Detecting Significant Locations from Raw GPS Data Using Random Space Partitioning

Nobuharu Kami¹, a) Teruyuki Baba¹ Satoshi Ikeda¹ Takashi Yoshikawa¹ Hiroyuki Morikawa²

Received: October 23, 2011, Accepted: April 2, 2012

Abstract: We present a fast algorithm for probabilistically extracting significant locations from raw GPS data based on data point density. Extracting significant locations from raw GPS data is the first essential step of algorithms designed for location-aware applications. Most current algorithms compare spatial/temporal variables with given fixed thresholds to extract significant locations. However, the appropriate threshold values are not clearly known in priori, and algorithms with fixed thresholds are inherently error-prone, especially under high noise levels. Moreover, they do not often scale in response to increases in system size since direct distance computation is required. We developed a fast algorithm for selective data point sampling around significant locations based on density information by constructing random histograms using locality-sensitive hashing. Theoretical analysis and evaluations show that significant locations are accurately detected with a loose parameter setting even under high noise levels.

Keywords: significant locations, GPS, random partitioning, LSH

1. Introduction

The widespread use of GPS-enabled mobile devices, such as smartphones, enables easy collection of location data and accelerates development of a variety of location-aware applications [1]. In addition to simply visualizing a geographical trajectory of user activities, the essential first step in processing raw GPS data is to extract points of interest (POIs) that represent significant locations on the trajectory such as shopping centers and sightseeing spots. A set of POIs provides a summary of activities, and many algorithms designed for understanding user behavior automatically extract significant locations.

Since a location is assumed to be significant if one stays there for a long time, most algorithms distinguish between “staying” and “moving” segments by comparing spatial/temporal variables, such as stay duration and roaming distance, with fixed threshold values. Although this naive method is intuitive and easily implemented, finding an appropriate threshold value is often difficult in practice since an appropriate threshold value, which is unknown in priori, often depends strongly on input GPS data. For example, we often have to tune parameters such as roaming distance for each case and set a large margin to be on the safe side when the spatial noise level is high. However, excessively large threshold values may even degrade detection quality, e.g., due to crosstalk between neighboring significant locations.

Another method for detecting significant locations is analyzing the spatial distribution of data points and determining high-density locations as significant. Histogram-based methods are perhaps the most straightforward for analyzing density information, but one may encounter a similar parameter setting problem as in threshold-based methods since the result is strongly affected by the binning process that determines the appropriate size and boundary value for each bin. There are many powerful spatial statistics techniques that analyze distribution features of input data points for detecting high-density locations. Generally, they are often powerful for general purpose applications, and finding good parameter values is not as critical as finding them in the methods described above. However, one should pay attention to computation time in processing a massive amount of data points. In short, those spatial statistics techniques are general but often too detailed for detecting significant locations since they provide too much information, which increases computation time.

Since what is really needed is often only the representative geolocation and importance of each significant location, we rather take the approach of quickly making a sketch of precise peak locations in density distribution with a loose parameter setting. To achieve scalability, an algorithm should be also designed in such a way that direct distance computation over a massive number of data points is carefully avoided in analyzing density information.

The core idea for achieving these requirements is to make the histogram-based methods discussed above less independent of the setting of parameters by introducing randomization. To this end, we have developed a randomized algorithm that selectively samples data points from high-density regions using random histograms [2]. Since the obtained subset of the original GPS data is a set of sampled data points where high-density regions are spatially well separated, it is easy to extract a set of waypoints, each of which is a reference point that designates each significant
location and contains information regarding both geographical location and importance.

The remainder of this paper is as follows. Section 2 discusses existing work on detecting significant locations from GPS data and related algorithms. Section 3 describes our algorithm design and theoretical analysis. Section 4 presents the evaluation of the proposed algorithm, and Section 5 concludes the paper.

2. Related Work

Many location-aware services, such as the GeoLife project [1], use algorithms that automatically extract significant locations for understanding user activity patterns. For example, Ashbrook and Starner [3] designed a fixed-threshold-based algorithm for detecting significant locations from GPS data and using a set of those locations for behavior prediction. Hariharan and Toyama [4], Liao et al. [5], [6], and Zheng et al. [7], [8], [9] also developed similar fixed-threshold-based algorithms for detecting segments of GPS data and identifying the most representative point in each segment. Although there are many variations, all these algorithms have the basic principle of using spatial/temporal thresholds for detecting locations where one stays at least for a certain time in a limited region. However, fixed-threshold-based algorithms do not generally work well under high noise levels, and it is difficult to set the optimal parameters. Agamennoni et al. [10] developed an algorithm for extracting significant locations by introducing a score associated with each location using velocity information and linking the top-scored locations to create clusters that designate significant locations. Although this algorithm exhibits good noise tolerance, it still uses a velocity threshold to compute the score. Fixed-threshold-based algorithms are inherently error-prone where one cannot make a good guess about the optimal quantity in the control variable.

On the other hand, there is a different method that takes advantage of spatial statistics techniques for detecting high-density locations as significant locations because they imply that one stays for a long time at such locations. Perhaps the simplest method for detecting high-density locations is to construct histograms by partitioning a space into small bins (cells). However, the detection accuracy using an ordinary histogram is strongly affected by the binning process that determines the appropriate size and boundary value for each bin. A more sophisticated method of partitioning a space is tree-based space indexing, e.g., Octree [11]. Tree-based space indexing is a powerful and efficient spatial data management method that recursively partitions a space into smaller subspaces in response to the distribution of data points. However, as long as the partitioning process is deterministic, a similar problem as in the threshold-based methods described above arises in configuring bins or analyzing data points indexed by the bins for measuring density information. Another effective method is clustering. In particular, hierarchical clustering, such as Ward’s method [12], is useful when one does not know the exact number of stay locations or the noise distribution. The major drawback is long computation time, typically requiring $O(N^3)$ computation time, and it is not suitable for processing a massive amount of data points. Also, one can also use spatial statistics techniques to detect regions around significant locations by assuming that the distribution of data points in staying segments will differ from those in moving segments. One example of spatial statistics techniques is Byers and Rafty [13], which detects features in spatial point processes. They find the distribution of the distance from a randomly chosen point to its k-th nearest neighbor and extract a certain pattern by removing clutter in a given set of spatial data points. However, all methods for intensively analyzing distribution features are very powerful but are often too detailed for our purpose since they provide too much information, which increases computation time.

