Adaptive Learning of Semantic Locations and Routes

Keshu Zhang, Haifeng Li, Kari Torkkola, and Mike Gardner
Motorola Labs
2900 S Diablo Way, Tempe, AZ 85282
keshu.zhang@motorola.com

ABSTRACT

Adaptation of devices and applications based on contextual information has a great potential to enhance usability and mitigate the increasing complexity of mobile devices. An important topic in context-aware computing is to learn semantic locations and routes of mobile device users. Several batch methods have been proposed to learn these locations. However, such offline methods have very limited usefulness in practice. This paper describes an online adaptive approach to learn user's semantic locations. The proposed method models user's GPS data as a mixture of Gaussians, which is updated by an online estimation. The learned Gaussian mixture is then evaluated to determine which components most likely correspond to the important locations based on *a priori* probabilities. With learned semantic locations, we also propose a minimax criterion to discover user's frequent transportation routes, which are modeled as sequences of GPS data. Finally, we describe an application of the proposed methods in a cell phone based automatic traffic alert system.

1. INTRODUCTION

Context aware applications can sense, infer, and predict a user's environment in order to provide services that benefit the user by automating processes, providing relevant information, and predicting events or behavior. With the increasing computing power of mobile devices and the increase in bandwidth for wireless communications, mobile devices become a logical platform for context aware applications. Context aware applications for mobile devices can fall into several different categories including, but not limited to:

- Adapt. Adaptive applications alter the state of the mobile device depending upon context. For example, the cell phone profile may be changed to the vibration mode automatically when the user is a meeting room.
- Search. Context aware search applications allow users to search for more information, which can include looking up a specific topic on the Internet, finding relevant points of interest, and finding locations of friends. For example, the user location may be used as a search filter to make the output from Internet searches more relevant to the user.
- Assist. Assistive applications aid the user in completing a task more efficiently and smoothly. For example, an intelligent calendar system can send a reminder that takes into account the user location, meeting location, and traffic condition. Assistive applications can be proactive in determining tasks to be completed and offer information and advisories to aid the user. For example, most frequently called numbers may be listed near the top of the phone contact list to facilitate the dialing process.

AUTONOMICS 2007, 28-30 October 2007, Rome, Italy

DOI 10.4108/ICST.AUTONOMICS2007.2231

An example of an assistive application is a navigation and/or traffic advisory system. Currently, personal or car navigation systems are accurate and easy to use for navigation purposes and they may also provide real-time traffic information. In order to get directions from location A to location B, the user usually enters these locations manually, which may be a time-consuming and visually demanding task. The traffic and accident information obtained through the system from the Internet or satellite radio may be general in nature and not tailored to the specific routes that the user will take.

One does not use a navigation system for daily routines – the commute routes are familiar to the user. However, one typically always carries a personal cell phone while driving. What we propose is to let the cell phone learn these daily commute routes together with when the routes are taken. The device will then automatically fetch traffic conditions along possible routes in anticipation (or after detection) of the user starting his/her commute, and inform the user if there are any problems. The key is that no effort on the part of the user is required to set up or operate the application. Useful information just magically appears on the phone screen at the time when it is needed.

Krumm and Horvitz and their colleagues have attempted to predict user's routes from travel history [8, 9]. We adopt a similar approach to learn user's usual locations and routes from the usage history, and predict likely routes based on the location and time information. Unlike Krumm and Horvitz, we do not attempt to handle previously unseen destinations.

The resulting traffic advisory system is a functional prototype which uses time and location information in conjunction with learned patterns of user behavior to determine when a user is leaving a location for a learned destination. The application reviews past routes that the user has taken to traverse from the current location to the predicted destination and checks available traffic monitoring services to determine if there are any unusual alerts. Only if there is such an unusual alert will the application provide the information to the user.

The most crucial component of such a system is learning the locations that are important to a user (we call these *semantic locations*.) Moreover, the learning needs to be invisible and adaptive, starting as the user installs the application on his/her cell phone, continuing in the background as long as the application remains installed. We describe first previous work in location learning in Sec. 2, followed by our approach to adaptive learning of user's important locations in Section 3. This is the main contribution of the paper. We then discuss route learning and roure comparison in Section 4. Section 5 describes the actual traffic advisory application and its implementation. Section 6 concludes the paper.

2. PREVIOUS WORK

Several batch methods have been proposed to discover user's important locations. These methods try to detect important locations through geographic information systems (GIS) or machine learning methods. For example, a landmark-based location system uses cell towers of a GSM phone network to learn important places in a user's daily routine [15]. Although this approach uses existing infrastructure and is cost-efficient, the resolution of the derived places is very coarse (from about hundreds meters to a few kilometers). Therefore, GPS receivers may be employed to identify important locations instead. Some early work on location extraction with GPS uses loss of signal to infer important indoor locations [2]. That is, if GPS signal disappears and then reappears around a small region, the region is regarded as a location. Such an approach is sufficient to identify some small indoor locations such as home. However, it does not account for larger indoor locations (e.g. office complex), and is prone to generating false positives due to many possible outdoor GPS shadows. Driving speed can be used to identify important locations. For example, as a part of their work on identifying a user's route, Patterson *et al.* can also infer mobile places, as well as the location of parking lots and bus stops with the help of real-world knowledge of bus schedules and stop locations, along with acceleration and turning speed information [13]. Liao *et al.* use mode-changes such as GPS signal loss and acceleration peaks to identify frequent locations in an unsupervised manner [12]. Learning the important locations can also be formulated as a clustering problem [14, 16]. Clearly, the staying time is an important factor to identify locations in this approach. Note that traditional methods such as K-means and Gaussian mixture model are not sufficient in this situation because they require the number of clusters to be known in advance, which is a serious limitation in practice.

3. ADAPTIVE LOCATION LEARNING

We describe now an approach to important location learning that overcomes the cited disadvantages, is lightweight enough to run on a cell phone, is adaptive, and operates on a continuous stream of incoming location data. An important location is here defined as a location where the user spends the most of his/her time, or most often is engaged in some activities, such as placing calls or sending messages.

3.1 Initial visualizations of the location data

Before any larger scale effort in data analysis, it usually pays off to visualize the data in question. We present some illustrations of the location data to justify the viability of location learning and route learning.

Data was collected from fourteen participants. These participants represent three groups of interest: students, office employees, and independents. The student group consists of six undergraduate students. The office employee group consists of five persons who worked in regular business days at fixed location. The independents group includes three persons who were either stay-at-home parents or business people who did not work consistently in an office environment (e.g. realtors).

Each participant was provided with a commercially available mobile phone equipped with Bluetooth, along with a separate Bluetooth GPS receiver to acquire location data. Data logging software was installed on the mobile phones. GPS location and Cell ID were logged every 30 seconds for a period of 2-3 months.

