Project Lachesis: Parsing and Modeling Location Histories

Ramaswamy Hariharan[1] and Kentaro Toyama[2]

[1] School of Information and Computer Science,
University of California, Irvine,
Irvine, CA 92697, USA
rharihar@ics.uci.edu
[2] Microsoft Research, One Microsoft Way,
Redmond, WA 98052, USA
kentoy@microsoft.com

Abstract. A datatype with increasing importance in GIS is what we call the *location history*–a record of an entity's location in geographical space over an interval of time. This paper proposes a number of rigorously defined data structures and algorithms for analyzing and generating location histories. *Stays* are instances where a subject has spent some time at a single location, and *destinations* are clusters of stays. Using stays and destinations, we then propose two methods for modeling location histories probabilistically. Experiments show the value of these data structures, as well as the possible applications of probabilistic models of location histories.

1 Introduction

A datatype with increasing importance in GIS is what we call the *location history*–a record of an entity's location in geographical space over an interval of time. In the past, location histories have been reconstructed by archaeologists and historians looking at migrating populations or census takers tracking demographics, at temporal resolutions of decades or centuries and spatial resolutions of tens or hundreds of kilometers. Recent advances in location-aware technology, however, allow us to record location histories at a dramatically increased resolution. Through technologies such as GPS, radio triangulation, and localization through mobile phones, 802.11 wireless systems, and RFID tags, it becomes feasible to track individual objects at resolutions of meters in space and seconds in time–in some cases, even greater resolution is possible.

Although this increase in resolution is merely quantitative, the sheer volume and granularity of data opens up possibilities for intricate analysis and data mining of a qualitatively different nature. In this paper, we propose generic data structures and algorithms for extracting interesting information in high-resolution location histories, develop probabilistic models for location histories, and some present applications of these analytical tools.

In the geographic sciences, Hägerstrand is credited with introducing the first rigorous tools for the analysis of human migratory patterns. His *space-time prism* provided a useful visualization of movement, both of groups and individuals [3]. The space-

M.J. Egenhofer, C. Freksa, and H.J. Miller (Eds.): GIScience 2004, LNCS 3234, pp. 106–124, 2004.

time prism plotted time on the independent axis, and an interval, rather than a point, on the dependent axis to indicate the extent or uncertainty in location of a population. He used data from government censuses and manually collected logs to study migration. Geographers have built on this work considerably since its introduction, but have so far restricted attention to coarse location histories [12].

More recently, a body of work has focused on modeling location histories using "bead and necklace" representations, which capture the uncertainty of an object's location given point samples; the beads fatten as they move away from known samples at a rate proportional to bounds on object speed [4, 6, 9]. One version of this representation allows a scaled inspection of the data dependent on choice of data granularity [6]. This work has also been applied to track health-related information over many individuals [9].

Another application of location histories is in optimization of mobile phone networks, some of which allow consumers to keep track of "buddies." Mobile phones operate by switching their connectivity from tower to tower as the phone moves between cells. By predicting the movement of mobile devices, the number of location updates with each phone in service can be minimized [2, 7, 11]. Work in this area is often targeted to the task of optimizing mobile-phone operations, but at core, there are similar data structures and algorithms for handling location histories. Most often, geography is represented as a partition into cells, and movement is modeled as transition probabilities between cells [2, 7]. Others propose a more continuous approach where traditional filtering and smoothing techniques are used to estimate future state [11].

Consumer-oriented applications use similar predictive algorithms to help form personal to-do lists [10] or to give trusted friends and co-workers a better sense of one's current location [8]. A number of single-user and multiple-user applications that are made possible by using location-aware wearable computers [1]. If a wearable computer includes a GPS, clusters of logged GPS coordinates can be used to determine destinations of interest, and transitions between clusters can provide training data for developing a probabilistic model of personal movement [1].

There has also been some work in efficient updating of location histories in databases [14, 15]. Since moving objects continuously change positions, algorithms for avoiding overly frequent updates are desirable. Solutions here propose representations of movement as function of time and other parameters to predict future movements. Hence an update to the database is made only when motion parameters change.

Thus, most work to date with location histories has focused on specific applications or on particular methods for logging location, with the processing of location histories tailored to the task at hand. In this paper, we attempt to define general data structures that are independent of both application and method of acquisition. Our algorithms are likewise independent of the method or resolution at which location histories are gathered. Applicability, however, is not sacrificed at the expense of generality, and we illustrate the kind of analysis that can be performed with the proposed tools.

After defining some notation in the next section, Section 3 discusses parsing of raw location histories into *stays* and *destinations*, which we take as fundamental data structures in Section 4, for building probabilistic models of location histories. These sections strive for generality with respect to representation of location and resolution of data. Finally, in Section 5, we show the kind of analyses that can be accomplished when our basic data structures and algorithms are applied to data collected by GPS.

2 Notation

We assume the simplest possible representation of raw location data: data consist of a time-stamp and a point location. A body of raw data is, therefore, a set, $\mathsf{R} = \{r_i\}$, consisting of pairs, $r_i = (t_i, \mathbf{l}_i)$, each containing a time-stamp and a location. Without loss of generality, we assume the data is labeled such that $1 \le i \le R$ (where $R = |\mathsf{R}|$) and sorted in time order: $r_i < r_j$ if $t_i < t_j$, for any *i* and *j*.

We define locations in the most general way. They may be any identifier that identifies a single, unique, geographic point location–*n*-tuples of real values are probably the most typical, but alternate representations, such as a text label, are possible. What is critical, however, is that the locations exist in a metric space. That is, there must be a metric function, *Distance*($\mathbf{l}_i, \mathbf{l}_j$), which computes the distance between two locations, and which satisfies all of the criteria of a true mathematical metric, namely that the function is (1) positive definite: *Distance*($\mathbf{l}_i, \mathbf{l}_j$) ≥ 0 for any $\mathbf{l}_i$, $\mathbf{l}_j$; (2) *Distance*($\mathbf{l}_i, \mathbf{l}_j$) = 0, if and only if $\mathbf{l}_i$ and $\mathbf{l}_j$ represent the same location; and (3) the triangle inequality holds: *Distance*($\mathbf{l}_i, \mathbf{l}_j$) + *Distance*($\mathbf{l}_j, \mathbf{l}_k$) $\ge$ *Distance*($\mathbf{l}_i, \mathbf{l}_k$). We point out that the data structures and algorithms proposed below require only that this metric function exists–they are not dependent on how location *per se* is represented. Fig. 1a shows an example of a location history overlaid on a map.