Unlike the work described above, our design principle is to develop a probabilistic technique that requires only loosely setting parameters for making a rough sketch of significant locations with minimal computation cost. The core idea is to randomize the binning process of histogram-based methods and mitigate the difficulty in setting the parameters while benefiting from the simplicity of these methods. To this end, we have investigated a probabilistic algorithm [2] that detects significant locations using random histograms originally developed for representing feature sets of multimedia objects and analyzing their similarity [14]. We are particularly motivated to apply the space partitioning properties of random histograms to our newly proposed algorithm. The benefit is the randomization of the space partitioning process for simplifying parameter setting even under high noise levels and loose optimization even when we do not precisely know the optimal threshold value in the control variables. Furthermore, the proposed algorithm is expected to provide excellent scalability in response to the increase in the number of data points because it does not require direct distance computation over a massive number of data points.

3. Algorithm Design

The goal with our algorithm described in this section is to return a set of waypoints as output in response to input GPS data $X$. Note that we simplify $X$ as a sequence of periodically recorded location vectors, $X = \{x \in \mathbb{R}^D \}$. The dimension is at most $D = 3$ for GPS data but most of the argument described here can be applied to identifying the reference points to meta-stable segments in a set of general $D$-dimensional data points including a variety of sensor data.

3.1 Design Overview

Figure 1 illustrates the operations of the proposed algorithm. It performs density-dependent random sampling, which returns a subset of input data points sampled selectively from high-density regions, followed by waypoint extraction from a returned subset of the input dataset.

Density-dependent random sampling samples data points selectively from high-density regions in given GPS data by random space partitioning (indexing) using a hash function that gives data points an index number (discussed later in Section 3.2.1) and returns a set of well-separated clusters of data points strongly distributed around peak locations in density distribution. The core idea is to count frequency of data points in each partitioned region (bin) of the space for constructing multiple random histograms and sample high-
3.2 Density-dependent Random Sampling

3.2.1 LSH Sketch to Base C

Locality-sensitive hashing (LSH) is a probabilistic method of hashing objects such that two similar objects are likely to collide into the same bucket in response to the degree of similarity. Let $\mathcal{F}$ be an LSH family for $L_2$ distance. Note that we selected the LSH family for $L_2$ distance in this paper because we consider that $L_2$ distance most naturally represents similarity (geographical distance) between a pair of GPS data points and hence allows us to capture density information. Please refer to Charikar [15] for details of the LSH family definition for various distance measures. A hashing function, $f \in \mathcal{F}$, is implemented by taking advantage of the property of $p$-stable distribution [14], and LSH sketch [16], [17] takes only the least significant bit of the hash value represented in the binary numeral system. We extend the base of this LSH sketch to the general value $C$, which is a positive integer greater than or equal to 2:

$$f(x) = \left\lfloor \frac{a \cdot x + u}{W} \right\rfloor \mod C,$$  

(1)

where $u$ is a real number drawn from a uniform distribution $U(0, W)$, $a$ is a $D$-dimensional vector with entries independently drawn from a standard normal distribution, and $W$ is a parameter called window size. Note that when $C = 2$, Eq. (1) reduces to a binary LSH sketch [16]. Observing that $a \cdot x$ follows a normal distribution $N(0, |a|^2)$, we obtain an analytical form of collision probability $p(d) = \Pr[f(p) = f(q)]$ for two vectors $p, q \in \mathbb{R}^D$ with distance $d = |p - q|$:

$$p(d) = \int_0^W \frac{2}{d} \sum_{k=0}^{n} \phi\left(\frac{kCW + t}{d}\right) \left(1 - \frac{t}{W}\right),$$  

(2)

where $Z$ is a set of integers and $\phi(t)$ denotes the probability density function of the standard normal distribution (see A.1 for deriving Eq. (2)). We call $f(x)$ an atomic label of vector $x$, and concatenation of independent atomic labels constructs a label of vector $x$ as $(f_1(x), \ldots, f_d(x))$. Figure 2(a) plots the collision probability $p(d)$ of two vectors with distance $d$ for $B = 1$ and $C = 2, 3, 5$. The collision probability of two vectors with the same label is given by $[p(d)]^B$, which is illustrated in Fig. 2(b) for $B = 1, 3, 5$ and $C = 2$. We can see that the collision probability almost linearly decreases in response to an increase in $d$ until it quickly converges to $p_{ref} \sim C^{-B}$, the probability with which any pair of vectors with distance in this range collide, after $d$ reaches a certain value. One benefit of introducing parameter $C$ is to limit the range of the hash value and simplify implementation. Another benefit is to provide powerful controllability over the shape of the

![Fig. 1](Image)  
**Fig. 1** Operation diagram. Input data points are indexed by performing random space partitioning using LSH for constructing subset that contains data points sampled from high-frequency bins (rectangles). Waypoints are extracted from subset of input data points as the most representative location.

![Fig. 2](Image)  
**Fig. 2** Collision probability of two vectors with distance $d$. (a) $B = 1$ and $C = 2, 3, 5$ (b) $B = 1, 3, 5$ and $C = 2$. 