We depict one user's location data in Fig. 1. Time acts in the visualization as the z-coordinate. This figure illustrates two important facts. First, there are a small number of seemingly important locations where the phone usage concentrates. These become now explicit as "columns of activities" in Figure 1. We can see that there are four major location clusters of phone activities. Second, the user's daily commute routes, as well as weekend routines, become visible as repeating loops stacked on top of each other. Because of this regularity, Figure 1 suggests that it is not only possible to learn the routes from the data but also to predict them.

3.2 GPS Data Preprocessing

The semantic location discovery method is based on GPS data, which are sampled every 30 seconds in our study. However, GPS signal may get lost due to various reasons, such as entering into buildings or concrete canyons in urban areas or because GPS device powers off. Therefore, re-sampling GPS data is important to obtain equally sampled data without losing the important location clusters. In the re-sampling procedure, the void GPS data are made up by repeating the last valid GPS reading, if the area where GPS signal disappeared and reappeared is covered by the same group of cell IDs. The detailed re-sampling procedure is depicted in Fig. 2, where α is the GPS sampling rate , the two nearest GPS sampling times are t_{i+1} and t_i, and the corresponding GPS locations are g_{i+1} and g_i. Also, e_k denotes the time that GPS reading is invalid, and s_j denotes the last sampling time in one log file.

3.3 Adaptive Location Clustering

After re-sampling, the GPS data will be fed to the learning algorithm. Only GPS readings with speed less than 5

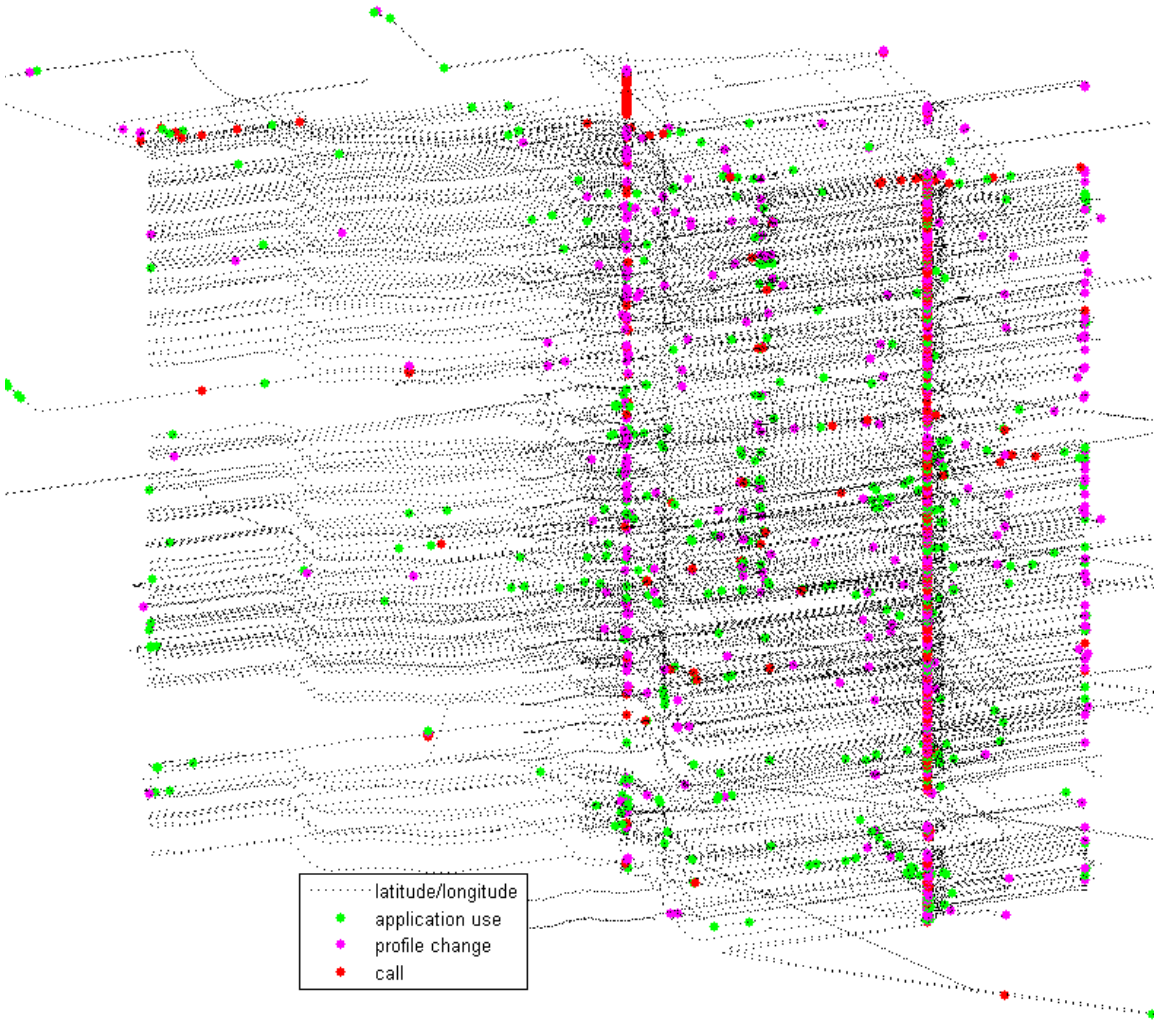

Figure 1: Visualization of one subject's data base. User's GPS coordinate traces (x,y) and time (z) plotted together with certain phone application events. Time covers about three months and is increasing from bottom to top.

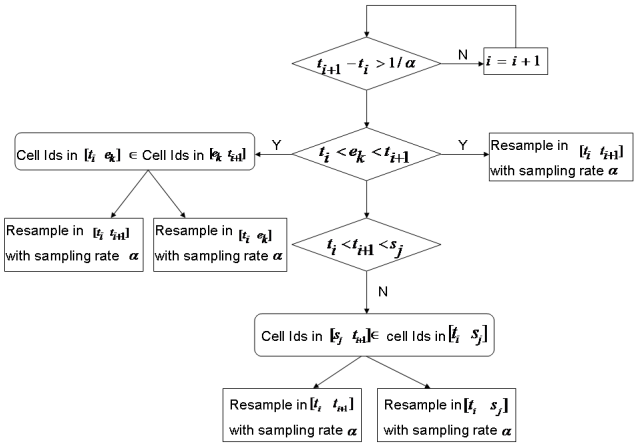

Figure 2: Decision rule for GPS data resampling.

miles/hour will be considered to learn the stationary locations.