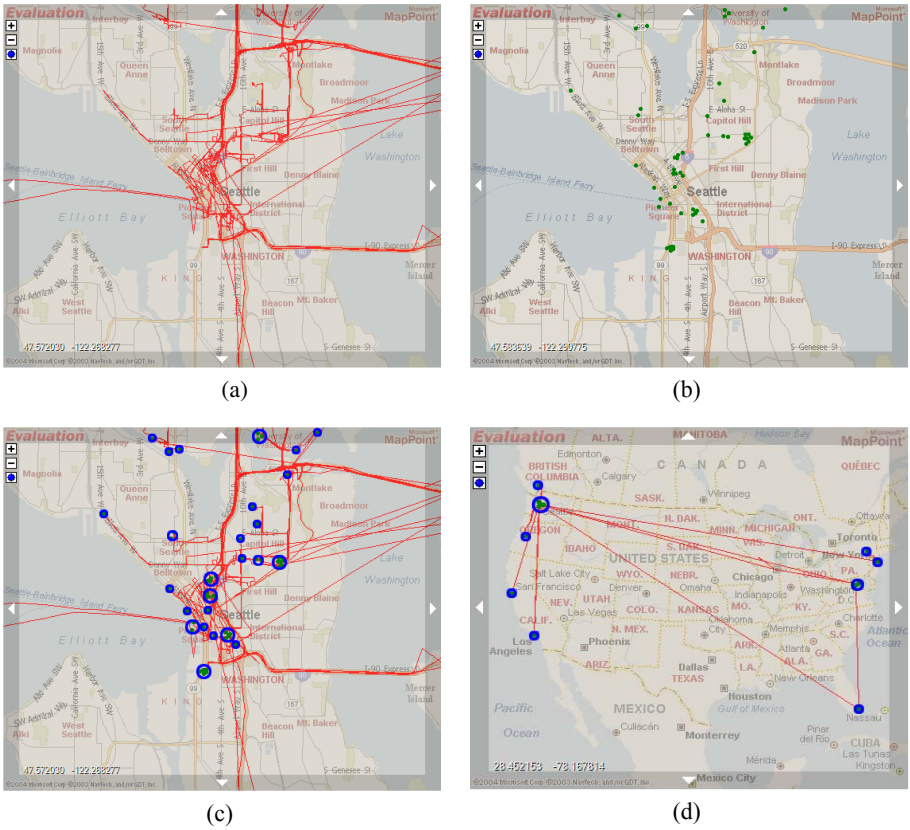
3 Parsing Location Histories

In order to analyze location histories, we parse raw location data to extract symbols that approximate intuitive semantic notions of location. In particular, we believe the following four concepts are intuitively meaningful (we will use the word *place* to mean a neighborhood around a point location):

- A ***stay*** is a single instance of an object spending some time in one place.
- A ***destination*** is any place where one or more objects have experienced a stay.
- A ***trip*** occurs between two adjacent stays (made by a single object).
- A ***path*** is a representation of the description of a set of trips between destinations.

For example, four hours spent at the office today could be a single *stay*. The office itself would be a *destination*. The particular timed trajectory going from home to office would be a *trip*. Multiple trips over the same spatial trajectory would form a *path*.

Stays and destinations are identified with places, whereas trips and paths are concerned with trajectories between places. Destinations and paths can be thought of as "timeless" generalizations of their time-dependent counterparts, respectively stays and trips. This paper focuses on what can be done with stays and destinations. A future paper will focus on trips and paths, which require their own in-depth treatment.



(a) (b)

(c) (d)

Fig. 1. Data from a few months of Subject A's location history, collected using a handheld GPS device: (a) line segments connecting adjacent points in the location history; (b) extracted stays marked as dots; (c) destinations marked as circles; and (d) stays and destinations extracted at a much coarser resolution

In the subsections to follow, we present rigorous definitions of stays and destinations, as well as algorithms for extracting them from a location history. Our approach considers a data-driven approach using variations of clustering algorithms; destinations are defined independently of *a priori* information about likely destinations. In particular, we postpone attempts to correlate stays and destinations with geographic entities defined by an existing map or GIS, and focus on destinations that appear naturally in the data themselves. We believe this is a more general approach, as it would be straightforward to associate data-driven destinations *post hoc* with existing geographical entities, if necessary.

3.1 Stays

A stay is characterized by "spending some time in one place." We would like to capture this concept rigorously while maintaining the breadth required to encompass the

semantic intuition. In particular, we note that a five-minute visit to the restroom, a half-day lounge at the beach, and a one-week vacation in Hawaii all represent different stays, even though they might all occur within the same two-week time interval. This sort of nested or overlapping structure happens throughout a given object's location history and what creates it is *scale*: stays can occur at various geographic and temporal scales. Stays at some scale might be relevant for some applications, but not for others. A hierarchical nesting of scales might be useful for yet other applications.

In any case, these examples show that the extraction of stays from a location history is dependent on two scale parameters, one each for time and spatial scale. We call these the *roaming distance* and the *stay duration*. The roaming distance, Δl^{roam}, represents the maximum distance that an object can stray from a point location to count as a stay; and a stay duration, Δt^{dur}, is the minimum duration an object must stay within roaming distance of a point to qualify as staying at that location. These parameters can be set according to the needs of the application, or the algorithm can be run multiple times with increasing scale values to create a hierarchy of stays.

