Mining User Position Log for Construction of Personalized Activity Map

Hui Fang[1], Wen-Jing Hsu[1], and Larry Rudolph[2]

[1] Singapore-MIT Alliance, Nanyang Technological University
N4-02b-40, 65 Nanyang Drive, Singapore 637460
fang0025@ntu.edu.sg, hsu@pmail.ntu.edu.sg
[2] VMware Inc.
5 Cambridge Center, Cambridge, MA, USA
rudolph@vmware.com

Abstract. Consider a scenario in which a smart phone automatically saves the user's positional records for personalized location-based applications. The smart phone will infer patterns of user activities from the historical records and predict user's future movements. In this paper, we present algorithms for mining the evolving positional logs in order to identify places of significance to user and representative paths connecting these places, based on which a personalized user activity map is constructed. In addition, the map is designed to contain information of speed and transition probabilities, which are used in predicting the user's future movements. Our experiments show that the user activity map well matches the actual traces and works effectively in predicting user's movements.

1 Introduction

Personal positioning refers to the inference of a mobile user's current position based on user's historical positional records and occasional measurements. It has inspired many location-aware applications for personal use with the development of wearable computers that are equipped with position sensors. See, e.g., [9, 6, 14]. The raw data collected from the position sensors, or the user's log, provides an important source for mining useful patterns for personal positioning.

The existing data mining techniques for personal positioning can be classified into two categories: (i) location prediction based on location-to-location transition probabilities [10, 15, 11], and (ii) position tracking based on Bayesian filters [5, 2]. Techniques in the first category aim at inferring user's presence at certain locations (called the *significant places*) and the transitions among these places; but these techniques are generally not designed for the purpose of inferring accurate positions. Techniques in the second category involve online estimation of the user's exact positions, and as such, they usually require well-defined dynamics model of the target. A well-designed model of user activities thus constitute the basis for accurate prediction of user's nondeterministic movements.

The challenges for personal positioning lies in the construction and maintenance of this personalized map, and the modeling of the user movements. In a mobile and pervasive context, the device may be allowed to access some positioning sensors, but the large-scale map resources such as central geographic database may be unavailable. A smart device can assist the user's movement only with geometrically accurate map of his environment. In other words, a smart device needs to figure out the map of user's activities on its own in many cases. Unfortunately, the mobile user's positional records can increase quickly with regular uses, which poses difficulty in both storage and information retrieval. Therefore, the key problem here is to build an internal representation of user's environment from user's own movements. This map should be different from generic maps in two aspects: (i) it evolves from user's movement log; (ii) it highlights the geographical features (locations and paths) of significance specific to the user.

These challenges motivate the paper. We propose a new data mining approach to generate a map of user activities for use in personal positioning. Our construction of the user activity map includes three parts: identifying significant places, identifying representative paths, and building the speed and transition model. The experiments show that the position tracking based on user activity map provides sufficient accuracy.

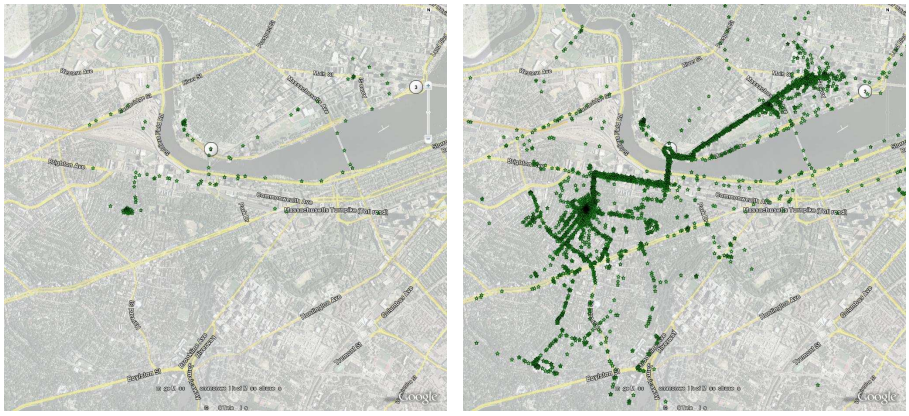
The paper is organized as follows. Section 2 presents the problem statement. Section 3 shows the construction of significant places. Section 4 shows the construction of representative paths. Section 5 shows the construction of user activity map. Section 6 shows the experimental results. Finally Section 7 concludes the paper.