The adaptive algorithm is a sequential method that processes only a small amount of GPS data each time. The algorithm continuously adapts the learned model to new GPS readings. We model the GPS data as a mixture of locations with noise, where each location follows Gaussian distribution. Given new GPS readings, an on-line algorithm is employed to update the model, i.e. adding/deleting/merging locations, updating the parameters of the Gaussian mixture model, etc. Consider a set of sequential GPS readings $\{g_1, \ldots, g_t\}$ in time order, where

$$g_i = [x_i, y_i]' \qquad 1 \leq i \leq t$$

and x, y are GPS coordinates. Assume at the moment t that the true but unknown density of GPS value g is of the

form

$$p(g|\theta_t) = \sum_{\ell=1}^{N_t} \pi_t^\ell \phi(g; \mu_t^\ell, \Sigma_t^\ell)$$

where $N_t < \infty$ is the number of components which may vary with time, parameter vector $\theta_t = \{\pi_t^1, \mu_t^1, \Sigma_t^1, \ldots, \pi_t^{N_t}, \mu_t^{N_t}, \Sigma_t^{N_t}\}$; the nonnegative mixing coefficients π_t^ℓ $(\ell = 1, \ldots, N_t)$ sum to unity, and

$$\phi(g; \mu_t^\ell, \Sigma_t^\ell) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_t^\ell|^{\frac{1}{2}}} \exp[-\frac{1}{2}(g - \mu_t^\ell)'(\Sigma_t^\ell)^{-1}(g - \mu_t^\ell)]$$

is the Gaussian probability density function with two-dimensional mean μ_t^ℓ and covariance Σ_t^ℓ. For simplicity, the covariance matrix is assumed to be diagonal.

At time t, the number of Gaussians N_t in the mixture model represents the number of significant location candidates, the weight of Gaussians π_t^ℓ $(\ell = 1, \ldots, N_t)$ denote the significance of the learned locations, and the mean μ_t^ℓ and variance value Σ_t^ℓ in each Gaussian specifies the important location center and size. Below, we will describe how to adapt the parameters as well as the structures of the mixture model with the new incoming observations.

Location Update Rule

If there is no change in the GPS value distribution, the number of components N_t in the mixture will be constant, and all observations up to the current time $\mathbf{g}_{1:t} = \{g_1, \ldots, g_t\}$ can be used as the training set to estimate the parameter vector θ_t. Denote an estimate for $p(g|\theta_t)$ with the estimated parameter vector $\hat{\theta}_t$ as $\hat{p}_t(g) = p(g|\hat{\theta}_t)$. The EM algorithm [6] is a maximum likelihood estimation of these parameters. Titterington [7] presents a recursive procedure to update θ_t for the normal distribution ϕ. When a new observation arrives, the training set $\mathbf{g}_{1:t}$ is updated, and $\hat{p}_t(g)$ is re-estimated.

Let $\beta_t = \frac{1}{t}$ and $\rho_t^\ell = \pi_{t-1}^\ell \frac{\phi^\ell(g_t)}{\hat{p}_t(g_t)}$, then

$$\begin{aligned}
\pi_t^\ell &= \pi_{t-1}^\ell + \beta_{t-1}\left[\rho_t^\ell - \pi_{t-1}^\ell\right] \\
\mu_t^\ell &= \mu_{t-1}^\ell + \frac{\rho_t^\ell \beta_{t-1}}{\pi_{t-1}^\ell}(g_t - \mu_{t-1}^\ell) \\
\Sigma_t^\ell &= \Sigma_{t-1}^\ell + \frac{\rho_t^\ell \beta_{t-1}}{\pi_{t-1}^\ell}\left[(g_t - \mu_{t-1}^\ell)(g_t - \mu_{t-1}^\ell)^T - \Sigma_{t-1}^\ell\right]
\end{aligned} \tag{1}$$

We call the procedure (1) *update rule*, denoted as $\hat{\theta}_t = U_t(g_t, \beta_{t-1}, \hat{\theta}_{t-1})$. Convergence results regarding the recursive update rule $U_t(\cdot)$ have been given in [7]. This procedure is generally also called the recursive EM algorithm.

The mixture weights in the model represent the sizes of the clusters, or the number of data points in a cluster. They denote the proportion of time the user spends at each location. However, this definition ignores locations that user frequently visits but stays for a brief time (e.g. dropping child off for school every day). Meanwhile, a location (e.g. travel attraction) that user visits rarely but stays for a long time, would be temporarily misclassified as an important location. To solve the problem, we need also record the number of visits for each location. The weight f_t^ℓ of the visit frequency for each location needs also be updated online.

Denote v_t^ℓ as the visit detection for location cluster ℓ, then

$$v_t^\ell = \begin{cases} 1 & \text{if user just enter into location } \ell \text{ boundary} \\ 0 & \text{otherwise} \end{cases}$$

Therefore, at time t, the total number of visits for all locations is $M_t = M_{t-1} + \sum_{\ell=1}^{N_t} v_t^\ell$, and $M_0 = 0$. Similarly to the update procedure for the weight in the mixture model, the weight of the visit frequency f_t^ℓ for location ℓ can also be updated as

$$f_t^\ell = f_{t-1}^\ell + \gamma_{t-1}[v_t^\ell - f_{t-1}^\ell] \tag{2}$$

where $\gamma_t = \frac{(M_t - M_{t-1})}{M_t}$.

Now (1) and (2) together are denoted as the new update rule $\hat{\theta}_t = U_t(g_t, \beta_{t-1}, \gamma_{t-1}, \hat{\theta}_{t-1})$.

Then the importance of a location is defind by the adjusted weight

$$\varpi_t^\ell = \alpha \pi_t^\ell + (1 - \alpha) f_t^\ell$$

where $\alpha \in [0, 1]$. With this linear combination, both how long and how often the user stays and visits in a location are considered in determining the importance of a location. Parameter α controls the compromise between the two factors, usually we use $\alpha = 0.5$ to treat the visit time and the visit frequency with equal importance.

If an approximation of the density $p_t(g)$ by a finite mixture is used, the number of components N_t and an initial estimate must be chosen. We describe next how to choose N_t from the data in a recursive manner. The approach taken by an adaptive mixtures estimator is to recursively adapt not only the parameters, as above, but also the number of components needed to fit the data. The adaptive mixtures approach is designed to allow the number of terms to grow, but at a much slower rate than that of a kernel estimator. The adaptive mixtures approach is much less computationally and memory intensive in practice, can produce a more useful small sample estimator, and allows general consistency results [7].

Addition Rule

The decision to add a component is made by checking the Mahalanobis distance from the observation to each of the terms; if the minimum of these exceeds a threshold (called the create threshold, $T_D = 2.5$) then the observation point is in some sense too far away from the existing terms, and a new term should be created. The square of the Mahalanobis distance between a point g_t and a component with the mean μ_{t-1}^ℓ and the standard deviation Σ_{t-1}^ℓ is defined by $M(z_t; \mu_{t-1}^\ell, \Sigma_{t-1}^\ell) = (g_t - \mu_{t-1}^\ell)'(\Sigma_{t-1}^\ell)^{-1}(g_t - \mu_{t-1}^\ell)$, $1 \le \ell \le N_{t-1}$. A binary decision function $D_t(\cdot)$ decides when to add a component in the mixture.

$$\begin{aligned}
D_t(g_t, \hat{\theta}_{t-1}) &= 1 \quad \text{if} \min_{1 \le \ell \le N_{t-1}} M(g_t; \mu_{t-1}^\ell, \Sigma_{t-1}^\ell) > T_D \\
D_t(g_t, \hat{\theta}_{t-1}) &= 0 \quad \text{if} \min_{1 \le \ell \le N_{t-1}} M(g_t; \mu_{t-1}^\ell, \Sigma_{t-1}^\ell) \le T_D
\end{aligned}$$