A single stay is characterized by a location vector, start time, and end time: $s_i = (\mathbf{l}_i, t_i^{start}, t_i^{end})$. Our algorithm, which recovers a set of stays, $\mathsf{S} = \{s_i\}$, from the raw data, is given in Table 1. The functions $Medoid(\mathsf{R}, i, j)$ and $Diameter(\mathsf{R}, i, j)$ are computed over the set of locations represented in the set of raw data $\{r_k : r_k \in \mathsf{R}\}$, for $i \le k < j$. The *Diameter* function computes the greatest distance between any two locations in a set, and the *Medoid* identifies the location in a set that minimizes the maximum distance to every other point in the set (i.e., it is the data point nearest to the "center" of the point set). The algorithm essentially identifies contiguous sequences of raw points, which remain within the roaming distance for at least as long as the stay duration.

Table 1. Algorithm for extracting stays from raw data

Input: raw location history, $\mathsf{R} = \{r_i\}$ **Output:** a set of stays, $\mathsf{S} = \{s_i\}$

Initialize: $i \leftarrow 1$, $\mathsf{S} \leftarrow \varnothing$
while $i < R$
  $j^* \leftarrow \min j$ s.t. $r_j \ge r_i + \Delta t_{dur}$;
  if ($Diameter(\mathsf{R}, i, j^*) > \Delta l_{roam}$)
    $i \leftarrow i + 1$;
  else
  begin
    $j^* \leftarrow \max j$ s.t. $Diameter(\mathsf{R}, i, j) \le \Delta l_{roam}$;
    $\mathsf{S} \leftarrow \mathsf{S} \cup (Medoid(\mathsf{R}, i, j^*), t_i, t_{j^*})$;
    $i \leftarrow j^* + 1$;
  end
end

In the worst case, the algorithm is an $O(n^2)$ algorithm for n data points, since medoid and diameter computations require distance computations between all pairs in a

stay cluster. In practice, however, clusters over which these computations must take place are far smaller than n, and performance is effectively $O(n)$. Many of the problems of clustering unordered points (e.g., as encountered in [15]) are avoided because of the temporally ordered nature of the data.

Examples of stays extracted in this manner are shown in Fig. 1. In Fig. 1b, stays were extracted with a roaming distance of 50 m and stay duration of 10 minutes, whereas Fig. 1d shows the results for a roaming distance of 20 km and a stay duration of 24 hours.

3.2 Destinations

A destination is any place where one or more tracked objects have experienced a stay. Destinations are dependent on geographic scale, but not on temporal scale (i.e., beyond the temporal scales used to identify stays). The scale determines how close two point locations can be and still be considered part of the same destination. As with stays, the scale of a destination is dependent on the intended usage, and so it is a parameter that must be set explicitly. For example, a scale representing ~3 m might be appropriate for extracting destinations corresponding to offices in a building, but a scale of ~100 m would be necessary for identifying whole buildings as destinations.

Given a set of locations, $\mathsf{L} = \{\mathbf{l}_i\}$, our aim is to extract all the destinations $\mathsf{D} = \{d_j\}$ at a particular geographic scale Δl^{dest}. Each destination will be represented by a location and the scale used: $d_j = (\mathbf{l}_j, \Delta l_j^{dest})$.

Determining destinations from a set of location vectors is a clustering task. There are many options for clustering points, ranging from *k*-means clustering to hierarchical clustering techniques. We choose to use a type of agglomerative clustering, because it allows us to specify the spatial scale of the clusters, rather than the number of clusters or the number of points contributing to a cluster, neither of which we know *a priori*.

Let a cluster be characterized by a set of point locations: $\mathbf{c} = \{\mathbf{l}\}$. The clusters are initialized by assigning each input point location to a cluster, and hence there are as many clusters as location points at the beginning. During each iteration of the algorithm, the two closest clusters are identified. If the cluster resulting from merging the two clusters would be within the specified scale, Δl^{dest}, they are merged. Otherwise, the algorithm stops and outputs all remaining clusters as destinations. This is an $O(m^2)$ algorithm for m stays, because of the need to compute distances between all pairs of stays.

Table 2 shows pseudocode for this algorithm. The function *FindClosestPair* finds the closest two clusters from the cluster set, *Radius* computes the combined radius of the two clusters assuming that they are merged, and *Merge* combines two clusters into one. The *Radius* of a set of locations is the distance from the set's medoid to the location within the set, which maximizes the distance.

It will be useful for later sections to define a function $d(\mathbf{l})$, which returns the nearest destination to location $\mathbf{l}$. This may be further extended to $d(\mathbf{l}, \Delta l^{dest})$, which returns a null value if the location is not within Δl^{dest} of any known destination.

Finally, destinations can be further computed hierarchically across scales, by allowing the medoids of each cluster created at one scale, Δl_j^{dest}, to be used as input locations to compute destinations at a greater scale, Δl_{j+1}^{dest}.

Table 2. Algorithm for computing destinations

Input: a set of point locations, $\mathsf{L} = \{\mathbf{l}_i\}$ **Output:** a set of destinations, $\mathsf{D} = \{d_j\}$

Initialize: $\mathbf{c}_i \leftarrow \mathbf{l}_i$, for $1 \le i \le L$, and $\mathsf{C} = \{\mathbf{c}_i\}$

loop

$(\mathbf{c}_i, \mathbf{c}_j) \leftarrow FindClosestPair(\mathsf{C})$;

if $Radius(\mathbf{c}_i, \mathbf{c}_j) \le \Delta l^{dest}$

$\mathbf{c}_i \leftarrow Merge(\mathbf{c}_i, \mathbf{c}_j)$;

$\mathsf{C} \leftarrow \mathsf{C} - \mathbf{c}_j$;

else

exit

end

foreach $\mathbf{c}_i \in \mathsf{C}$, create destination $d_i = (Medoid(\mathbf{c}_i), \Delta l^{dest})$;

Fig. 1c and Fig. 1d show destinations after clustering stays with this algorithm. The circles indicate both the location and radius of each destination. The destinations in Fig. 1c were clustered with a scale setting of $\Delta l^{dest} = 250\,\text{m}$, in Fig. 1d, $\Delta l^{dest} = 25\,\text{km}$.

Armed with data structures for stays and destinations, we can proceed to construct probabilistic models of location histories.

