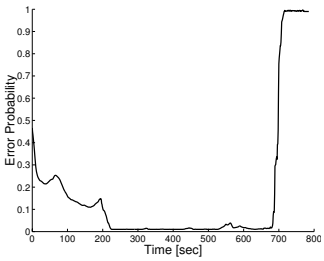
Fig. 9. An experiment on the Opportunity Knocks (Part I)




The system was unable to identify that the user was on the wrong bus because the routes coincided for the first portion of the bus ride. Before getting to the correct bus stop for going home, the system observed that the user had departed from the expected trip segment and turned west.

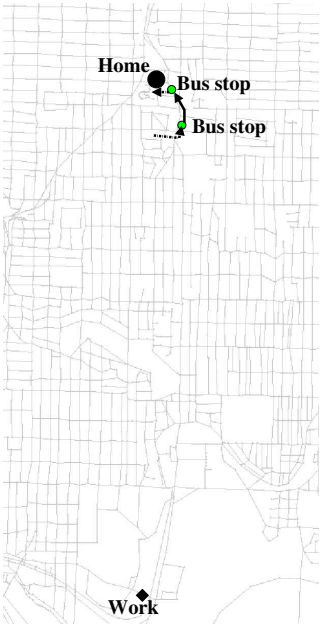
**4**


This graph shows how the system reasons about errors. When the error probability rises above 50% confidence, the system believes an abnormal condition exists. *OK* waits until a clamped curve breaks 80% confidence before alerting.


When the bus diverted from the correct route, the system identified the behavior condition as an error. This was possible *even though* the user was on a frequently taken route. In response it proactively made its door knocking alert noise and showed a corresponding message

**5**





Once off the incorrect bus, the user reselects home as the destination. This time OK queries a real-time bus planning system for a route home. The user is directed to walk back to the arterial road and catch a different bus that is going the correct way.

**6**

Fig. 10. An experiment on the Opportunity Knocks (Part II)

mation [1]. This model was extended to include memory nodes, which enables the transfer of context information over multiple time steps [21]. Efficient inference algorithms were developed for their models using Rao-Blackwellised particle filters. Since our model has a similar structure to theirs, we apply the inference mechanisms developed in [21]. Our work goes beyond the work of Bui *et al.* in that we show how to learn the structure and the parameters of the hierarchical activity model from data. Furthermore, our low level estimation problem is more challenging than their indoor tracking problem.

Gogate and colleagues [31] show how to perform efficient inference in hybrid dynamic mixed models, which are probabilistic models that contain deterministic constraints. They apply this technique in a simplified version of our scenario, not considering trip segments or different modes of transportation. However, their results indicate that inference in our model could be sped up by applying their modified RBPF inference algorithm.

In the context of mobile robotics, Cielniak *et al.* [32] applied a two level model to track and predict the location of people using a mobile robot equipped with a laser range-finder. Their model learns a person's trajectories using a mixtures of Gaussians approach, where each trajectory is assumed to contain the same number of Gaussians. Due to this representation, they are only able to track a person along paths the robot has observed during training. Thus, the technique is not able to track and detect novel behaviors.

In our work, the street graph gives a natural discretization of continuous 2D space and is crucial for our inference and learning. Interestingly, the graph-based tracking and learning can also be applied to indoor environments, where street maps do not exist. For example, in [33,34], the generalized Voronoi graph of the free space in an indoor environment was automatically generated from the occupancy map, and then human movement patterns were learned on that artificial "street map". In [35], a similar graph was constructed manually, and a hidden Markov model was learned based on the graph.

The task of detecting anomalous events in time series data has been studied extensively in the data-mining community [36], but remains an open and challenging research problem. We present the first results on novelty and error detection in location and transportation prediction using an integrated system that compares the likelihood of a learned hierarchical model against that of a prior model.

## 8 Conclusions and Future Work

We have described the foundations and experimental validation of a hierarchical model that can learn and infer a user's daily movements and use of different modes of transportation. The model can be learned using unlabeled data, and online inference can be efficiently performed. Our results show that the approach can provide predictions of movements to distant goals, and support a simple and effective strategy for detecting novel events that may indicate user errors.

The main limitation of the system is that it uses fixed thresholds to extract goals and mode transfer locations. In practice, any fixed threshold leads to errors. Some significant goals, for example, the place where the user drops off his children at school, may be visited only briefly, and so would be excluded by a high threshold. A lower threshold, however, would include too many insignificant locations, for example, a place where the user briefly waited at a traffic light. Our most recent work addresses this problem by developing a unified model that simultaneously solve the tasks of identifying and labeling significant locations, and inferring transportation routines.

## 9 Acknowledgments

This work was funded in part by National Science Foundation grants IIS-0120307, IIS-0307906, and IIS-0093406, National Institute on Disability and Rehabilitation Research grant H133A031739, Office of Naval Research grant N00014-02-1-0932, DARPA's SDR Program (grant number NBCHC020073), and Intel Corporation.

## References

[1] H. Bui, S. Venkatesh, G. West, Policy recognition in the abstract hidden markov model, Journal of Artificial Intelligence Research (JAIR) 17.

[2] D. Patterson, L. Liao, D. Fox, H. Kautz, Inferring high-level behavior from low-level sensors, in: International Conference on Ubiquitous Computing (UbiComp), 2003.

[3] D. J. Patterson, L. Liao, K. Gajos, M. Collier, N. Livic, K. Olson, S. Wang, D. Fox, H. Kautz, Opportunity Knocks: a System to Provide Cognitive Assistance with Transportation Services, in: I. S. Nigel Davies, Elizabeth Mynatt (Ed.), Proceedings of UBICOMP 2004: The Sixth

International Conference on Ubiquitous Computing, Vol. LNCS 3205, Springer-Verlag, 2004.