Creation Rule

A new observation is checked against the existing K Gaussian distributions. If none of the K distributions match the current GPS value, a new Guassian component is added to the mixture. Assuming that the system has decided to add a new component ($D_t(g_t, \hat{\theta}_{t-1}) = 1$), a creation rule, $\hat{\theta}_t = \vartheta_t(g_t, \beta_t, \gamma_t, \hat{\theta}_{t-1})$, can be derived from the fact that the kernel estimator based on t observations is closely related to the kernel estimator based on $t-1$ observations.

$$\begin{aligned} \mu_t^{N_t} &= g_t \\ \Sigma_t^{N_t} &= \Sigma_t^0 \\ \pi_t^{\ell} &= \pi_{t-1}^{\ell}(1-\beta_t)(\ell = 1, \ldots, N_{t-1}) \\ \pi_t^{N_t} &= \beta_t \\ f_t^{\ell} &= f_{t-1}^{\ell}(1-\gamma_t)(\ell = 1, \ldots, N_{t-1}) \\ f_t^{N_t} &= \gamma_t \end{aligned} \tag{3}$$

Where $N_t = N_{t-1} + 1$. Σ_t^0 may be user defined or derived from the terms in the neighborhood of the observations. Creation rule can be also used to initialize the mixture density with $\mu_1^1 = g_1$, $\Sigma_1^1 = \Sigma_t^0$ and $\pi_1^1 = 1$.

The adaptive procedure proposed by Priebe [5] has the fundamental assumption that $\{g_1, \ldots, g_t\}$ are drawn from the same distribution $p(g)$. For learning the important locations, the GPS distribution is time variant due to the changes in user's daily life. When the GPS distribution changes, the new components will always be added in. The complexity may not be controlled. An intelligent system also needs to prune the unnecessary terms to forget the locations that user stops visiting, and to merge close nearby terms. We describe these now in detail.

Pruning Rule

The purpose of the pruning rule is to discard the redundant components in the mixture which do not entertain current observations anymore. How to select the pruning candidates is based on the Bayesian hypothesis test.

Denote C_t^{ℓ} : ℓth Gaussian component belongs to the current important location candidates $\ell = 1, \ldots, N_t$. then $P(C_t^{\ell}) = \varpi_t^{\ell}$, and

$$p(g_t|C_t^{\ell}) = \frac{1}{(2\pi)^{n/2}|\Sigma_t^{\ell}|^{1/2}} \exp[-.5(g_t - \mu_t^{\ell})'(\Sigma_t^{\ell})^{-1}(g_t - \mu_t^{\ell})]$$

The probability that the observation g_t belongs to the ℓth Gaussian component in the mixture is

$$P(C_t^{\ell}|g_t) = \frac{p(g_t|C_t^{\ell})P(C_t^{\ell})}{p(g_k)}$$

Therefore, for the Gaussian component ℓ_1 and ℓ_2 in the mixture, if

$$P(C_t^{\ell_1}|g_t) > P(C_t^{\ell_2}|g_t) \tag{4}$$

the component ℓ_2 is more likely to be selected as the pruning candidate. According to the 3σ property of Gaussian distribution, $\exp[-.5(g_t - \mu_t^{\ell})'(\Sigma_t^{\ell})^{-1}(g_t - \mu_k^{\ell})] \to 0$ when $(g_t - \mu_t^{\ell})'(\Sigma_t^{\ell})^{-1}(g_t - \mu_k^{\ell}) > 9$ for g_t that does not belong to the component ℓ. Eq. (4) can be written as

$$\pi_t^{\ell_1} > \pi_t^{\ell_2}$$

Because we adjust the location cluster weight by also considering the visiting frequency of the location, the candidate components to be discarded from the mixture model are selected from those that $D_t(g_t, \hat{\theta}_{t-1}) = 1$ with $\varpi_t^{\ell} < T_P$, where T_p is a pre-set threshold. After dropping the existing terms in the mixture, the maintained mixture model weights π_t^{ℓ} ($\ell = 1, \ldots, N_t$) need to be re-normalized. We denote this pruning procedure as $\hat{\theta}_t = \mathcal{P}_t(\hat{\theta}_t)$.

Generally, one can pre-set the largest number $N_{\max}$ of Gaussians to be allowed in the mixture model according to the system requirement. When the mixture model meets this bound, as the decision rule detects that a new component needs to be added into the mixture model, we can replace the old component with the new one. The candidate component for replacing can also be chosen by the Pruning rule.

Pruning Threshold Selection

At time t_0, when the new component ℓ is added into the mixture model, we usually allow several days of adaptation to adjust visit time weight $\pi_{t_0}^{\ell}$ and visit frequency weight $f_{t_0}^{\ell}$. To determine a reasonable pruning threshold T_p, there are some general rules. As discussed above, the initial weights $\pi_{t_0}^{\ell}$ and $f_{t_0}^{\ell}$ for the new components are often set as $\bar{\beta}$ and $\bar{f}$ respectively. Meanwhile, the current learning rates are adjusted as $\beta_{t_0} = \bar{\beta}$ and $f_{t_0} = \bar{f}$. Therefore, if the ℓth component in the mixture does not entertain the observations for $T = 7$ days, it has

$$\begin{aligned} \pi_{t_0+T}^{\ell} &= (1-\beta_{t_0+T}) \cdots (1-\beta_{t_0+1})(1-\beta_{t_0})\pi_{t_0}^{\ell} \\ f_{t_0+T}^{\ell} &= (1-\gamma_{t_0+T}) \cdots (1-\gamma_{t_0+1})(1-\gamma_{t_0})\pi_{t_0}^{\ell} \end{aligned}$$

since $\beta_{t_0+i} = \frac{\beta_{t_0}}{i\beta_{t_0}+1}$, then

$$\begin{aligned} \pi_{t_0+T}^{\ell} &= (1 - \frac{\beta_{t_0}}{T\beta_{t_0}+1})(1 - \frac{\beta_{t_0}}{(T-1)\beta_{t_0}+1}) \cdots (1-\beta_{t_0})\pi_{t_0}^{\ell} \\ &= \frac{(T-1)\beta_{t_0}+1}{T\beta_{t_0}+1} \cdot \frac{(T-2)\beta_{t_0}+1}{(T-1)\beta_{t_0}+1} \cdot \cdots \cdot (1-\beta_{t_0})\pi_{t_0}^{\ell} \\ &= \frac{(1-\beta_{t_0})}{T\beta_{t_0}+1} \cdot \pi_{t_0}^{\ell} = \frac{(1-\bar{\beta})}{T\bar{\beta}+1} \cdot \bar{\beta} \end{aligned}$$

Similarly,

$$f_{t_0+T}^{\ell} \geq \frac{(1-\bar{f})}{T\bar{f}+1} \cdot \bar{f}$$

Therefore, we may specify the pruning threshold as

$$T_P \geq \alpha \frac{(1-\bar{\beta})\bar{\beta}}{(T\bar{\beta}+1)} + (1-\alpha)\frac{(1-\bar{f})}{T\bar{f}+1} \cdot \bar{f}$$

Merging Rule

The merging rule $\hat{\theta}_t = M(\hat{\theta}_t)$ combines two Gaussian components into a single Gaussian when the two components in the mixture model are very close to each other.