4 Modeling Location Histories

The goal of our location-history models is to condense, understand, and predict the movements of an object over a period of time. We investigate two probabilistic models for location histories, one with and one without first-order Markovian conditioning of the current location on subsequent location. Our experiments in Section 5 show that both have value, depending on the kind of questions that are asked of the model. The next two subsections define some notation and establish assumptions made by our model. The subsections after that describe our model, together with algorithms for training, estimation, and prediction.

4.1 Notation

The *destination set*, $\mathsf{D} = \{d_i\}$, is the set of all destinations (as determined in Section 3.2), where $1 \le i \le n$ and $n = |\mathsf{D}|$ denotes the total number of destinations.

We need to distinguish between three different units of time. A *time instant*, t, represents an instantaneous moment in time; if time is thought of as a real-valued entity of one dimension, a time instant represents a single point on the real number line. Next, for a given interval unit of time, δt (e.g., an hour), a *time interval*, $\mathbf{t}$, represents a half-open unit interval on the real number line, aligned to the standard calendar and clock. For example, for δt equal to an hour, $\mathbf{t}$ might be a time interval starting at 18:00UTC today and going up to, but not including 19:00UTC. Finally, a *recurring time interval*, τ, is the set of all time intervals that represents a regularly recurring interval of time. Continuing the example, τ might be the set of all times occurring between 18:00 and 19:00, regardless of date. A set of non-intersecting, recurring time intervals that covers all times will be denoted $\mathsf{T} = \{\tau_k\}$, for $1 \le k \le m$, with $m = |\mathsf{T}|$ indicating the number of recurring time intervals required to cover all of time.

The granularity, δt, of a recurring time interval and the period with which it recurs is something that must be decided for a particular model *a priori*. Thus, we might decide for a particular model that δt represents an hour and recurring time intervals cycle each day (in which case, $m = 24$) or that each hour of the week should be different recurring intervals ($m = 168$). If so, then $\mathbf{t}_p \subset \tau_k$ if $\mathbf{t}_p$ represents the particular hour between 18:00 and 19:00 on September 30, 2003, and τ_k represents the recurring time interval 18:00-19:00.

Finally, we define a function, $\tau(t)$, that extracts the recurring time interval that contains a time instance: $\tau(t_p) = \tau_k$, if and only if $t_p \in \tau_k$. With minor abuse of notation, we also let $\tau(\mathbf{t}_p) = \tau_k$, if and only if $\mathbf{t}_p \subset \tau_k$.

4.2 Model Assumptions

Both of the location-history models presented in this paper are based on the following assumptions:

- At the beginning of a given time interval, an object is at exactly one destination.
- During any given time interval, an object makes exactly one transition between destinations. A transition may occur from a destination to itself (a *self-transition*).

These are not ideal assumptions, by any means. For example, the possibility of multiple transitions occurring within a time interval is not explicitly modeled by our current algorithms. We chose these assumptions, however, to strike a compromise between allowing arbitrary transitions and expressive power of the model–a compromise that would not require unreasonable amounts of data to train.

Based on the above assumptions we define the following probability tables, in a manner analogous to Hidden Markov Models [13] . The critical difference from the standard HMM formulation is that we incorporate time-dependence into the model, where transition probabilities are conditioned on recurring time intervals, rather than being fixed regardless of the time. This was a deliberate design decision that allows us to capture cyclical behavior that is, for example, dependent on time-of-day. With this

modification, we can model the fact that at 8am, it is far more likely that we travel from home to office than at 4am.

The probability of the object starting time interval τ_k at destination d_i is represented by a matrix of probabilities, $\Pi = \{\pi(d_i, \tau_k)\}$ where

$$\pi(d_i, \tau_k) = \Pr(d = d_i \text{ at the start of } \tau_k) \tag{1}$$

and

$$\pi(d_i, t_p) = \pi(d_i, \tau_k) \text{, for } t_p \in \tau_k \text{.} \tag{2}$$

such that, $\sum_{i=1}^{n} \pi(d_i, \tau_k) = 1$.

Next, the probability that the object makes a transition from destination d_i to d_j during interval τ_k is given by a table, $A = \{a(d_i, d_j, \tau_k)\}$,

$$a(d_i, d_j, \tau_k) = \Pr(d = d_j) \text{ where at the start of } \tau_{k+1} \mid d = d_i \text{ at the start of } \tau_k \tag{3}$$

such that, $\sum_{i=1}^{n} a(d_i, d_j, \tau_k) = 1$. Also, $a(d_i, d_j, t_p) = a(d_i, d_j, \tau_k)$ where $t_p = \tau_k$.

To complete the HMM analogy, we include the observation probability. $B = \{b(d_i, d_j)\}$ represents the probability of observing that the object is at destination d_j, given that the object is actually at destination d_i, with

$$b(d_i, d_j) = \Pr(d^{observed} = d_j \,\big|\, d^{actual} = d_i) \tag{4}$$

Together as $\lambda = (\Pi, A, B)$, these tables represent a probabilistic generative model of location for the object modeled. Once the parameters are learned, this model can be used to solve problems such as finding the most likely destination occupied at a particular time, determining the relative likelihood of a location history sequence, or stochastically generating a location history sequence.

4.3 Training the Model

We now present algorithms for learning model parameters λ from training data. Our training data consist of a set of stays, $S = \{s_i\}$, as extracted from the raw data in Section 3. Recall that each stay, *s*, is a 3-tuple containing a start time, an end time, and a destination: $s_i = (d_i, t_i^{start}, t_i^{end})$.

4.3.1 Computing Π

To compute Π, we simply count the number of occurrences in the training data where the object started a recurring time interval in a particular destination and normalize it over all training data for that recurring interval. Table 3 shows pseudocode.