2 Patterns in User Activities

As "history often repeats itself", with more and more positional samples of regular use over a period of time, a repeated path will be accumulated into the user's trace log. The user's trace log is a sequence of position measurements ordered by time. Fig. 1(a) shows the GPS records gathered from a user over a week. Fig. 1(b) shows the GPS data of the same user over three months. The figures suggest that some repetitive patterns exist in the user's trace log.

The required map of user activities should be adaptive to the log updates. This map is an abstraction of user's history of movements. Thus, if the user repeats a routine activity, the size of the map should not increase drastically. On the other hand, the map must reflect the recent significant change of information about the user's environment. For example, if a user recently starts to use a detour around a street-block, the map should be revised accordingly.

The required map of user activities should identify the locations and paths that are significant to the user. Unfortunately, *location* is a rather imprecise notion, which often means different things to different people. People often refer to a location or place without precise position. The size of a location ranges widely. Cities, streets, buildings, or even rooms can be called locations. GPS



(a) one week. (b) three months.

Fig. 1. GPS samples of a user collected in Boston-Cambridge (MA) area. Each GPS sample is marked by a star. The background is shown by Google Earth.

coordinates are seldom made references of by ordinary people. In addition, people establish relationship between locations to learn the environment. For example, one can tell a direction by saying "five minutes walk from the shop, you will get to the post-office". The shop, the post-office, and the path between them, thus form the significant components of a map.

3 Identifying Significant Places

To extract patterns, one crucial step is to identify "significant places". A mobile user might stay at one location for some time, or he may visit this location frequently (while the device may keep recording). Thus, a location is identified to be significant if user's dwelling time at this location is sufficiently long, or if the density of the location is sufficiently high. A location is defined to be a tuple $(x, y; r)$ where (x, y) represents the coordinates of a position and r denotes the radius of the associated circle centered at (x, y). Further, the definitions of location density and dwelling time are given below.

Definition 1 (location density). *The* density *of a location is the number of sample points within the associated circular region.*

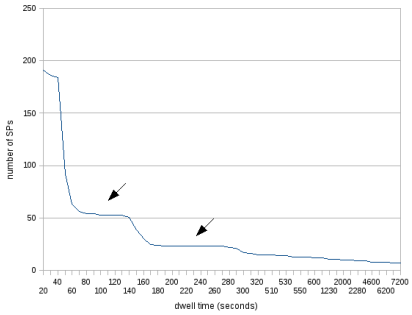
Definition 2 (dwelling time). *The* dwelling time *at each location is approximated by averaging the time staying at this position for each trace.*

Clustering is one of the main techniques used in identifying the significant places by density. After clustering, the positional points that are geographically or temporally nearby within a specified distance will be identified as within one cluster, i.e., they can be represented by a cluster head whose density is the

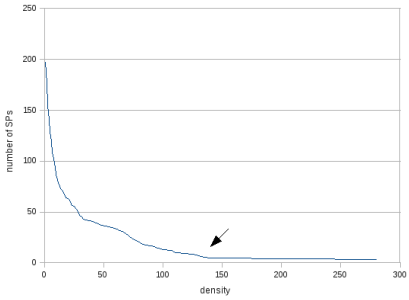
number of the points lying inside the cluster. Many clustering methods have been proposed for the purpose of identifying the significant places [8, 3, 7, 12].

Two thresholds T_{dw}, λ_{min} are given for dwelling time and location density respectively. Figure 2 shows the number of significant places for different value of thresholds.

Figure 2(a) draws the number of significant places against the dwelling time threshold using the 3-month history of a voluntary user. For example, a point at $(60, 53)$ means that there are 53 significant places whose dwelling time is no less than 60 seconds. The diagram also shows that the number of significant places is a constant (=53, or 25 respectively) in between some dwelling time intervals ($60-140$ or $160-280$ seconds resp.). The relatively stable number of SPs implies that the user stays long enough only in places that are of significance to him.



(a) The number of significant places remains invariant when the dwelling time is inside certain interval.

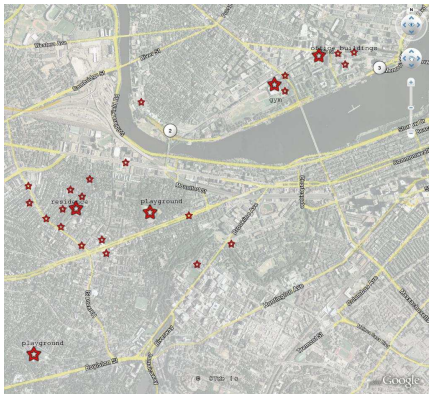


(b) The number of significant places obtained by setting density threshold. The arrow shows the point from which the number of SPs changes slowly with the increase of density.