Suppose the two Gaussian components are

$$p(g|C_{\ell_1}) = N(g, \mu^{\ell_1}, \Sigma^{\ell_1}) \quad P(C_{\ell_1}) = \pi^{\ell_1}$$
$$p(g|C_{\ell_2}) = N(g, \mu^{\ell_2}, \Sigma^{\ell_2}) \quad P(C_{\ell_2}) = \pi^{\ell_2}$$

In probability theory, Kullback-Liebler Divergence is a quantity which measures the difference between two probability distributions. For two Gaussian distributions

$$\begin{aligned} &KL(p(g|C_{\ell_1}); p(g|C_{\ell_2})) + KL(p(g|C_{\ell_2}); p(g|C_{\ell_1})) \\ &= 0.5(|(\Sigma^{\ell_2})^{-1}\Sigma^{\ell_1}| + |(\Sigma^{\ell_1})^{-1}\Sigma^{\ell_2}|) \\ &+ 0.5(\mu^{\ell_1} - \mu^{\ell_2})'[(\Sigma^{\ell_1})^{-1} + (\Sigma^{\ell_2})^{-1}](\mu^{\ell_1} - \mu^{\ell_2}) - n \end{aligned}$$

When

$$KL(p(g|C_{\ell_1}); p(g|C_{\ell_2})) + KL(p(g|C_{\ell_2}); p(g|C_{\ell_1})) < \varepsilon$$

we can merge two Gaussian components $p(g|C_{\ell_1})$ and $p(g|C_{\ell_2})$ into one $p(g|C_{\ell'}) = N(g, \mu^{\ell}, \Sigma^{\ell})$ and obviously $\pi^{\ell} = \pi^{\ell_1} + \pi^{\ell_2}$, $f^{\ell} = f^{\ell_1} + f^{\ell_2}$and

$$\begin{aligned} \mu^{\ell} &= E(g|C_{\ell}) = \frac{\mu^{\ell_1}\pi^{\ell_1} + \mu^{\ell_2}\pi^{\ell_2}}{\pi^{\ell_1} + \pi^{\ell_2}} \\ \Sigma^{\ell} &= E(g^2|C_{\ell}) - \mu^{\ell}(\mu^{\ell})' \\ &= \frac{(\Sigma^{\ell_1} + \mu^{\ell_1}\mu^{\ell_1\prime})\pi^{\ell_1} + (\Sigma^{\ell_2} + \mu^{\ell_2}\mu^{\ell_2\prime})\pi^{\ell_2}}{\pi^{\ell_1} + \pi^{\ell_2}} - \mu^{\ell}\mu^{\ell\prime} \end{aligned}$$

As we have seen, the pruning rule and merging rule are used to eliminate the unnecessary components and merge similar components. The two described procedures can keep the most efficient model for a time varying GPS distribution at the same time saving computation and memory.

Learning Rate Adjustment

In the above adaptive location clustering process, β_t is the learning rate for the parameter adaptation of the Gaussian mixture model. The convergence result regarding the recursive update formula (1) with β_t converging to 0 have been given by [7]. Since this is an online location learning process, a small learning rate will result in slow adaptation to the current situation, i.e., any new location cluster will need a long time to become significant in the system, and the already learned location clusters are very unlikely to be pruned from the system. To solve this problem, we set a lower bound for the learning rate β_t . When $\beta_t < \beta_L$, reset $\beta_t = \bar{\beta}$. Here $\bar{\beta} > \beta_L$. Also, when the system decides to create a new location cluster, i.e., $D_t(g_t, \hat{\theta}_{t-1}) = 1$, and $\beta_t < \bar{\beta}$, we also reset $\beta_t = \bar{\beta}$. Now, all new location clusters are adjusted with the same learning rate from beginning.

The proposed learning rate adjustment makes the system more flexible in adjusting to changes. Therefore, user's daily life will be timely updated in the system. The same technique can be applied on the learning rate γ_t to adapt the weight of visit frequency.

3.4 Significant Location Identification

In location learning process, the location clusters are represented by the major components in the mixture model. The adjustment weights in the mixture model denote the significance of the locations. However, not all location clusters belong to the significant locations since some location clusters have been recently added to the model and need time to justify themselves. The location clusters only provide the candidates of the important locations. To determine the significant locations, at time t, we need evaluate the adjustment weight $\varpi_t^1 \geq \varpi_t^2 \geq \cdots \geq \varpi_t^{N_t}$, and pick the top N corresponding location clusters such that $\sum_{\ell=1}^{N} \varpi_t^{\ell} \geq 95\%$ and $\sum_{\ell=1}^{N-1} \varpi_t^{\ell} < 95\%$, where $0 < N \leq N_t$.

The described adaptive mixture method automatically generates the location clusters. The online update procedure adjusts model parameters as well as the number of components in the mixture. Thus, by choosing the prominent clusters in the mixture, the significant locations are identified.

4. LEARNING AND PREDICTION OF ROUTES

We use raw GPS data sequences to represent routes. A GPS string between an important location pair is denoted as a route. The starting point and ending point of the route involves detecting of the user leaving one important location boundary and entering into another important location boundary. In fact, through location clustering, the cluster boundaries also serve as the important location boundaries.

4.1 Route Learning

Although it is easy to extract GPS data sequence between two locations as routes, it is not straightforward to determine if two sequences represent the same routes. Note that people often have several different daily routes for the same location pair. The system has to distinguish them for providing traffic information by comparing GPS data sequences.

To solve the problem, we developed a MinMax criterion to compare two routes. Suppose two routes of GPS sequences are

$$R_1 = \{g_1^1, g_2^1, \ldots, g_n^1\} \text{ and } R_2 = \{g_1^2, g_2^2, \ldots, g_m^2\}$$

The best match of R_1 in terms of R_2 is

$$\hat{R}_1 = \{\hat{g}_1^1, \hat{g}_2^1, \ldots, \hat{g}_n^1\}$$

where $\hat{g}_i^1 = \arg\min_{g_j^2} ||g_i^1 - g_j^2||$. Similarly, $\hat{R}_2 = \{\hat{g}_1^2, \hat{g}_2^2, \ldots, \hat{g}_m^2\}$ is the best match R_2 in terms of R_1, where $\hat{g}_j^2 = \arg\min_{g_i^1} ||g_i^1 - g_j^2||$.

We regarded two routes as the same if

$$\max ||g_i^1 - \hat{g}_i^1|| < d_1 \text{ and } \max ||g_j^2 - \hat{g}_j^2|| < d_2$$

where d_1 and d_2 are the largest sampling intervals of R_1 and R_2, respectively.