The rectangles in Fig. 1 indicate the bins with the two highest frequencies, and the figure illustrates that data points in those bins are returned for each random space partitioning operation as a set of clusters each of which contains data points located in the region where each actual stay location is likely to be located inside (the example in Fig. 1 has two different random space partitioning).

Waypoint extraction reconstructs a set of clusters in such a way that each cluster satisfies a given clustering policy, e.g., how far pairs of distinguishable clusters are from each other. Then the algorithm extracts a set of waypoints from those clusters, each of which contains the most representative point in each cluster and a scoring metric reflecting density information at the location for the purpose of ranking.

© 2012 Information Processing Society of Japan
probability by controlling \((B, C, W)\) and adjusting the characteristic distance up to which collision is distance-sensitive. As discussed later in Section 3.2.3, geometrical interpretation of Eq. (1) is that it performs a random space partitioning using parallel lines whose spatial interval is \(W/|a|\), so all data points located in a region (cell) surrounded by the lines take the same value (fall into the same bin). Therefore, \(W\) should be adjusted in such a way that the cell should span the region around peak location in density distribution. And \(B\) determines the number of directions in which those space partitioning lines are oriented. Generally speaking, \(B\) that is slightly larger than space dimension \(D\) will work well for most cases. If \(C\) takes a large value, the collision probability decays slowly as distance becomes long (\(P_{\text{coll}}\) goes to zero), and we can suppress the probability with which a data point located very far from some stay location happens to take the same hash value that data points around the stay location take. However, the impact of such an undesirable event on performance is fairly small in practice, and it would be rather beneficial to upper-bound the hash value and simplify implementation by using \(C\) that is not so large.

### 3.2.2 Formal Definition of Random Histograms

Let \(h(x) = (f_1(x), \cdots, f_d(x))\) be a labeling function that maps \(X\) to a label space \(L\). If we interpret \(L\) as a set of bin labels, computation of \(h(x)\) for \(x \in X\) determines to which bin \(x\) is registered. Let \(\mathcal{H} = \{h = (f_1, \cdots, f_d) | f_i \in \mathcal{F}\}\) be a set of labeling functions and \(\Lambda_{h \in \mathcal{H}}(x) = \varnothing \in \mathcal{F}\) be a bin with label \(l\), a set of data points mapped to \(h\) is randomly chosen from \(\mathcal{H}\). From a table (a set of bins) of \(X\), \(\Lambda_{h}(x) = \{\Lambda_{h,i}(x)\}_{i \in \Lambda_{h}}\), we can define a random histogram (frequency distribution) of \(X\) over \(L\) by setting a frequency \(\lambda_{h} = |\Lambda_{h,i}|\) for a bin labeled by \(l\). This random histogram allows us to sample data points selectively from high-density regions of the distribution of \(X\). For the positive integer \(Q\), a density-dependent random sampling operation of \(X\), \(\mathcal{R}_{Q}(X)\), returns a set of bins with frequency being among the \(Q\)-highest in \(\Lambda_{h}(x)\), i.e., \(\mathcal{R}_{Q}(X) = \{\Lambda_{h,i}(x) | i =1, \cdots, Q\}\) where \(L = \{l_1, \cdots, l_d\}\) is permuted in descending order of \(\lambda_{h}\). Since histogram construction is a probabilistic operation, we need to repeat the same operation \(\mathcal{R}_{Q}(X)\) independently \(N\) times with \(h_i\) chosen randomly from \(\mathcal{H}\) and maintain a set of sampled bins, \(\Xi_{H,Q,N}(X) = \{\mathcal{R}_{Q}(X), \cdots, \mathcal{R}_{Q}(X)\}\). That note that a large enough value of \(Q\) and \(N\) ensures good accuracy, choosing these optimal values, \(N\) in particular, requires careful consideration because values that are too large directly affect computation time.

### 3.2.3 Randomization Effect

Each set of data points \(\Lambda_{h,i}(x)\) in \(\Xi_{H,Q,N}(X)\) has data points distributed in a strongly localized region around one of the stay locations in \(X\), and the average location over the data points may be a good estimate of the stay location.

Geometrical interpretation of Eq. (1) is that it performs random space partitioning using linear lines. The partitioning lines are randomized from the following three perspectives: (1) zero-position of the lines randomized by the random number \(u\), (2) direction of the lines randomized by the random direction vector \(\hat{a}\) (\(\hat{a}\) denotes a normalized vector of \(a\)), and (3) partitioning width \(W/|a|\) (spatial interval) between the lines randomized by \(|a|\). The first two randomizations, (1) and (2), help to make detection accuracy less dependent on the geolocational configuration of stay locations by attenuating the negative impact, for example detection quality degradation, that results from stay locations happening to be located on partitioning lines or in the same single bin. The last randomization (3) works for making detection accuracy less dependent on noise distribution features. Preferably, partitioning width should be just a little bit larger than the stretch in spatial noise distribution, but one often knows little about the shape of noise distribution (at best a rough estimate of the typical stretch in noise distribution). Furthermore, the partitioning width optimal for one stay location does not necessarily work well for another stay location since noise distribution is generally different from one stay location to another even in the same GPS data. Therefore, randomizing the partitioning width is a good strategy for tackling the uncertainty of noise distribution.