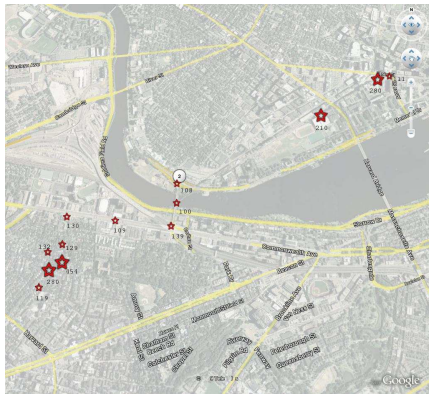
Fig. 2. The number of significant places against T_{dw}, λ_{min}, resp.

Figure 2(b) draws the number of significant places against the density threshold using the 3-month history. For example, a point at $(100, 13)$ means that there are 13 significant places whose density is no less than 100. Figure 2(a) shows a similar property: the number of SPs decreases with greater threshold value of the density, and it remains almost a constant number in a range of density values (e.g., $130 - 250$).

Figure 3(a) shows some locations whose dwelling time is more than 5 minutes are inferred from user's historical data. These places, identified as office, home, playground etc. on the map, are mostly important to the user's activities. Figure 3(b) also shows that the set of the significant places obtained by setting density threshold (=100) is rather similar to the set as seen in Figure 3(a). That is, important places to user, such as office and residence, are included by both schemes for identifying significant places.



(a) Twenty-six locations are identified as significant in terms of dwelling time (over 5 minutes). Google Earth shows that they are office, gym, playground, residence, etc. Some locations on the road have long dwelling time possibly because of traffic lights.



(b) Thirteen locations are identified as significant in terms of density (>= 100). Each significant location is marked by a star and its density number. Four locations with densities over 200 are identified as the residences, a gym, and an office building. The remaining locations are on the paths connecting these four locations.

4 Identifying Representative Paths

Multiple traces between two significant places do not necessarily correspond to the same path. Two nearby traces may actually correspond to two disjoint paths that only join at the two ends, or they may share some portion of a path but split at a branching point. In addition, noises are inherent in the trace samples. Consequently, identifying representative paths implies two closely related problems:

1. The first problem is to classify traces into different paths according to certain metric of trace similarity.
2. The second problem is to form a representative path for those similar traces.

The existing techniques for path reconstruction can be roughly categorized into two types: polygonal reconstruction approach and clustering approach. A path $\mathbb{P}$ is treated as a curve $x(t)$ in the plane. A trace of $\mathbb{P}$ is a sequence P of sample points $\{z_k\}_k$ from this path. Denote $z_k = x_k + n(t_k)$, where $x_k = x(t_k)$ is the path position at time point t_k, and $n(t_k)$ is called noise. Let $P^{(i)} = \{z_k^{(i)} : k = 1, 2, ..., K^{(i)}\}$ denote a trace. The *path reconstruction* problem is to estimate $x(t)$ from a set of N_s traces $\{P^{(i)}\}, 1 \leq i \leq N_s$ for the same path.

The polygonal approach for path reconstruction aims to find a series of $x(t)$ that minimize the mean squared error while preserving the correct temporal ordering of the sample points of the path. Two points $x(t_1)$ and $x(t_2)$ ($t_1 \leq t_2$) are adjacent if no other sample point $x(t)$ exists on the arc $\{x(t) : t_1 < t < t_2\}$. If

$x = x(t)$ is a smooth and twice-differentiable curve in the plane, and P is a finite deterministic sample curve with unknown sampled time points, Amenta et al. [1] defined a polygonal reconstruction of the curve from the sample points that connects every pair of sample points that are adjacent on the curve. Amenta's algorithm preserves the order of the curve correctly when there is sufficient sampling density. Therefore, samples from multiple traces can be taken as a large set of points and paths can be constructed by using the polygonal reconstruction method.

In contrast, the clustering approach is an approximation of the curve reconstruction from noisy samples. To merge multiple traces to produce a single curve, this approach clusters the sample points of all traces and replaces them by their cluster centers. Using this approach, similar traces are generally merged into one simplified curve, but there is no guarantee that the result will preserve the original temporal order.