Table 3. Algorithm for computing Π, the prior probabilities of being at a destination at a given recurring time interval

Input: set of stays, $\mathsf{S} = \{s_i\}$ **Output:** probability table, $\Pi = \{\pi(d_i, \tau_k)\}$

Initialize: $count(d_i, \tau_k) \leftarrow 0$, for $1 \le i \le n$ and $1 \le k \le m$
// count
for each $s_i \in \mathsf{S}$
  if $\tau(t_i^{start}) = \tau(t_i^{end})$ **and** $t_i^{end} - t_i^{start} < \delta t$
    continue
  else
    for $t \leftarrow Ceiling(t_i^{start}) : t_i^{end} : \delta t$
      $count(d^{(i)}, \tau(t)) \leftarrow count(d^{(i)}, \tau(t)) + 1$;
    end
end
// normalize
for each i, k in $1 \le i \le n$ and $1 \le k \le m$
  $\pi(d_i, \tau_k) \leftarrow count(d_i, \tau_k) / \sum_j count(d_i, \tau_k)$;
end

An example of the result of this algorithm is given in Section 5.4.

4.3.2 Computing *A*

To compute *A*, we count the number of occurrences in the training data where the object makes a transition from a particular destination to another destination (or itself) during a recurring time interval and normalize it over all the training data for that recurring interval. This algorithm is shown in Table 4.

4.4 Location History Analysis

We now use the location history model, λ, to estimate the relative likelihood of a new location history, $\tilde{\mathsf{H}} = \{d(\mathbf{t}_u)\}$, defined over $u \in [start, finish]$. We propose two different processes for doing this.

4.4.1 Non-Markovian Solution

We determine the probability of the location history by computing the joint probability $\pi(d(\mathbf{t}_u), \mathbf{t}_u)$ and $b(d(\mathbf{t}_u), d(\mathbf{t}_u))$ from time $\mathbf{t}_{start}$ to time $\mathbf{t}_{finish}$, and marginalizing (summing) the joint probabilities over all possible location history sequences. This can be represented by the following equation:

$$\Pr_\pi(\tilde{\mathsf{H}} \mid \lambda) = \sum_{\mathsf{H} \in \{d_{h_{start}}, \ldots d_{h_{finish}}\}} \prod_{u=start}^{finish} \pi(d_{h_u}, \mathbf{t}_u) b(d(\mathbf{t}_u), d_{h_u}) \tag{5}$$

Table 4. Algorithm for computing *A*, the probability table showing the likelihood of transition between destinations at a given recurring time interval

Input: $\mathsf{S} = \{s_i\}$ **Output:** probability table, $A = \{a(d_i, d_j, \tau_k)\}$

Initialize: $count(d_i, d_j, \tau_k) \leftarrow 0$, for $1 \le i, j \le n$ and $1 \le k \le m$

for each $s_i \in \mathsf{S}$

// count self-transitions

if $\tau(t_i^{start}) = \tau(t_i^{end})$ **and** $t_i^{end} - t_i^{start} < \delta t$

continue

else

for $t \leftarrow Ceiling(t_i^{start}) : t_i^{end} : \delta t$

$count(d^{(i)}, d^{(i)}, \tau(t)) \leftarrow count(d^{(i)}, d^{(i)}, \tau(t)) + 1$;

end

end

// count other transitions

if $i \neq |S|$ **and** $\tau(t_i^{end}) \neq \tau(t_{i+1}^{start})$

$t = count(t_{i+1}^{start})$;

$count(d^{(i)}, d^{(i)}, \tau(t)) \leftarrow count(d^{(i)}, d^{(i)}, \tau(t)) + 1$;

end

end

// normalize

for each i, j, k in $1 \le i, j \le n$ and $1 \le k \le m$

$a(d_i, d_j, \tau_k) \leftarrow count / \sum_j count(d_i, d_j, \tau_k)$;

end

If observations are accurate, this reduces to

$$\Pr_{\pi}(\tilde{\mathsf{H}} \mid \lambda) = \prod_{u=start}^{finish} \pi(d(\mathbf{t}_u), \mathbf{t}_u) \tag{6}$$

This approach assumes that there is no conditional dependency of state between time intervals.

4.4.2 Markovian Solution

Another method of determining the probability of location history is by computing the joint probability of the observation sequence and the state sequence and marginalizing over all possible location history sequences:

$$\Pr_{A}(\tilde{\mathsf{H}} \mid \lambda) = \sum_{\mathsf{H} \in \{d_{h_{start}}, \ldots, d_{h_{finish}}\}} \pi(d_{h_s}, \mathbf{t}_s).b(d(\mathbf{t}_u), d_{h_s}) \prod_{u=start}^{finish-1} a(d_{h_{u+1}}, d_{h_u}, \mathbf{t}_u).b(d(\mathbf{t}_{u+1}), d_{h_{u+1}}) \tag{7}$$

This reduces to

$$\Pr_A(\tilde{\mathsf{H}} \mid \lambda) = \pi(d(\mathbf{t}_s), \mathbf{t}_s) \prod_{u=start}^{finish-1} a(d(\mathbf{t}_{u+1}), d(\mathbf{t}_u), \mathbf{t}_u) \tag{8}$$

if observations are accurate. This approach uses the transition probabilities A, and assumes that the object's destination at a time interval is conditionally dependent on the destination at the previous time interval. This is equivalent to the standard "forward algorithm" used to evaluate the probability of a sequence of observations in an HMM [13], but with the modification for time-dependent transition probabilities.

Whichever method is used, the output is a true probability in the strict sense, but only given the assumptions of the respective estimates. In reality, probabilities of events over time intervals are ill-defined–for one thing, the probability of a particular event approaches zero as the event is sampled over shorter sub-intervals. Thus, these values are most meaningful when interpreted as relative likelihoods between events observed using the same interval unit. For example, we can compare the relative likelihoods of two location histories of a week's length with δt equal to one hour and judge their relative rarity. We could also set thresholds for a history dependent on the length of the history, to determine whether an input history appears normal or abnormal. Finally, given multiple models, λ_i, we can determine which model best explains a given history by computing $\arg\max_i \Pr(\tilde{\mathsf{H}} \mid \lambda_i)$.

4.5 Stochastic Generation

Using the model parameters, λ, we can stochastically generate a location history $\mathsf{H}_{gen} = \{d(\mathbf{t}_u)\}$ for $u \in [start, finish]$, where $d(\mathbf{t}_u)$ is the destination occupied at time interval $\mathbf{t}_u$. We outline two methods for generating location histories.