With the adaptive location clustering, the significant locations are gradually discovered and updated. Once an important location pair is identified, the commute routes between them are recorded and learned. Through the route comparison, the unique ones are saved in the database. The unique routes do not distinguish the directions, i.e., the route that

user moves from place A to place B or moves from place B to place A are treated as the same. The frequencies of the routes are counted in terms of the starting location and the destination in an hourly base.

5. APPLICATION TO A CELL PHONE BASED TRAFFIC ADVISOR SYSTEM

Once we have the capability for route prediction, a traffic advisory system can be built by fetching real-time traffic conditions along the predicted route, and presenting this information to the user.

5.1 Design of the application

Ideally, the traffic advisor is designed to assist users during transition times as the user is transitioning out of one location and into a vehicle environment. Previous research has shown that messages from context aware devices are better received during transitions [17], and in our case it is during transitions that the information provided by the application is of the greatest relevance.

The application is designed to automatically detect when the user is leaving a location and to predict where they are likely headed and along which route. When the detection occurs, traffic information will be fetched. To adopt a user-centric approach and minimize the obtrusiveness, traffic history may be studied to derive traffic averages and variances and traffic alerts are only provided when current or predicted traffic flow exceeds the average by a certain margin (e.g., one standard deviation above the average flow). If a traffic alert is detected that is out of the ordinary for that route and time of day (i.e., if the route the user takes is always delayed by ten minutes due to heavy traffic, the user will not always be reminded of this), then the user will receive an audio or haptic alert, depending upon the current profile settings of the phone. In addition, a pop-up window will appear alerting the user that there is an alert, and offering the user the choice of ignoring the alert or obtaining more information. If the user selects the latter, more detailed information on the alert (e.g. type of incident and location) will be provided with an option for the application to recommend a new route to the predicted location.

5.2 Implementation of the application

The developed prototype is a client-sever system. The client software is running on a cell phone. The client collects GPS data (through Bluetooth connection to a receiver) and cell IDs and learns semantic locations and routes between them with our adaptive algorithm. An easy to use user interface is also developed to alert users about problematic traffic situations (Fig. 3).

The server side provides map, real-time traffic, and other information to clients based on J2EE. The communication between clients and the server side is based on standard HTTP protocol. An option to use free traffic services available in the Internet is provided, too. The architecture of the system is depicted in Fig. 4.

5.3 Evaluation of the adaptive location learning

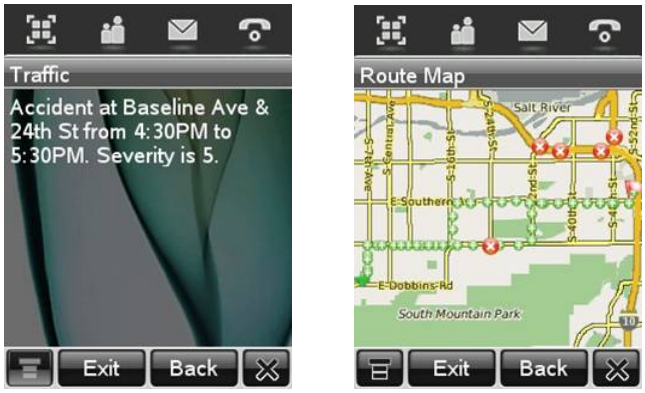

Figure 3: Screen shots of the application. Left side depicts a textual pop-up alert, right side the map view of the alert.

To evaluate the location learning feature in the traffic advisory system, we compare the developed adaptive location learning algorithm to a batch learning method. As the batch method we employ the DBSCAN (density-based clustering) algorithm, because it can automatically find the number of clusters, identify outliers, it works well for arbitrary-shaped clusters, and is also very efficient for large datasets [11, 10].

In the study, we first evaluate the algorithms on one office employee who worked in regular business days at a fixed location. Both batch and adaptive algorithms are applied to the re-sampled GPS data, the comparison results for important location discovery are generated for the first day, the first week, and the first month. The visualization results of the discovered significant locations are shown in Figures 5, 6, and 7. The corresponding weights of each discovered location for both batched and adaptive methods are listed in Table 1.

After the first day, the discovered locations by both batched and adaptive methods match very well. The adjustment weight ω generated by the adaptive method provides the significance measure for the locations by considering the visit duration as well as the visit frequencies. For example, in the first day, the adaptive method finds that the "School", where the user drops his child off is comparably as important as the "Gym" (Table 2). Although the visit time of the "School" is much shorter than the "Gym", the "School" has been visited twice for that day. In the first week, compared with the batch method, the adaptive method discovered location L_0 instead of L_1 (Fig. 6). Since the adaptive method adjusts the parameters online, location L_1 which was visited early in the week is already discarded from the significant location candidates due to the short visit time and low visit frequency (only one visit for that week). Instead, location L_0 which has been visited most recently had higher initial weight and then becomes more important than location L_1. The same reason explains results obtaining location L_3 instead of L_2 from adaptive learning method on first month's data (Fig. 7). Meanwhile, both algorithms can discover several nearby locations in certain large areas, i.e. the "University" campus area and the "Gym".