The existing path reconstruction methods, however, are not suitable for personal positioning. Firstly, the polygonal reconstruction approach is not able to classify the user traces into different path groups. Secondly, the clustering approach cannot guarantee to preserve the original order of the traces. This inspires us to design new path reconstruction algorithms for personal positioning.

Our path reconstruction is carried out in two steps. Firstly, the user's log, or a sequence of position measurements ordered in time, is segmented into multiple disjoint traces according to their similarity under both temporal and spatial metrics. Secondly, similar traces will be consolidated, and some disjoint but geographically close traces can be transformed into connected paths. The resulting representative paths will be further used in map construction.

The output of data segmentation is a set of traces connecting each pair of significant places. The trace connectivity is defined as follows. The traces are samples of the paths in the map of user activities. The connectivity in the traces reflects the user's experience that one location is reachable from another.

Definition 3 (trace connectivity). *Given two threshold values D_{th}, T_{th}, two sample positions $z(t_i), z(t_{i+1}) \in \mathbb{R}^2$ are said to be linked if and only if: $|t_i - t_j| < T_{th}$ and $\|z(t_i) - z(t_j)\| < D_{th}$.*

The distance between two traces is defined by their Hausdorff distance as follows. Let $P = \langle p_1, ..., p_n \rangle$ be a polygonal curve, where $p_i = (x_i, y_i) \in \mathbb{R}^2, 1 \leq i \leq n$ and n denotes the size of P. Let $\overline{p_1 p_2} = \{(x, y) : (x_2 - x_1)y = (y_2 - y_1)x + y_1 x_2 - x_1 y_2, x \in [x_1, x_2]\}$ be the line segment between points p_1, p_2.

Definition 4 (Hausdorff distance). *The distance between a point p and a curve Q, written as $d(p, Q)$, is the shortest distance between p and points on Q, i.e., $d(p, Q) = \min_{q \in Q} \|p - q\|$. The directional distance from curve P to curve Q, written as $d_H(P|Q)$, is given by: $d_H(P|Q) = \max_{p \in P} d(p, Q)$. The Hausdorff distance between two curves P, Q, written as $d_H(P, Q)$, is given by:*

$$d_H(P, Q) = \max\{d_H(P|Q), d_H(Q|P)\}.$$

A path reconstruction algorithm named PCM (pairwise curve-merging) is presented below. After path reconstruction, similar traces will be consolidated, and some disjoint but geographically close traces can be transformed into connected paths. The resulting representative paths will be further used in map construction.

4.1 PCM Algorithm: Reducing Multiple Traces to Representative Paths

The basic idea of PCM algorithm is to iteratively compare every two traces and merge similar segments of the two traces. The algorithm is applied pairwise on the traces and a predefined distance threshold value, ϵ, is given. When the iterative process terminates, the remaining traces will all be distinct based on the distance metric.

The PCM algorithm consists of subroutine $Merge(P, Q, \epsilon)$ in Algorithm 1 and routine $Merge(M, \epsilon)$ in Algorithm 2, where M is a set of traces and P, Q are any two traces of M. Routine $Merge(M, \epsilon)$ uses the subroutine $Merge(P, Q, \epsilon)$ to compare two traces P, Q for the given tolerance ϵ. In subroutine $Merge(P, Q, \epsilon)$, each point p of P is compared with its nearest neighbor q in Q. If p is within the tolerance distance of q, and if p is closer to q than to any adjacent point of p in P, then p is replaced by q. Algorithm 1 is illustrated in Figure 3.

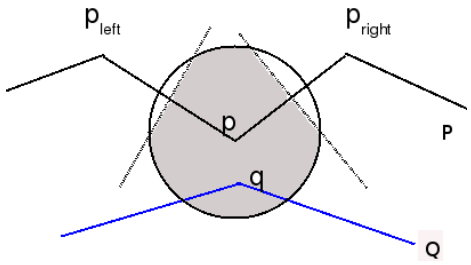


Fig. 3. Illustration of the subroutine $Merge(P, Q, \epsilon)$. The circle is centered at point p with radius ϵ. The two dashed lines are bisectors of segments $\overline{pp_{left}}$ and $\overline{pp_{right}}$ respectively. The shadow indicates that q can be replaced by p according to the algorithm.