In the first, we use only the Π parameters, and randomly sample from the set of destinations for each time interval without conditional dependence between time intervals. Destinations are chosen such that

$$\Pr(d(\mathbf{t}_u) = d_i) \propto \sum_j \pi(d_j, \mathbf{t}_u) b(d_j, d_i) \tag{9}$$

In practice, this can be done by a basic Monte Carlo "coin-tossing" process to generate an "actual" destination, d_j, using π, which is then followed by another coin toss to determine the observed destination, d_i, using B. This simplifies to a single coin toss per time interval in the case where observations of destinations are noiseless.

In the second technique, we utilize the full Markov model and perform a similar Monte Carlo sampling using the transition probabilities, A, in all but the first time interval. Thus,

$$\Pr(d(\mathbf{t}^{start}) = d_i) \propto \sum_j \pi(d_j, \mathbf{t}^{start}) b(d_j, d_i) \tag{10}$$

as before for $\mathbf{t} = \mathbf{t}_{start}$, but

$$\Pr(d(\mathbf{t}_u) = d_i \mid d(\mathbf{t}_{u-1}) = d_k) \propto \sum_j A(d_k, d_j, \mathbf{t}^{start}) b(d_j, d_i) \tag{11}$$

for the remaining time intervals. Again, this is implemented in practice as a simple series of Monte Carlo coin tosses.

5 Experimental Results

We conducted experiments with the raw location histories of two subjects, who together collected over two years of data using handheld GPS devices carried on their person. We have 346 days of data for Subject A, and 386 days for Subject B.

5.1 Stays

We extracted stays from the raw points using the algorithm described in Table 1. Generating stays at five different temporal scale parameters shows the effect of time scale on number of stays. Time duration, Δt^{dur}, was set to 10, 20, 40, 80, and 160 minutes. In all cases, the roaming distance was set to $\Delta l^{roam} = 30$ m to account for GPS noise. Fig. 2 summarizes these results.

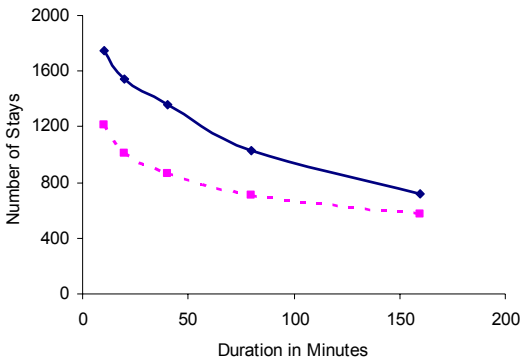


Fig. 2. Plots of the number of stays versus the stay duration parameter used to extract stays, for two subjects

As would be expected, fewer stays are generated if the time threshold for considering a pause a stay is lengthened. While the two subjects show differences in the absolute number of stays, there is an approximately exponential fall-off in the number as stay duration is increased. This confirms the intuition that stays might conform to a power law, where short stays are far more likely than long stays–one is much more likely to make short trips to the bathroom than to take week-long vacations in Hawaii.

5.2 Destinations

Given stays, we then cluster them into unique destinations, using the algorithm described in Table 2. In computing destinations, geographic scale is a key factor. We

confirm that scale affects extraction of destinations, with destinations generated at geographic scales of 250, 2,500, and 25,000 m. For both the experiments, stays at $\Delta t^{dur} = 10$ minutes and $\Delta l^{roam} = 30$ m were used. Table 2 summarizes these results. Fig. 1c shows destinations extracted when $\Delta l^{scale} = 250$ m; and Fig. 1d, when $\Delta l^{scale} = 25$ km.

Table 5. Destinations generated for subjects A and B at different geographic scales

Subject	Δl_{scale} (meters)	# Destinations
A	250	234
	2,500	78
	25,000	20
B	250	179
	2,500	72
	25,000	26

5.3 Some Simple Analysis

The power of extracting stays and destinations is illustrated in the following examples, where we compute a variety of statistics about the subject's lives.

Table 6. Top five destinations by number of stays at each, for two subjects

Subject	Destination	# Stays at Destinations	Total Time at Destinations (hours per year)
A	work, primary	443	3,365
	home	388	2,190
	gym	70	135
	mall	40	39
	friend's house	38	131

Subject	Destination	# Stays at Destinations	Total Time at Destinations (hours per year)
B	home	411	3,924
	work, primary	180	1,117
	work, secondary	76	420
	work, other	20	15
	murphy's corner	19	27

In Table 6, we show a sample of the kind of information that can be easily extracted by displaying each of our two subject's top five most frequently visited destinations. In this case, the destination names were provided by the subjects, who viewed the destinations on a map, but even this process could be automated by using a combination of GIS-lookup to map destinations to established place names, and heuristics to learn person-specific destinations (e.g., time of day and amount of time spent at a location will give strong indicators of home and work). One application of this information is for cell-phone location privacy. The location-based services (LBS) industry, for example, is marketing such services as location-sensitive coupons, if

users are willing to allow merchants to know their current location. Consumers may be reluctant to give away this information for privacy reasons, if it discloses sensitive destinations, such as when exactly they are at home. By automatically determining where a user's home is, however, the carrier can offer "location dithering" services that would either limit or intentionally coarsen estimates of location when the user is in the neighborhood of a sensitive destination.

We can also analyze subtle patterns of behavior through simple manipulation of the data. For example, in Fig. 3, we show the average number of hours spent at Subject A's primary office location, computed by histogramming stays by day of the week and dividing them by the number of total days of raw data. There is a clear trend where the number of hours spent at work peaks on Tuesdays and gradually trails off toward Friday. Subject A confirms, "I always felt most productive on Tuesdays."