This comparison shows that the proposed location learning

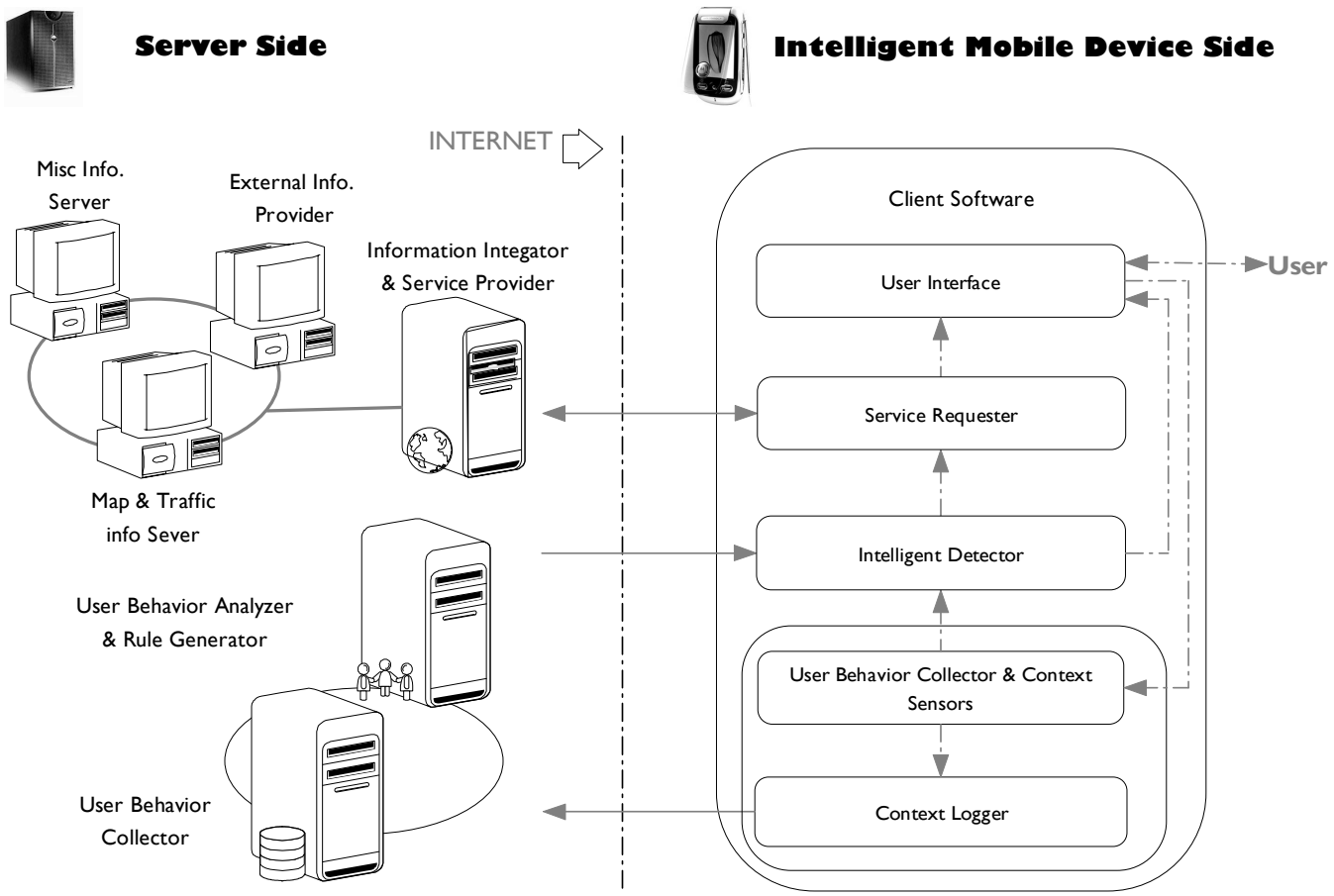

Figure 4: Architecture of the server/client system. The mobile device is on the right side of the figure. This figure depicts also the possibility of the mobile device sending collected context data (GPS in this case) to a server (bottom), which trains the location models, and sends back to the mobile device. As we have described in this paper, all adaptation can be done at the mobile device, and this possibility is currently reserved for more complex cases in the future.

algorithm is effective in location identification and can adapt to the user's daily life.

5.4 Evaluation of Route Learning

As the important locations become available, the transportation routes between the location pairs, i.e. GPS strings, are recorded in the system. With the proposed Min-Max criterion, the unique routes are detected and the most frequent ones are discovered. In our study on one user's data, after two month learning, 47 unique routes from a total of 170 routes are obtained.

The route frequency is derived by considering all different routes starting from Location A to Location B. As an example, we show the route frequencies from "home" to "office" in Table 2, and those from "office" to "home" in Table 3.

Naturally, the most frequent route is simple to obtain once we have the frequency for each route between semantic locations. We select the most frequent routes between location A to B to cover 95% of total routes and the maximum number of the most frequent ones is three for the application. Figure 8 displays the most frequent routes for different hour and different location pairs. With these thresholds none of the routes that the users perceived important were discarded.

5.5 Evaluation of the implementation

All specific aspects of the implementation of the design of the traffic advisor, as well as the overall acceptance of the application are currently being validated by real users from specific demographic areas of interest in a field evaluation of the application.

6. CONCLUSIONS

Context knowledge such as the user's needs, locations, and travel routes is a major enabler for providing a cell phone user with unobtrusive assistance. In this paper, the focus has been on a location-based traffic advisory system that is based on learning and predicting the user's patterns using data available in the mobile devices. We have demonstrated a successful implementation of a traffic condition assistant that learns user's important locations, routes between those, and learns to predict those routes. In order to maximize the system effectiveness and user acceptance, advisories are provided to the user only when there is a real need (e.g., a traffic problem on the predicted route).

	Locations		Home	Office	Gym	Univ.	School
first day	DBSCAN	π(weight)	0.8355		0.1583		0.0062
	Adaptive	π(weight)	0.7583		0.1889		0.0474
		f(frequency)	0.5714		0.1429		0.2857
		ω(adjust weight)	0.6649		0.1659		0.1666
first week	DBSCAN	π(weight)	0.8508	0.0214	0.0235 0.0804	0.0111	0.0051
	Adaptive	π(weight)	0.7910	0.0193	0.0965	0.0148	0.0083
		f(frequency)	0.4359	0.1026	0.1538	0.0256	0.1795
		ω(adjust weight)	0.6134	0.0609	0.1252	0.0202	0.0939
first month	DBSCAN	π(weight)	0.8333	0.0206	0.1073	0.0117	0.0049
	Adaptive	π(weight)	0.6227	0.1283	0.0075 0.1104	0.0038 0.0193 0.0383	0.0085
		f(frequency)	0.3557	0.1237	0.0258 0.1804	0.0103 0.0361 0.0928	0.1031
		ω(adjust weight)	0.4892	0.1260	0.0166 0.1454	0.0071 0.0277 0.0655	0.0558

Table 1: Significant location learning methods comparison

Route ID	4	8	25	26	27	28	29	30
frequency	0.6	0.1429	0.0286	0.1143	0.0286	0.0286	0.0286	0.0286

Table 2: Frequencies for routes from home to office

Route ID	8	28	35	36	37	38	39	40	41
frequency	0.4643	0.2143	0.03576	0.0357	0.0357	0.0714	0.0714	0.0357	0.0357

Table 3: Frequencies for routes from office to home

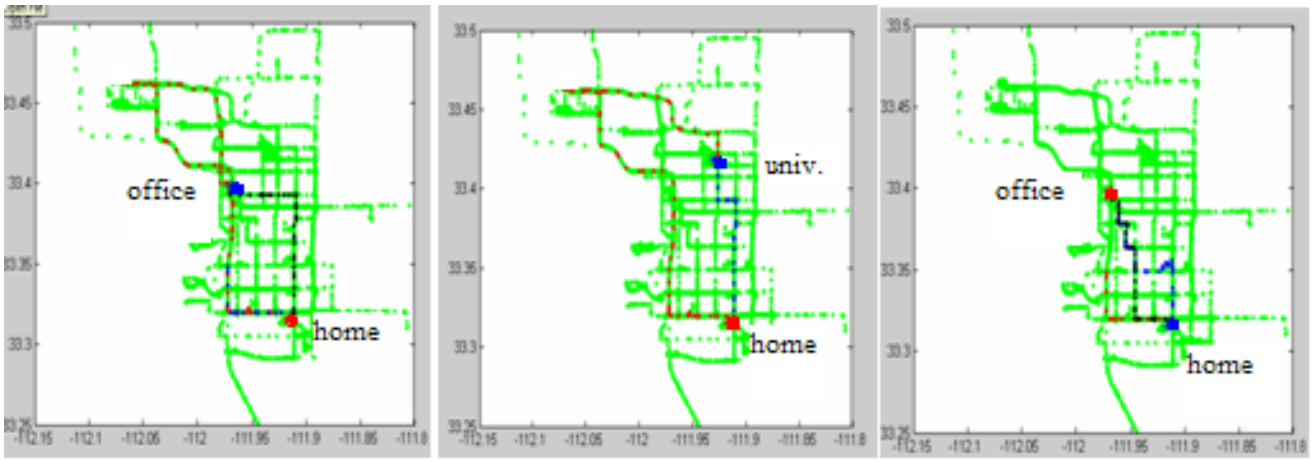

Figure 8: Illustration of one user's frequent routes.