Intuitively, the PCM algorithm enforces two conditions when carrying out the merging operation. Firstly, the algorithm repeatedly merges two curves whenever nearby (but not the same) points between the curves are found. Secondly, one polygonal curve partitions the plane into many zones by bisecting the line segments of the curve. Each point of the curve dominates one zone. For one point $p \in P$, only the points of Q that lie inside p's zone can replace p. The second condition ensures the order of the curve points.

Input: ϵ and two curves P and Q
Output: Updated P

for *each point* $p \in P$ **do**
  q = the point in Q that is nearest to p;
  p_{left}, p_{right} = p's left and right point in P, resp.;
  if $p \neq q$ *and* $d(p,q) \leq \min\{\epsilon, d(p_{left}, q), d(p_{right}, q)\}$ **then**
    $p = q$;
  end
end
return P;

Algorithm 1: Merge(P, Q, ϵ): Merging P into Q by tolerance ϵ

Input: ϵ and a set of traces, M
Output: Updated M

changed = True;
while *changed* **do**
  changed = False;
  for $P, Q \in M$ *and* $P \neq Q$ **do**
    $\bar{P}$ = Merge(P, Q, ϵ) in Algorithm 1;
    if $\bar{P} \neq P$ **then**
      changed = True;
    end
  end
end
return M;

Algorithm 2: Merging a set of traces, M, by tolerance ϵ

Lemma 1 below shows that the process of the PCM can terminate. Furthermore, Lemma 2 shows that two nearby curves in Hausdorff distance will still remain close by after merging.

Lemma 1. *PCM can finish in finite number of steps.*

Proof. Each time when two points on different curves are merged, the number of all points on all traces will decrease by 1. So the substitution operation must finish in a finite number of steps. □

Lemma 2. *Let $\bar{P}, \bar{Q}$ denote the new curves obtained from P and Q, respectively, after executing subroutine $Merge(P, Q, \epsilon)$. If $d_H(P, Q) \leq \epsilon$, then*

$$\max\{d_H(\bar{P}, P), d_H(\bar{P}, Q), d_H(\bar{Q}, P), d_H(\bar{Q}, Q)\} \leq \epsilon.$$

Proof. The subroutine $Merge(P, Q, \epsilon)$ does not change Q. So $\bar{Q} = Q$. We only need to prove $d_H(\bar{P}, P) \leq \epsilon$ and $d_H(\bar{P}, Q) \leq \epsilon$ respectively.

Consider each point $x \in \bar{P}$. If $x \in P$, the distance from x to P, $d(x, P) = 0$. Otherwise, it should be the case that $x \in Q$. Since the Hausdorff distance between P and Q is less than or equal to ϵ, we have $d(x, P) \leq d_H(Q, P) \leq \epsilon$. Therefore, $\forall x \in \bar{P} : d(x, P) \leq \epsilon$, which leads to $d_H(\bar{P}|P) \leq \epsilon$.

Now we prove $d_H(P|\bar{P}) \leq \epsilon$. Let x be any point in P, and y be the nearest point in Q to x. Obviously, it is true that $d(x, y) = d_H(x, Q) \leq d_H(P, Q) \leq \epsilon$. According to Algorithm 1, if x is not replaced by y, then x will be kept in $\bar{P}$, i.e., $x \in \bar{P}$; otherwise, there will be $y \in \bar{P}$. In each case, $\exists y \in \bar{P}, s.t., d(x, y) \leq \epsilon$. This means $d(x, \bar{P}) \leq \epsilon$, which leads to $d_H(P|\bar{P}) \leq \epsilon$.

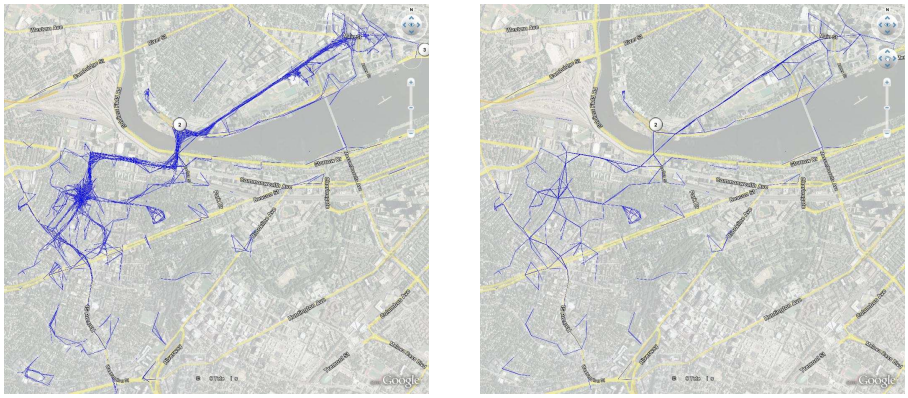
$d_H(\bar{P}|P) \leq \epsilon$ and $d_H(P|\bar{P}) \leq \epsilon$ together prove that $d_H(\bar{P}, P) \leq \epsilon$.