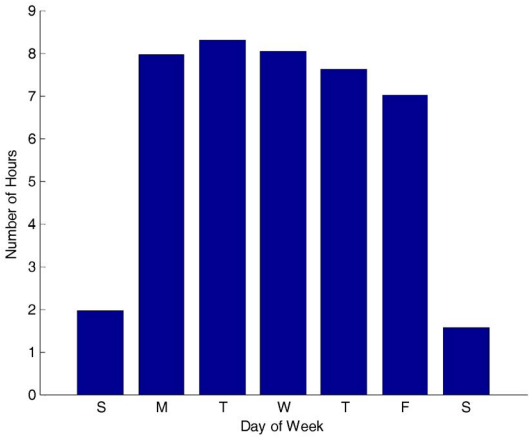


Fig. 3. Histogram of average number of hours spent by Subject A at his primary workplace

In Fig. 4, we show another easily generated plot of the average number of destinations for Subject A, broken down by month. We can instantly see that Subject A has a reasonably steady routine that involves little daily travel, but that there is greater variance in August and December, probably due to vacation activities.

5.4 Experiments with Location History Modeling

We used the low-level processed information, stays and destinations generated from the raw data of user, to train our location-history models. We chose location histories recurring at a period of one week, with recurring time interval, δt , of one hour (m =168 hours per week), for our models. Fig. 5 shows "typical" and "atypical" weeks for Subject A, as picked by the subject. The x-axis plots the day of the week, and the y-axis the index of the destination (an arbitrary number chosen per destination during the clustering process). Indeed, the vast majority of the weeks in this dataset reveal a pattern of activity that qualitatively looks like Fig. 5a, where most of the time is spent either at home or at work.

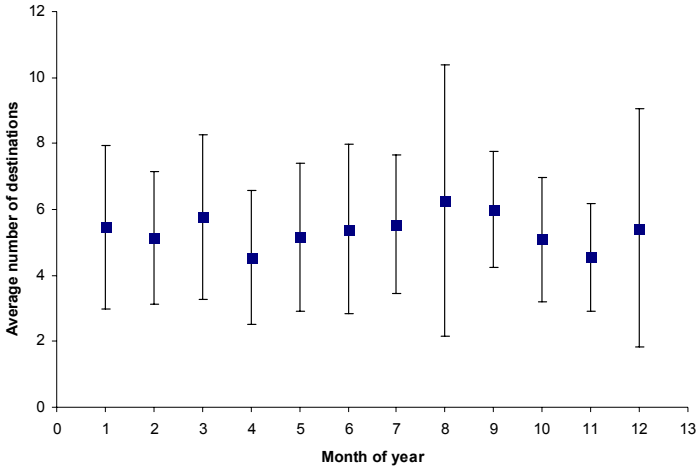
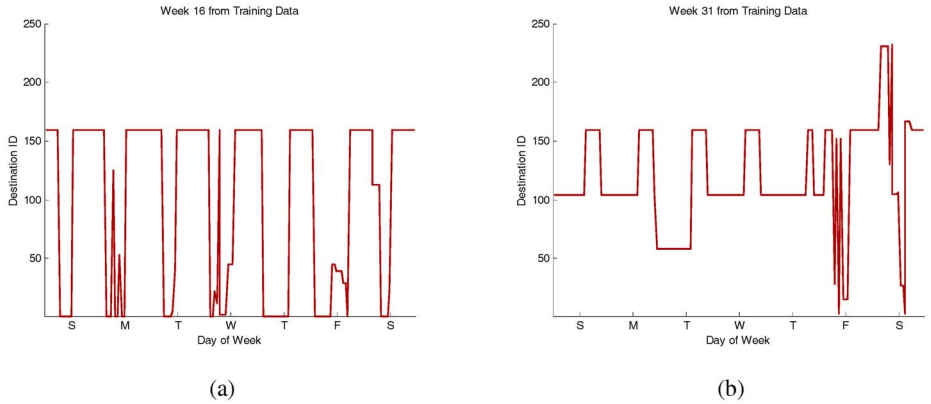


Fig. 4. Average number of destinations visited by Subject A, by month of year



(a) (b)

Fig. 5. (a) Typical and (b) atypical weeks for Subject A. The index of the destination is plotted against time. (The ordering of the destination indices is entirely arbitrary)

5.5 Evaluation of Location Histories

In this section we present results of the experiments conducted to evaluate the likelihood of a week's location history. This sort of analysis could distinguish between typical and atypical patterns of behavior, and might be used, for example, to provide more sophisticated electronic calendars, which take into account a person's recent location history to predict future location and work cycles.

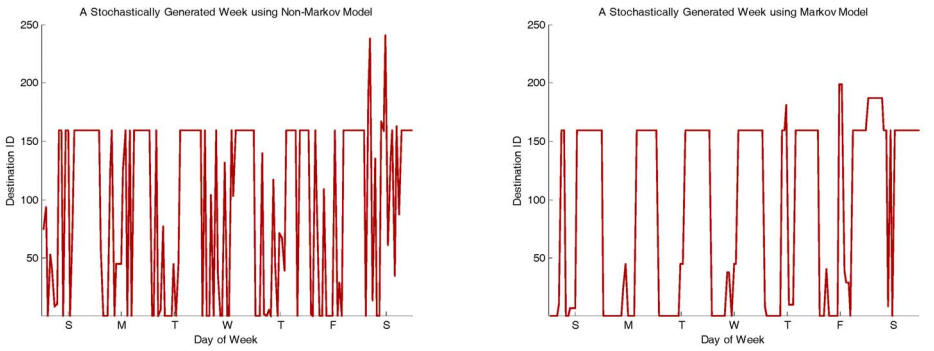
We computed the model parameters $\lambda = (\Pi, A, B)$ using the algorithms described in Tables 3 and 4. We did not consider stays that were less than an hour while calculating the probability table Π.

As a simple verification of the evaluation process, we computed likelihoods of week-long location histories, given trained models (that did not include the week

evaluated). The goal is to find an evaluation process that gives us higher likelihoods for typical weeks and lower likelihoods for atypical weeks. As indicated in Section 4.4, we can use either a Markovian or non-Markovian approach for estimation.