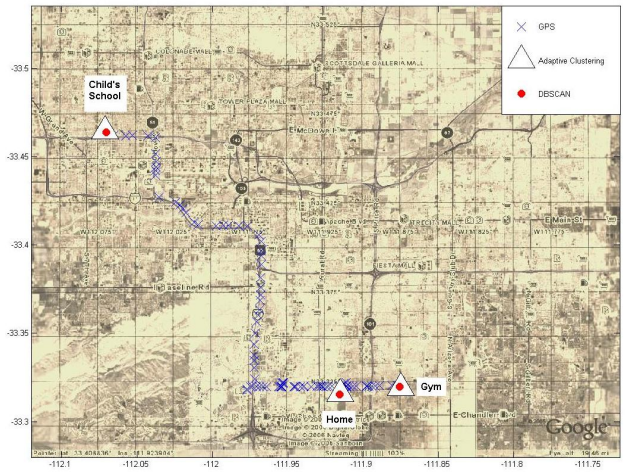

Figure 5: Visualization of the first day of one subject's data. User's GPS coordinates are plotted together with important locations.

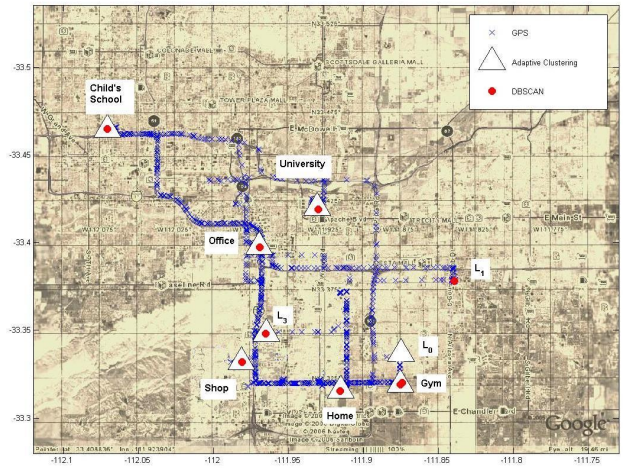

Figure 6: Visualization of the first week of one subject's data. User's GPS coordinates are plotted together with important locations.

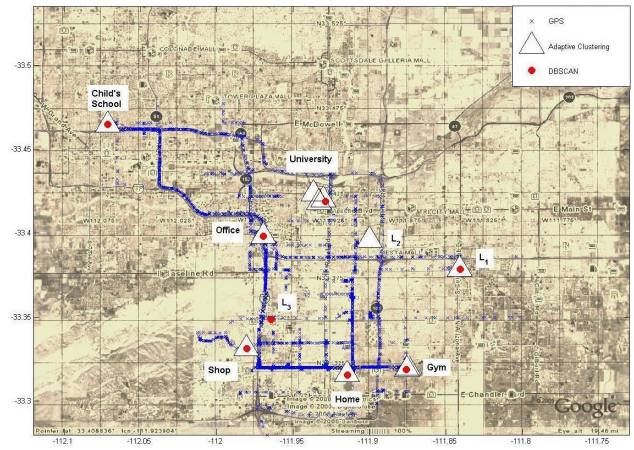

Figure 7: Visualization of the first month of one subject's data. User's GPS coordinates are plotted together with important locations.

We envision that context aware features like this including a user modeling component will become essential in supporting new applications and concepts that connect consumers with media and services through multiple devices in the office, car, or home.

7. REFERENCES

[1] E. Horvitz, A. Jacobs and D. Hovel, *Attention-sensitive alerting*, *Proc. of UAI'99 conference on Uncertainty in Artificial Intelligence*, Stockholm, Sweden, 1999, pp.305-313.

[2] N. Marmasse and C. Schmandt, *Location-aware information delivery with comMotion*, *Proc. of International Symposium on Handheld and Ubiquitous Computing*, Bristol, UK, Sep. 2000, pp. 157-171.

[3] N. Marmasse and C. Schmandt, *A User-Centered location model*, *Personal and Ubiquitous Computing*, vol 6, no. 5-6, 2002, pp. 318-32

[4] N. Marmasse, C. Schmandt and D. Spectre, *WatchMe: communication and awareness between members of a closely-knit group*, *Proc. of Ubicomp 2004: Ubiquitous Computing*, Sep. 2004. pp 214-231.

[5] C. E. Priebe, *Adaptive mixtures*, *Journal of American statistical Association,* 1994, pp. 796-806.

[6] A. P. Dempster and N. M. Laird and D. B. Rubin, *Maximum likelihood form incomplete data via the {EM} algorithm*, *Journal of Royal Statistical Society*, 1977, pp. 1-38.

[7] D. M. Titterington, *Recursive parameter estimation using incomplete data*, *Journal of Royal Statistical Society*, 1984, pp. 257-267.

[8] J. Krumm and E. Horvitz, *Predestination: Inferring Destinations from Partial Trajectories*, *Proc. of Ubicomp* 2006, Sep. 2006.

[9] J. Letchner and J. Krumm and E. Horvitz, *Trip Router with Individualized Preferences (TRIP): Incorporating Personalization into Route Planning*, *Eighteenth Conference on Innovative Applications of Artificial Intelligence (IAAI-06)*, July 2006.

[10] J. Tang and L. Meng, *Learning significant locations from GPS data with time window. Proc. of SPIE*, 2006, Vol. 6418.

[11] M. Ester et al., *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Proc. of KDD*, 1996

[12] L. Liao, et al., *Learning and Inferring Transportation Routines. In Proc. of AAAI*, 2004.

[13] D. J. Patterson, et al., *Inferring High-Level Behavior from Low-Level Sensors. In Proc. of Ubicomp*, 2003.

[14] R. Hariharan and K. Toyama. *Project Lachesis: Parsing and Modeling Location Histories. In Proc. of International Conference on Geographic Information Science*, 2004.

[15] K. Laasonen et al., *Adaptive On-Device Location Recognition. In Proc. of Pervasive*, 2004.

[16] J. H. Kang and W. Welbourne. *Extracting Places from Traces of Locations. In Proc. WMASH*, 2004.

[17] J. Ho and S. Intille, *Using Context-Aware Computing to Reduce the Perceived Burden of Interruptions from Mobile Devices. Proc CHI*, 2005.