For quantitative evaluation of detection accuracy, we formally define the quality measure as the expected value of the drift distance from a detected location to an actual stay location. For simplicity, we basically conduct one-bit label partitioning using Eq. (1) in the two-dimensional space (i.e., label length \(B = 1\) and dimension \(D = 2\)) and assume that data points around a stay location \(\psi\) are distributed according to a two-dimensional Gaussian noise distribution \(\mathcal{N}(\psi, \Sigma)\), where \(\Sigma = \text{diag}(\sigma^2, \sigma^2)\) for some \(s > 0\). The detected location \(r\) is a centroid of all data points registered in a partition with highest frequency, i.e., the partition that contains \(\psi\) inside and the detection accuracy can be measured using the expected Euclidean distance from the detected location \(r\) to the stay location \(\psi\). Let \(p(\psi | \omega)\) be a probability density function (pdf) of partitioning width \(\xi\) with average value of \(\omega\). Then, the expected value of the drift distance \(\Delta_{\omega}\) is given by a function of \(\omega\) as:

\[
E[\Delta_{\omega}] = \int_0^\infty d\xi p(\psi | \omega)g(\xi | \omega),
\]

where \(g(\xi | \omega)\) is a probability density function (pdf) of partitioning width \(\xi\) with average value of \(\omega\).

### 3.2.4 Computational Complexity

For quantitative evaluation of detection accuracy, we formally define the quality measure as the expected value of the drift distance from a detected location to an actual stay location. For simplicity, we basically conduct one-bit label partitioning using Eq. (1) in the two-dimensional space (i.e., label length \(B = 1\) and dimension \(D = 2\)) and assume that data points around a stay location \(\psi\) are distributed according to a two-dimensional Gaussian noise distribution \(\mathcal{N}(\psi, \Sigma)\), where \(\Sigma = \text{diag}(\sigma^2, \sigma^2)\) for some \(s > 0\). The detected location \(r\) is a centroid of all data points registered in a partition with highest frequency, i.e., the partition that contains \(\psi\) inside and the detection accuracy can be measured using the expected Euclidean distance from the detected location \(r\) to the stay location \(\psi\). Let \(p(\psi | \omega)\) be a probability density function (pdf) of partitioning width \(\xi\) with average value of \(\omega\). Then, the expected value of the drift distance \(\Delta_{\omega}\) is given by a function of \(\omega\) as:

\[
E[\Delta_{\omega}] = \int_0^\infty d\xi p(\psi | \omega)g(\xi | \omega),
\]
3.3 Waypoint Extraction

Once we receive a set of sampled bins $\Xi_{H,Q,N}(X) = \{ \mathcal{Q}_{0,h}[X], \ldots, \mathcal{Q}_{0,b}[X] \}$, the final task is to extract a set of waypoints from it by reconstructing clusters (bins) in such a way that each one contains data points coming from the same stay location. For extracting waypoints, one must define a spatial scale of interest as positional resolution of extracted waypoints. To this end, we introduce the parameter resolution $\zeta$, which designates the shortest permissible distance between the two closest clusters. If the distance between the centroids of two adjacent clusters (waypoint estimates) is less than $\zeta$, then we assume that the centroids represent the same waypoint and these clusters should be merged into a single cluster. Note that a typical spatial stretch in locational distribution of data points in each sampled bin is much shorter than $\zeta$ and the proposed algorithm uses $\zeta$ as a policy parameter to check if extracted centroids are too close to represent different waypoints, in contrast to existing threshold-based algorithms that use such a parameter as a threshold value (for example, roaming distance) to extract stay segments themselves from input data. There are many kinds of possible methods for merging clusters under the merging policy. For example, hierarchical clustering methods such as Ward’s method [12] will work for general $\Xi_{H,Q,N}(X)$. Although the total number of data points in $\Xi_{H,Q,N}(X)$ is significantly reduced compared to $|X|$, it may still take a long time. Observing that data points in each cluster $\Lambda_{h,j}(X) \in \mathcal{Q}_{0,h}[X]$ are already well clustered, we use a simpler method that works well for most of $X$. It repeatedly merges clusters, with a centroid distance less than $\zeta$, into a single cluster. Note that when merging two clusters, we allow each one to have duplicated data points for a centroid of the cluster as a...
better estimate of the stay location. Given a set of reconstructed clusters \( \mathcal{X}(X) = \{ \Lambda_i \} \) and the positive integer \( K \), a set of \( K \) waypoints \( \Omega = \{ (r_i, s_i) \in \mathbb{R}^2 \times \mathbb{R} \} \) is extracted by computing the most representative location \( r_i = \arg \min_{s \in \Lambda_i} |x - y|^2 \) and a scoring metric \( s_i = |\Lambda_i| \) for each \( \Lambda_i \) and selecting \( K \)-most important clusters \( \Lambda_i \) from \( \mathcal{X}(X) \) in descending order of a scoring metric defined \( s_i \).

4. Evaluation

4.1 Evaluation Using Artificially Generated Test Datasets

Using an artificially generated dataset \( X \) in which the noise level is under control, we evaluate the performance and noise tolerance in the parameter setting of the proposed algorithm (denoted by \( \mathcal{A}_{DDRS} \)) by using density-dependent random sampling and comparing it to a typical fixed-threshold-based algorithm (denoted by \( \mathcal{A}_{FT} \)). The input GPS data denoted by \( X \) is a history of periodically (every 15 seconds) recorded locations that contain \( K \) “staying” periods alternating with \( K + 1 \) “moving” periods. In generating \( X \), we used a simple random walk model where one roams around a stay location during the “staying” period. Note that two-dimensional Gaussian noise according to \( N(0, \text{diag}(\sigma^2, \sigma^2)) \) where \( \sigma \) is a parameter that controls the noise level, is added to the stay location to emulate actual measured data points during each “staying” period. For details of the algorithm for generating \( X \), please see A.3.