Using similar arguments we can show that $d_H(\bar{P}, Q) \leq \epsilon$. This completes the proof. □

The PCM algorithm will be applied to generate a map. The algorithm reduces the redundancy in the user's log and outputs a simplified representation of paths in the map. In practice, the value of ϵ was chosen to be the road width.

The computational complexity of the PCM algorithm is estimated as follows. Let n be the number of curves, and m be the maximum number of sample points on a trace. The PCM algorithm at most runs $n(n-1)$ iterations of subroutine $Merge(P, Q, \epsilon)$. For $Merge(P, Q, \epsilon)$, we can firstly construct a Voronoi diagram for the points of Q to speed up the nearest-neighbor searching. The construction costs $O(m \log m)$ time, and each nearest-neighbor searching costs $O(\log m)$ time [4]. Since $Merge(P, Q, \epsilon)$ traverses all the points of P, the total running time is $O(m \log m)$. As a result, the running time of PCM algorithm is $O(n^2 m \log m)$.

Figure 4(a) shows the user's traces given as the input of the pairwise curve merging algorithm. Figure 4(b) shows the corresponding output of the pairwise curve merging algorithm. To compare the merged results with the actual roads and streets, the output is also shown on the maps of Google Earth. It can be seen that the redundancy in the traces is dramtically reduced, and the reconstructed paths reflects the actual roads rather well.



(a) A user's raw data segmented into 748 traces, with thresholds T_{th} = 1200 seconds, D_{th} =0.2 km, and a total of 6003 GPS sample points. The disjointed points in the raw data are pruned off.

(b) Paths are reconstructed from the user's traces. The parameter ϵ = 0.050km. The resulted 197 significant places are connected by 289 edges.

Fig. 4. Paths are reconstructed from the user's traces by the PCM algorithm, and shown in Google Earth.

5 Extracting Other Map Attributes

A map of user activities can be constructed after identifying significant places and representative paths from historical log. The map is a 2-dimensional directed graph $G =< V, E >$ without self-cycles, where V is the set of significant places and E is the set of edges. Each vertex $v \in V$ has associated information of the x-y coordinates of the center and the radius of the circle representing the place, dwelling time, and density. The density of a location is defined to be the number of samples within the associated circular region. Each edge $e =< v_1, v_2 > \in E$ records the connectivity from v_1 to v_2. Two vertices v_1, v_2 have an edge when there is any path crossing from v_1 to v_2. Moreover, each edge also records the edge width, the speed and the number of traversals, which are calculated based on the historical records.

A sample map is given in Figure 5(a). The user speed on the edge e is approximated by the average speed of the historical traces that constitute e. For each edge $e =< v_1, v_2 >$, its speed is calculated by the distance between v_1 and v_2, divided by the time difference.

The number of traversals over the edges provides an approximation of the transition probabilities among the vertices. Figure 5(b) shows a subset of the map which contains three vertices and their transitions. Specifically, the transition probability density function (*pdf*) from vertex *src* is approximated by

$$Prob(z|src) \sim normalization\{\lambda(src, z) :< src, z > \in E\},$$

where $\lambda(src, z)$ is the number of traversals on edge $< src, z >$. Let $\pi(v)$ denote the transition probability density function for vertex v. For example, $\pi(A) = [\frac{20}{87}, \frac{29}{87}, \frac{2}{87}, \frac{2}{87}, \frac{16}{87}, \frac{11}{87}, \frac{7}{87}]$ for vertex A in Figure 5(b).

5.1 Speed and Transition Probabilities

Supported by the map with transition probabilities, certain inferences concerning the path likely to be used in between the significant places can be carried out. For example, a "shortest" path from the source vertex to the destination vertex can be obtained by applying a searching algorithm. We use a classical heuristic searching algorithm, *A* search* [13], in order to minimize the total estimated cost. Just like using distance $d(v_1, v_2)$ as a cost function in the shortest-path problem, we interpretate the transition probabilities as a part of the cost on the edges. The cost function is chosen to be

$$J(v_1, v_2) = \frac{d}{\log(1 + \lambda)},$$

where λ is the number of traversals on the edge $< v_1, v_2 >$, and d is the length of the edge. The rationale of this cost function is: when the user visits a path more often than the other paths from the same location, he/she is more likely to reuse this path than the other paths in future. Each use of a path is thus like shortening the distance between two locations by a $\frac{1}{\log(1+\lambda)}$ percent.