Fig. 6 shows the results of computing the log likelihood of each of 52 weeks in Subject A's location history. The circles indicate results using the non-Markovian evaluation, and crosses, for Markovian evaluation. Although the Markovian evaluation shows lower probabilities overall (due to inclusion of transition probabilities during evaluation), relative estimates are similar between the two instances of data, as expected. Unexpected, however, are the results for week 13 (indicated by arrows in the graph), which was an atypical week according to the subject. Whereas the non-Markovian process shows this to be a highly atypical week with low likelihood, the Markovian evaluation, somewhat counterintuitively, shows an unusually high likelihood. Investigation of the underlying data shows that Subject *A* engaged in an activity that occurred with almost identical patterns of infrequent movement, exactly once in the training data, and once during week 13 of the test data. Because the Markovian evaluation process incorporates transition matrices between destinations, near-match sequences between training and test data for atypical weeks will come out to be far more likely than typical weeks, which distribute transition probabilities more diffusely across a greater number of destinations and times.

Although this case could be handled by the Markovian model by training on larger sets of data, or by clustering Markovian models themselves with the frequency of their occurrence, our conclusion in this case is that for the purposes of identifying typical patterns of activity, the non-Markovian model is sufficient. Indeed, a threshold of -350 on the log likelihood for this data using non-Markovian analysis results in a perfect identification of atypical weeks, that is in synch with notes by the subject.



(a) (b)

Fig. 6. Plots of synthesized weeks, using a model trained on Subject A's data: (a) using the non-Markovian model and (b) with Markovian transitions

It should be noted, however, that even the Markovian generation does not result in histories that match the statistics of true data–careful comparison of Fig. 6b and Fig. 5a reveals that in actual data, the subject spends longer amounts of time at destination 157 (home). This is due to a known flaw of standard Markov chains, in that the length of time spent at a particular destination is necessarily exponential in distribution

(whereas real data may contain non-exponential distributions). The problem is actually mitigated in our algorithm, because our transition probabilities are time-dependent, but the effects of considering only first-order effects are still noticeable.

6 Conclusions

This paper proposed rigorous definitions for location histories, as well as algorithms for extracting stays and destinations from location histories in a pure, data-driven manner. Both Markovian and non-Markovian probabilistic models were also developed for modeling a location history. Experiments show that these techniques are effective at extracting useful information about detailed location histories, and that they can be applied to a variety of applications. We find that a non-Markovian approach is better suited for evaluating likelihoods of a location history, while the Markovian approach is superior for purposes of stochastically generating a history.

We believe analysis of location histories to be a rich area of research, with many technical approaches and interesting applications. In future work, we expect to extend the analysis to trips and paths (what happens between stays and destinations), as well as to develop more accurate location-history models.

Acknowledgments

We would like to thank the following people for early brainstorming on this project: Ross Cutler, John Douceur, Nuria Oliver, Eric Ringger, Dan Robbins, Andreas Soupliotis, and Matt Uyttendaele. Thanks also to Chris Meek for discussions on probabilistic modeling.

References

1. Ashbrook, D., Starner, T. Learning significant locations and predciting user movement with GPS. In: Billinghurst, M., eds. *6th International Symposium on Wearable Computers (ISWC)*, 2002, pp. 101-108, Seattle, WA, IEEE Computer Society.
2. Bhattacharya, A., Das, S.K. LeZi-update: an information-theoretic approach to track mobile users in PCS networks. In: Imielinski, T., and Steenstrup, M., eds. *5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1999, pp. 1-12, Seattle, WA.
3. Hägerstrand, T. What about people in regional science? Papers of the Regional Science Association, 1970, vol. 24, pp. 7-21.
4. Hariharan, R. Modeling Intersections of Geospatial Lifelines. M.S. thesis, Department of Spatial Information Science and Engineering, University of Maine, 2001.
5. Hershberger, J. Smooth kinetic maintenance of clusters. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, 2003, pp. 48-57, San Diego, CA.
6. Hornsby, K., Egenhofer, M.J. Modeling Moving Objects over Multiple Granularities. Annals of Mathematics and Artificial Intelligence, 2002, vol. 36 (1-2), pp. 177-194.
7. Lei, Z., Rose, C. Wireless subscriber mobility management using adaptive individual location areas for pcs systems. In *IEEE International Conference on Communications (ICC)*, 1998, vol. 3, pp. 1390-1394, Atlanta, GA.

8. Mantoro, T., Johnson, C. Location history in low-cost context awareness environment. In: Johnson, C., Montague, P., and Steketee, C., eds. *Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers*, 2003, vol. 21, pp. 153-158, Adelaide, Australia.
9. Mark, D., Egenhofer, M.J., Bian, L., Hornsby, K., Rogerson, P., Vena, J. Spatio-temporal GIS analysis for environmental health using geospatial lifelines [abstract]. In: Flahault, A., Toubiana, L., and Valleron, A., eds. *2nd International Workshop on Geography and Medicine*, 1999, GEOMED'99, pp. 52, Paris, France.
10. Marmasse, N., Schmandt, C. Location-aware information delivery with *ComMotion*. In: Thomas, P.J., and Gellersen, H., eds. *Handheld and Ubiquitous Computing, Second International Symposium (HUC)*, 2000, pp. 157-171, Bristol, UK, Springer.
11. Pathirana, P.N., Savkin, A.V., and Jha, S.K. Mobility modeling and trajectory prediction for cellular networks with mobile base stations. In 4th *International Symposium on Mobile Ad Hoc Networking and Computing* (*MobiHOC*), 2003, pp. 213-221, Annapolis, MD.
12. Pred, A. Space and time in geography: essays dedicated to Torsten Hägerstrand. Lund Studies in Geography, 1981.
13. Rabiner, L.R. A tutorial on Hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, 1989, vol. 77(2), pp. 257-285.
14. Revesz, P., Chen, R., Kanjamala, P., Li, Y., Liu, Y., Wang, Y. The MLPQ/GIS constraint database system. In: Chen, W., Naughton, J.F., Bernstein, P.A., eds. *SIGMOD 2000*, pp. 601, Dallas, TX.
15. Wolfson, O., Sistla, P.A., Chamberlain, S., and Yesha, Y. Updating and Querying databases that track mobile units. Distributed and Parallel Databases, 1999, vol. 7(3), pp. 257-287.