4.1.1 Performance Measures

Let \( \Omega = \{ r_i \}_{i=1}^k \) be a set of extracted waypoint locations and \( \Psi = \{ \psi_i \}_{i=1}^k \) be a set of actual stay locations. We can then define two quality measures for \( \Omega \); distance \( \delta(\Omega, \Psi) \) and detection ratio \( \chi(\Omega, \Psi) \). \( \delta(\Omega, \Psi) \) quantifies the distance between \( \Omega \) and \( \Psi \), i.e., a set distance. Since each \( \Omega \) should correspond to each actual stay location, we should use a set distance for one-to-one matching defined by,

\[
\delta(\Omega, \Psi) = \min_{\Delta} \frac{1}{\Delta} \sum_{r \in \Omega} \sum_{\psi \in \Psi} a_{\psi, r} |r - \psi| \tag{5}
\]

s.t. \( \forall \psi \in \Psi \), \( \sum_{r \in \Omega} a_{\psi, r} \leq 1 \), \( \forall r \in \Omega \), \( \sum_{\psi \in \Psi} a_{\psi, r} \leq 1 \), \( \forall r \in \Omega \), \( \forall \psi \in \Psi \), \( a_{\psi, r} \in [0, 1] \), \( \Delta = \min |\Omega|, |\Psi| \) = \( \sum_{r \in \Omega} \sum_{\psi \in \Psi} a_{\psi, r} \).

The coefficient \( a_{\psi, r} = 1 \) means that \( r \) is matched to \( \psi \). Note that the number of matches is \( \Delta = \min |\Omega|, |\Psi| \) where each member in \( \Omega \) and \( \Psi \) can be used at most once. The ratio of the number of matches to the number of actual stay locations denoted by \( \varphi = \Delta/|\Psi| \) is another quality measure for indicating how many actual stay locations are detected.

4.1.2 Noise Tolerance in Parameter Setting

For comparison, we also implemented a fixed-threshold-based algorithm \( \mathcal{A}_{FT} \), similar to the one described by Harirahan and Toyama [4], which is simple and intuitive but shows considerably good performance at least under low noise levels. It is a deterministic algorithm that has two threshold values; roaming distance \( l_{rh} \) and stay duration \( t_{rh} \), where \( l_{rh} \) represents the maximum distance that determines the region where one can roam in a “stay segment,” and \( t_{rh} \) is the minimum duration one must stay in a segment for it to be qualified as a “stay segment.” By examining the given dataset \( X \) with these two threshold values, we can detect staying segments with a longer duration than \( l_{rh} \) and a diameter of a staying region less than \( l_{rh} \). The waypoint in each segment is extracted in the same way. Obviously, \( l_{rh} \) strongly affects detection quality, and these parameters are difficult to be correctly set, especially when the spatial noise level is high and unknown. Therefore, tolerance in spatial parameter setting will decrease when noise level is high, and careful parameter tuning is required for maximizing detection quality. The parameter in \( \mathcal{A}_{DDRS} \) corresponding to \( l_{rh} \) is \( \rho = \xi/2 \) (\( \xi \): resolution defined in Section 3.3), which determines the maximum size of the two closest adjacent clusters.

To observe how well \( \mathcal{A}_{DDRS} \) detects waypoints and how much \( \mathcal{A}_{DDRS} \) eases parameter setting, we compared it to \( \mathcal{A}_{FT} \) using datasets with noise levels \( \sigma = (100, 200, 300, 400, 500, 600) \) and measured the tolerance in setting parameters \( \rho \) for \( \mathcal{A}_{DDRS} \) and \( l_{rh} \) for \( \mathcal{A}_{FT} \). Here, using \( \delta(\Omega, \Psi) \) and \( \chi(\Omega, \Psi) \), tolerance \( \pi_p = \{ \pi_p, \pi_{p_a} \} \) in a given parameter \( p = \{ l_{rh}, \rho \} \) is defined by a range in \( p \) that achieves \( \delta(\Omega, \Psi) \) \( \leq \sigma \) and \( \chi = 1 \), where \( \sigma \) is the spatial noise level configured in \( X \). This definition states that as long as \( p \in \pi_p \), we can find any actual stay location with an average distance being at most \( \sigma \) from each corresponding waypoint. Note that we only controlled \( \rho \) for simplicity and all other parameters were configured at the loosely optimized point. Table 1 summarizes the parameter values used for the evaluation.

Figure 5 (a) and (b) plot both the average values of \( \delta(\Omega_{alg}, \Psi) \) and \( \chi(\Omega_{alg}, \Psi) \) over ten sets of independently generated \( X \) for \( \sigma = 200 \) and \( \sigma = 500 \), respectively. The notation \( \Psi \) indicates a set of actual stay locations in \( X \), and \( \Omega_{alg} \) indicates the output of each algorithm \( \mathcal{A}_{alg} \). For \( \mathcal{A}_{alg} = DDRS \) or \( FT \). Note that we also executed ten independent trials for evaluating \( \mathcal{A}_{DDRS} \) for each dataset since it is a probabilistic algorithm and requires taking the average \( \delta(\Omega_{alg}, \Psi) \) and \( \chi(\Omega_{alg}, \Psi) \) of the trials for fair comparison. We also note that each set of \( X \) is generated in such a way that it contains at least one pair of neighboring stay locations in which distance is upper bounded by around \( 2\sigma \sim 3\sigma \) to limit the upper bound of tolerance \( \pi_p \) for \( p = l_{rh} \) and \( \rho \).