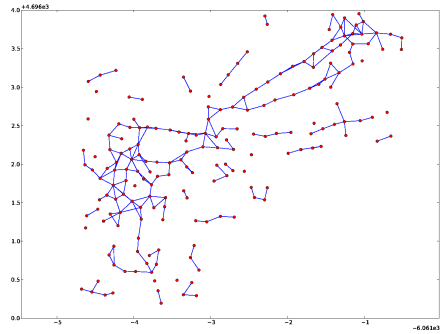
Figure 5(b) shows a "shortest" path in the map for the given source and destination. Figure 6(b) shows the speed distribution for this path. The average speed on the path (evaluated from Figure 6(b)) is 2.8 meters per second.

6 Experiments and Performance Evaluation

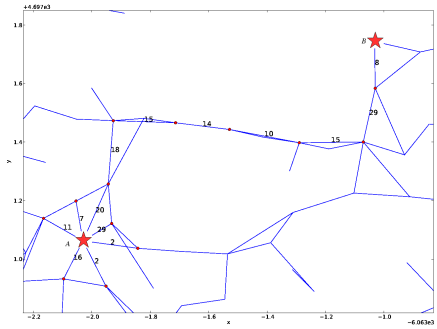
To verify the effectiveness of the user activity map, we set up experiments for a position tracking application as follows. Assume that a map of user activities is given (e.g., Figure 5(a)), which has been extracted from the user's historical positional data.

We test the tracking method for a given trace with the known path from source to destination. The position inference is activated periodically. The method takes into consideration user's speed model derived from the map. Specifically, the average velocity with its standard deviation can be statistically approximated from the historical traces. In this experiment, the estimated trajectory of the user is then compared with the actual trajectory at each time step. The *Root Mean Squared Error* (RMSE) [3], as a widely-used measure in the literature, is included here to evaluate the quality of the output produced by the algorithm. It reflects the deviation of the estimated trajectory to a given reference trajectory.

[3] For a given trajectory $\{x_k\}$ over K_s time steps and its estimate $\{\hat{x_k}\}$, the position RMSE is defined to be $RMSE = \sqrt{\frac{1}{K_s}\sum_{k=1}^{K_s} \|\hat{x_k} - x_k\|^2}$.

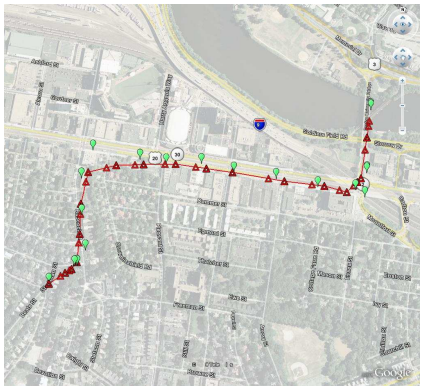


(a) a map of user activities. Each node represents a significant place. Each edge represents the path between two nodes.

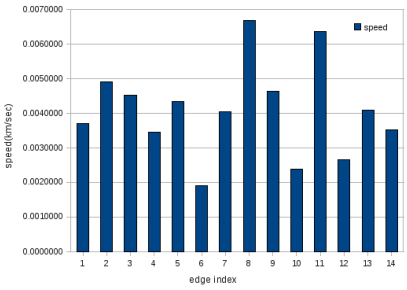


(b) A "shortest" path from source A to destination B is marked between two stars on the map. The numbers on the edge indicate the numbers of traversals. The numbers of traversals from vertex A to its neighbors are also marked.

Fig. 5. A map of user activities is represented as a graph.