[4] T. Dean, K. Kanazawa, Probabilistic temporal reasoning, in: Proc. of the National Conference on Artificial Intelligence (AAAI), 1988.

[5] K. Murphy, Dynamic bayesian networks: Representation, inference and learning, Ph.D. thesis, UC Berkeley, Computer Science Division (2002).

[6] A. Doucet, J. de Freitas, K. Murphy, S. Russell, Rao-Blackwellised particle filtering for dynamic Bayesian networks, in: Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI), 2000.

[7] K. Murphy, S. Russell, Rao-Blackwellised particle filtering for dynamic Bayesian networks, in: A. Doucet, N. de Freitas, N. Gordon (Eds.), Sequential Monte Carlo in Practice, Springer-Verlag, New York, 2001.

[8] A. Doucet, N. de Freitas, N. Gordon (Eds.), Sequential Monte Carlo in Practice, Springer-Verlag, New York, 2001.

[9] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, Cambridge, MA, 2005, iSBN 0-262-20162-3.

[10] Y. Bar-Shalom, X.-R. Li, T. Kirubarajan, Estimation with Applications to Tracking and Navigation, John Wiley, 2001.

[11] K. Murphy, Bayesian map learning in dynamic environments, in: Advances in Neural Information Processing Systems (NIPS), 1999.

[12] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, FastSLAM: A factored solution to the simultaneous localization and mapping problem, in: Proc. of the National Conference on Artificial Intelligence (AAAI), 2002.

[13] A. Doucet, N. Gordon, V. Krishnamurthy, Particle filters for state estimation of jump Markov linear systems, IEEE Transactions on Signal Processing 49 (3).

[14] C. Kwok, D. Fox, Map-based multiple model tracking of a moving object, in: RoboCup 2004: Robot Soccer World Cup VIII, Vol. 3276, 2004.

[15] D. Schulz, D. Fox, J. Hightower, People tracking with anonymous and id-sensors using Rao-Blackwellised particle filters, in: Proc. of the International Joint Conference on Artificial Intelligence (IJCAI), 2003.

[16] N. de Freitas, Rao-Blackwellised particle filtering for fault diagnosis, IEEE Aerospace,.

[17] R. Morales-Menéndez, N. de Freitas, D. Poole, Real-time monitoring of complex industrial processes with particle filters, in: Advances in Neural Information Processing Systems 15, 2002.

[18] G. Kitagawa, Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, Journal of Computational and Graphical Statistics 5 (1).

[19] S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for on-line non-linear / non-Gaussian Bayesian tracking, IEEE Transactions on Signal Processing XX.

[20] J. Baker, Reducing bias and inefficiency in the selection algorithm, in: Proc. of the Second International Conference on Genetic Algorithms., 1987.

[21] H. Bui, A general model for online probabilistic plan recognition, in: Proc. of the International Joint Conference on Artificial Intelligence (IJCAI), 2003.

[22] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, in: Proceedings of the IEEE, IEEE, 1989, iEEE Log Number 8825949.

[23] D. Ashbrook, T. Starner, Using GPS to learn significant locations and predict movement across multiple users, Personal and Ubiquitous Computing 7 (5).

[24] R. Hariharan, K. Toyma, Project lachesis: Parsing and modeling location histories, in: GIScience, 2004.

[25] J. H. Kang, W. Welbourne, B. Stewart, G. Borriello, Extracting places from traces of locations, in: The Second ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots, 2004.

[26] L. Liao, D. Fox, H. Kautz, Location-based activity recognition using relational Markov networks, in: Proc. of the International Joint Conference on Artificial Intelligence (IJCAI), 2005.

[27] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, Learning probabilistic relational models, in: S. Dzeroski, N. Lavrac (Eds.), Relational Data Mining, Springer-Verlag, 2001.

[28] B. Taskar, P. Abbeel, D. Koller, Discriminative probabilistic models for relational data, in: Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI), 2002.

[29] L. Liao, D. Fox, H. Kautz, Location-based activity recognition, in: Advances in Neural Information Processing Systems (NIPS), 2005.

[30] L. Liao, D. Fox, H. Kautz, Hierarchical conditional random fields for GPS-based activity recognition, in: S. Thrun, H. Durrant-Whyte, R. Brooks (Eds.), Robotics Research: The Eleventh International Symposium, Springer Tracts in Advanced Robotics (STAR), Springer Verlag, 2006.

[31] V. Gogate, R. Dechter, C. Rindt, J. Marca, Modeling transportation routines using hybrid dynamic mixed networks, in: Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI), 2005.

[32] M. Cielniak, G.; Bennewitz, W. Burgard, Where is ...? learning and utilizing motion patterns of persons with mobile robots, in: Proc. of the International Joint Conference on Artificial Intelligence (IJCAI), 2003.

[33] L. Liao, D. Fox, J. Hightower, H. Kautz, D. Schulz, Voronoi tracking: Location estimation using sparse and noisy sensor data, in: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2003.

[34] J. Letchner, D. Fox, A. LaMarca, Large-scale localization from wireless signal strength, in: Proc. of the National Conference on Artificial Intelligence (AAAI), 2005.

[35] J. Krumm, L. Williams, G. Smith, SmartMoveX on a Graph - An Inexpensive Active Badge Tracker, in: Proceedings of the 4th international conference on Ubiquitous Computing, 2003.

[36] V. Guralnik, J. Srivastava, Event detection from time series data, in: 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999.