Although both \( l_{rh} \) and \( \rho \) depend on the stretch in spatial noise distribution, finding good \( \rho \) in \( \mathcal{A}_{DDRS} \) is not as difficult as finding the appropriate \( l_{rh} \) in \( \mathcal{A}_{FT} \). Basically, the optimal points \( \rho^* \) and \( l_{rh}^* \) which are around the center of the tolerance \( \pi_p \) for \( p = \rho \) and \( l_{rh} \), respectively, increase in response to the increase in \( \sigma \), but the upper/lower bound of the tolerance in both parameters shows a different response. Since data points sampled using \( \mathcal{A}_{DDRS} \) are strongly localized around each stay location, small \( \rho \) works even better.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
<td>2000</td>
</tr>
<tr>
<td>( K )</td>
<td>10</td>
</tr>
<tr>
<td>( N )</td>
<td>10</td>
</tr>
<tr>
<td>( B )</td>
<td>5</td>
</tr>
<tr>
<td>( C )</td>
<td>21</td>
</tr>
<tr>
<td>( W )</td>
<td>10</td>
</tr>
<tr>
<td>( Q )</td>
<td>40</td>
</tr>
<tr>
<td>( t_{rh} )</td>
<td>10 min</td>
</tr>
</tbody>
</table>
and the lower bound \( p \) for dynamic range is important in parameter setting. In fact, Fig. 6 (a) illustrates profiles of the tolerance and dynamic range against \( DR \) for \( \sigma = 200 \) and \( \sigma = 500 \). Points designate average over ten different input datasets, whereas error bars range from min to max values. Note that tolerance is defined as region below dashed line, representing reference distance.

Finally, we would like to briefly discuss computational complexity. We believe that the proposed algorithm has two advantages for reducing computational complexity: elimination of direct distance computation for a massive number of data points and high affinity for scalable distributed computation. Empirically, we observed that \( A_{FT} \) requires a long computation time as expected when \( l_b \) is large, whereas \( A_{DDRS} \) did not show a noticeable difference in response to an increase in \( \rho \). This is because \( A_{FT} \) must perform distance computation among data points, which requires \( O(M^2) \) time for the number of data points denoted by \( M \). Since many data points are involved with distance computation when \( l_b \) is large, \( A_{DDRS} \) that requires no distance computation among data points but computes hash values (labels) individually for each data point has an advantage over \( A_{FT} \) in terms of computation time reduction. In comparison to existing

\[ DR = \frac{(\sigma_{max} - \sigma_{min})}{\sigma_{opt}} \]

under high noise levels, and the crosstalk between data points coming from two neighboring stay locations is also suppressed. For all \( \sigma \), small \( l_b \) degrades both \( \Omega_{FT} \) and \( \Omega_{DDRS} \), and the lower bound \( \pi_{\rho l} \) for \( p = l_b \) is relatively large. This is because \( l_b \) that is too small detects only few stay locations, as indicated by \( \psi(\Omega_{E}, \Psi) \sim 0.1 \), where \( \psi(\Omega_{E}, \Psi) \) for such \( l_b \) is too large to be displayed in the plot. On the other hand, \( A_{DDRS} \) shows good performance even for small \( \rho \) since the partitioning width can take a large value with a certain probability defined by Eq. (4), and it can sample many data points. On the opposite side of the spectrum, the upper bound of both parameters is basically determined by the crosstalk between data points coming from the most adjacent stay locations.

Figure 6 (a) and (b) illustrate profiles of the tolerance and dynamic range \( DR \) against \( \sigma \) ranging [100, 600] for parameter \( p = \rho \) and \( p = l_b \), where \( DR = \frac{(\pi_{\rho l} - \pi_{\rho u})}{\pi_{\rho u}} \) for \( p \) to the optimal point \( \rho' \): \( DR = \frac{(\pi_{u l} - \pi_{u u})}{\pi_{u u}} \). \( DR \) indicates how much ratio the actual parameter value can deviate from the optimal value, and a wide dynamic range generally allows the setting to work for a variety of input datasets having various noise levels. Since it is, in practice, often difficult to accurately estimate the noise level contained in actual GPS datasets, a wide dynamic range is important in parameter setting. In fact, Fig. 6 (a) shows that there is a parameter band \[ \{\max[\pi_{\rho l}(\sigma)], \min[\pi_{\rho u}(\sigma)]\} \] in which \( \rho \) works for all \( \sigma \), whereas there is no \( l_b \) that is universally valid for all \( \sigma \) and hence fixed-threshold-based algorithms require trial-and-error repetitions until an optimal parameter setting for each dataset can be found. We can also observe from Fig. 6 (b) that \( A_{DDRS} \) accordingly shows a large DR value due to a small lower bound \( \pi_{\rho l} \) even for large \( \sigma \).

Fig. 5 Performance comparison of distance \( \delta(\Omega_{alg} \Psi) \) and \( \delta(\Omega_{alg} \Psi) \) between \( alg = A_{DDRS} \) and \( alg = A_{FT} \) for (a) \( \sigma = 200 \) and (b) \( \sigma = 500 \). Points designate average over ten different input datasets, whereas error bars range from min to max values. Note that tolerance is defined as region below dashed line, representing reference distance.

Fig. 6 Tolerance \( \pi_{\rho l} \) (a) and dynamic range \( DR \) (b) for \( p = \rho \) in \( A_{DDRS} \) and \( p = l_b \) in \( A_{FT} \) for various \( \sigma \).

\[ DR = \frac{(\sigma_{max} - \sigma_{min})}{\sigma_{opt}} \]

\[ A_{DDRS} \] and \( A_{FT} \)
that encompasses

The enlargement in Fig. 7 shows the region around one

traveler stopped at several locations, such as sightseeing spots, distributed

around the island. We examined how well the algorithm detected

stopped at several locations, such as sightseeing spots, distributed

prefecture, Japan. The traveler basically traveled by car and

Miymako island, which is famous for panoramic ocean views. We

can see from the trajectory remaining in a localized region that the

traveler stopped and spent some time enjoying the landscape and

the wayptoint is located around the center of the region. For ex-

tracting the waypoints, the algorithm took a negligibly short time

and showed excellent responsiveness. Figure 7 also illustrates

a comparison between the (normalized) scoring metric (the first

metric $s$ of each waypoint) and the (normalized) measured den-

sity at each location. By observing excellent matching between

and the density at the corresponding location, we can confirm

that the proposed algorithm successfully samples data points such

that the scoring metric reflects the density information.