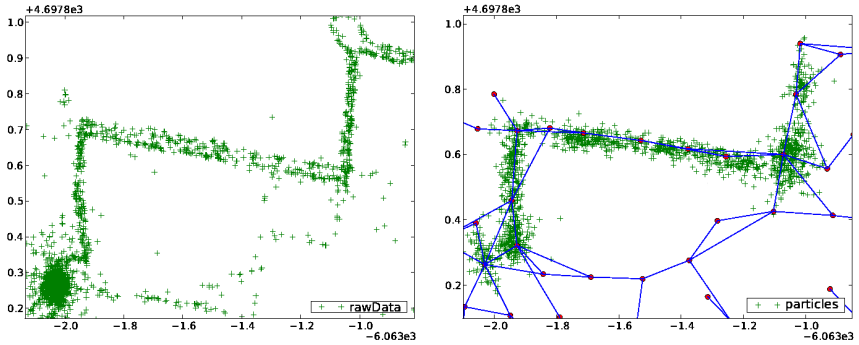
(a) Estimating a trajectory using the map. Each prediction is indicated by a triangle, while each measurement is drawn as a balloon. The predicted trajectory is shown in solid lines.



(b) The average speeds on the edges of the path.

Fig. 6. Searching a "shortest" path in the map.

Figure 6(a) draws all observed positions of a user's trajectory and a predicted trajectory. It shows that the predicted trajectory well matches the user's actual movements, which is even better than merely relying on GPS measurements. This is possible because GPS signals can be momentarily disturbed by external sources. The RMSE for this method is about 4.3 meters.



(a) The sample points in the user historical data.

(b) All the particles generated by the PF process for a trace prediction. The number of iterations, $K_s = 50$. The number of particles, $N_s = 100$. The map is represented by lines and circling points.

Fig. 7. Comparing historical data points and generated particle samples.

6.1 Comparison between Map and Raw Data

Figure 7 compares the distribution of the particles generated by particle filter with the samples from user's historical raw data. It is shown that the two distributions are quite similar, which both reflect the shape of the actual road. However, our map-guided particle filter algorithm does not require storing the huge raw data. It keeps only a map of user activities for the sampling purpose, and chooses the best result from the particles.

7 Discussions and Conclusion

This paper presents the method for mining the pattern of user activity from historical positional data. The method includes the algorithm for constructing significant places and representative paths. It also derives the information about user's speed and transition probabilities. The map of user activity is then applied to position tracking applications. The experiments show that the algorithm can be applied to personal position tracking, and it deals with user's non-linear movement behaviors fairly well. Thus we believe that our algorithm for mining

the user activity map can form a basis for many promising personal location-aware applications.

References

1. Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19:127–153, 2001.
2. M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Trans. on Signal Proc.*, 50(2):174–188, 2002.
3. Daniel Ashbrook and Thad Starner. Learning significant locations and predicting user movement with gps. *International Symposium on Wearable Computing*, pages 101–108, 2002.
4. Franz Aurenhammer and Rolf Klein. Voronoi diagrams. In *Handbook of Computational Geometry*, pages 201–290. Elsevier, 2000.
5. Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.
6. Thomas D'Roza and George Bilchev. An overview of location-based services. *BT Technology Journal*, 21(1):20–27, January 2003.
7. Erhan Gokcay and Jose C. Principe. Information theoretic clustering. *IEEE Transactions on Pattern Aanalysis and Machine Intelligence*, 24(2), February 2002.
8. John Hartigan. *Clustering Algorithms*. John Wiley and Sons Inc., 1975. ISBN: 0-471-35645-X.
9. B. Hoffmann-Wellenhof, Herbert Lichtenegger, and James Collins. *GPS: Theory and Practice. The 3rd ed.* Springer-Verlag, New York, 1994.
10. Thanos Manesis and Nikolaos Avouris. Survey of position location techniques in mobile systems. In *Proc. of the 7th Int. Conf. on Human Computer Interaction with Mobile Devices and Services*, volume 111, pages 291–294, Austria, 2005. ISBN:1-59593-089-2.
11. Eduardo F. Nakamura, Antonio A.F. Loureiro, and Alejandro C. Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys*, 39(3), August 2007.
12. Petteri Nurmi and Johan Koolwaaij. Identifying meaningful locations. In *the Third Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 1–8, San Jose, CA, July 2006. ISBN: 1-4244-0499-1.
13. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, 2nd Edition*. Prentice Hall, 2003. ISBN: 0-13-790395-2.
14. Emiliano Trevisani and Andrea Vitaletti. Cell-id location technique, limits and benefits: An experimental study. In *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04)*, pages 51–60, 2004.
15. Gokhan Yavas, Dimitrios Katsaros, Ozgur Ulusoy, and Yannis Manolopoulos. A data mining approach for location prediction in mobile environments. *Data and Knowledge Engineering'05*, pages 121–146, 2005.