5. Conclusion

We proposed an algorithm that automatically extracts waypoints,

points of reference designating significant locations, from

raw GPS data. In extracting waypoints, the proposed algorithm

probabilistically detects high-density regions using random histo-

grams constructed using LSH-based mapping for computing a label

of bins. Owing to the randomization effects on space partition-

ing, it simplifies the parameter setting even under high noise

level conditions, whereas it also benefits from the simplicity of

histogram-based methods. Since no direct distance computation

is required with our algorithm, it also shows excellent respon-

siveness to an increase in the number of data points. Evaluations

with artificially generated datasets with various noise levels re-

vealed that our algorithm possesses competitive waypoint extrac-

tion ability as well as very wide tolerance in parameter setting

compared to typical fixed-threshold-based algorithms. This re-

sult implies that the proposed algorithm greatly reduces the dif-


culty in setting parameters, and we can use the same parameter

settings for input data with a variety of noise levels. The case

study performed for actual travel data also showed excellent con-

sistency between extracted waypoints and actually visited loca-

tions. Also, the location of each extracted waypoint agrees with

the center of high-density regions, and the scoring metric reflects

actual density. Moreover, the proposed algorithm should show

high scalability in response to the increase in the number of data

points because the computation process is easily distributed in

principle and hence programmed to run on many available soft-

ware frameworks for distributed computing. Furthermore, it is

also applicable to general $D$-dimensional data for finding meta-
stable states in huge datasets. In the future, we will evaluate the scalability of our algorithm for an extremely massive number of actual higher dimensional data points including sensor data. We believe that the proposed algorithm works well for not only many location-aware applications but also applications that process massive high-dimensional datasets.

Acknowledgments This work was partly supported by the National Institute of Information and Communications Technology (NICT), Japan.

References


Appendix

A.1 Collision Probability of LSH to Base C

Given \( p, q \in \mathbb{R}^d \) with distance \( d = |p - q| \), assume that \( f(p) = f(q) = j \), i.e., \( a \cdot p + u = jW + \xi \) for some \( j \in \mathbb{Z} \) and \( 0 \leq \xi < W \). Since \( a \cdot (p - q) \) follows a normal distribution \( N(0, |p - q|^2) \), we can denote \( a \cdot q = jW + u + \xi + \eta \) using a random variable \( \eta \sim N(0, \hat{\sigma}^2) \). Because \( f(p) = f(q) \), \( \eta \) must be located in the range \([ -\xi + kCW : W - \xi + kCW] \) for \( k \in \mathbb{Z} \). Observing that a pdf of a random variable \( a \cdot p + u \) is given by a convolution of a pdf of \( a \cdot p \) and that of uniform distribution \( U(0, W) \), we obtain \( \Pr[a \cdot p + u - f(p) = \xi] = 1/\hat{W} \) independently of \( j \), so \( \xi \) is distributed uniformly at random in the range \([0, W] \). Therefore, \( \Pr(d) \) is given by:

\[
\Pr(d) = \int_0^W \frac{1}{\hat{W}} \frac{1}{2} \sum_{k \in \mathbb{Z}} \int_{kCW - \xi}^{kCW + \xi} d\eta \phi\left( \frac{\eta}{\hat{\sigma}} \right) = \int_0^W \frac{2}{\hat{W}} \sum_{k \in \mathbb{Z}} \phi\left( \frac{kCW + \xi}{\hat{\sigma}} \right) \left( 1 - \frac{\xi}{\hat{W}} \right),
\]

where \( \phi(\cdot) \) is a standard Gaussian function.

A.2 Expected Value of Drift for General D-dimensional Case

The one-bit \( \text{labeling} \) function using Eq. (1) partitions the entire space with parallel linear boundary lines with the partitioning width \( \xi = W/|a| \). Let \( \Lambda \) be a partition defined by a set \( \Lambda = \{ x \in \mathbb{R}^d \mid f(x) = f(r) \} \), where \( r \) is an actual stay location. Without loss of generality, we can set \( r = 0 \) by shifting the stay location to the origin. Then \( \Lambda \) is the partition where the origin is located. For \( x \in \Lambda \), let \( y_{\min} \) and \( y_{\max} \) be a minimum value and maximum value of \( y = \hat{a} \cdot x \), a projection of \( x \) to the direction \( \hat{a} \), respectively. Note that a maximum value of \( y_{\max} \) is given by \( y_{\max} = \hat{r} \). Let \( t \) be \((y_{\max} + y_{\min})/2 \). Since \( x \) is a random variable drawn from a normal Gaussian distribution \( N(0, \Sigma) \), where \( \Sigma = \text{diag}(\sqrt{s_1}, \ldots, \sqrt{s_d}) \), a drift distance from an expected value of \( x \in \Lambda \) to the origin is given by a function of \( \xi \):

\[
g(\xi) = \frac{1}{\hat{W}} \int_{-\xi/2}^{\xi/2} \frac{1}{\sqrt{2\pi\xi}} \left( \int_{x - \xi/2}^{x + \xi/2} dy \left( \sqrt{\Sigma} \right) \right) dy \left( \sqrt{\Sigma} \right),
\]

where \( \sqrt{\Sigma} \) is a pdf of \( y \), which is also a Gaussian function with a standard deviation \( s \). Substitution of \( \theta(y) = \frac{1}{\sqrt{2\pi\xi}} \) into Eq. (A.2) reduces \( g(\xi) \) to:

\[
g(\xi) = \frac{\sqrt{2\pi\xi}}{\sqrt{s_\xi}} \cdot \frac{1}{\sqrt{2\pi\xi}} e^{-\frac{0^2 - \xi^2}{2\hat{\sigma}^2}},
\]

where \( s_\xi \) is an error function and \( \xi = \frac{x}{\sqrt{d\hat{\sigma}^2}} \). Note that \( |a| \) is distributed according to a chi distribution with \( D \) degrees of freedom. A pdf of \( \xi \) is then given by:

\[
\Pr(\xi) = \frac{2^{1-\frac{D}{2}}}{\Gamma\left(\frac{D}{2}\right)} \frac{\hat{W}^D}{\hat{\sigma}^{2D}} e^{-\frac{\xi^2}{2\hat{\sigma}^2}},
\]

where \( \Gamma(\cdot) \) is a Gamma function. By substituting Eqs. (A.3) and (A.4) into Eq. (3), we obtain the analytical form of \( E[\Delta_\varphi] \) for a general \( D \)-dimensional case. Note that the expected value of \( \xi \) is given by:

\[
\omega = E[\xi] = \frac{\Gamma\left(\frac{D+1}{2}\right)}{\Gamma\left(\frac{D}{2}\right)} \frac{\hat{W}}{\hat{\sigma}^{D+1}},
\]

Eq. (4) is derived by Eqs. (A.4) and (A.5) for \( D = 2 \).
A.3 Artificial Test Dataset Generation

The artificially generated test dataset \( X \) is a history of periodically recorded locations that contain \( K \) “staying” periods alternating with \( K + 1 \) “moving” periods. In generating \( X \), we first prepare for \( K \) time slots \( \{ \tau_i \} \) with duration \( \tau \), drawn from the Poisson distribution with average \( \tau \), and randomly allocate these slots without any overlap in \( X \) with the total number of time steps being \( T \). Each time slot \( \tau_i \) indicates the staying period in which one stays at a single location and other parts of the dataset represent the moving period. In the moving period, the location vector \( \psi(t) \) is updated by the randomly generated step vector \( \Delta \psi_m(t) \) such that \( \psi(t + 1) = \psi(t) + \Delta \psi_m(t) \). The step width \( \delta \) is drawn from the Poisson distribution with average \( \delta_0 \), and \( \Delta \psi_m(t) \) is a unit vector with direction determined by random rotation whose angle is drawn from the uniform distribution \( U[-\theta_{\max}, \theta_{\max}] \), where \( \theta_{\max} \) is the maximum possible angle between the previous and next steps. When the moving period ends and the staying period starts at time \( t_0 \), \( \psi(t) \) keeps being updated by the formula \( \psi(t) = \psi(t_0) + \sigma \Delta \psi_s(t) \) until the time slot allocated for the staying period is consumed and the next moving period starts. The second term \( \sigma \Delta \psi_s(t) \), where \( \sigma \) represents the noise level and \( \Delta \psi_s(t) \) is drawn from the two-dimensional normal distribution, indicates the spatial noise introduced to an actual stay location due to a weak signal, e.g., one staying inside a building.

Nobuharu Kami received his B.E. and M.E. degrees from the University of Tokyo, Tokyo, Japan, in 1997 and 1999, respectively. Since 1999, he has been with NEC Corporation and engaged mostly in developing network technologies. His research interests include control theory, and an architecture and algorithm design for network computing systems. He is a member of the IEICE.

Teruyuki Baba received his B.E. and M.E. degrees in Electrical Engineering from the University of Tokyo in 1999 and 2001, respectively. In 2001, he joined NEC Corporation. His research interests include network management, sensor network and resource allocation. He is a member of the IEICE.

Satoshi Ikeda received his M.S. degree in informatics from Graduate School of Informatics, Kyoto University. Since 2007, he has been with the System Platforms Research Laboratories, NEC Corporation. His research interests are in the areas of mobile cloud computing and location-based services. He is a member of the IEICE.

Takashi Yoshikawa received his B.E., M.E., and Ph.D. degrees in instrumentation engineering from Keio University, Kanagawa, Japan in 1988, 1990, and 1998, respectively. He joined NEC Corporation in 1990 and is now a Research Manager of the System Platforms Research Laboratories. He is engaged in the research and development of a distributed and virtualized computing/networking platform and collaborative system.

Hiroyuki Morikawa received his B.E., M.E., and Dr. Eng. degrees in electrical engineering from the University of Tokyo, Tokyo, Japan, in 1987, 1989, and 1992, respectively. Since 1992, he has been in the University of Tokyo and is currently a full professor of the Research Center for Advanced Science and Technology at the University of Tokyo. From 1997 to 1998, he stayed in Columbia University as a visiting research associate. From 2002 to 2006, he was a group leader of the NICT Mobile Networking Group. His research interests are in the areas of computer networks, ubiquitous networks, mobile computing, wireless networks, photonic Internet, and network services. He served as a technical program committee chair of many IEEE/ACM conferences and workshops, Director of the IEICE, Editor-in-Chief of IEICE Transactions of Communications, and he sits on numerous telecommunications advisory committees and frequently serves as a consultant to government and companies. He has received more than 20 awards including the IEICE best paper award in 2002, 2004, and 2010, IPSJ best paper award in 2006, the Info-Communications Promotion Month Council President Prize in 2008, the NTT DoCoMo Mobile Science Award in 2009, and the Rinzaburo Shida Award in 2010. He is a fellow of IEICE, and a member of the IEEE, ACM, ISOC, IPSJ, and ITE.