Journal of Information Processing Vol.20 No.3 757–766 (July 2012)

[DOI: 10.2197/ipsjjip.20.757]

Regular Paper

Detecting Significant Locations from Raw GPS Data Using Random Space Partitioning

Nobuharu Kami[1,a)] Teruyuki Baba[1] Satoshi Ikeda[1] Takashi Yoshikawa[1]
Hiroyuki Morikawa[2]

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 increase 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 smart phones, 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

[1] System Platforms Research Laboratories, NEC Corporation, Kawasaki, Kanagawa 211–8666, Japan
[2] RCAST, The University of Tokyo, Meguro, Tokyo 153–8904, Japan
[a)] n-kami@ak.jp.nec.com

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 Raftary [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, $X = \left\{x \in \mathbb{R}^D\right\}$. 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-

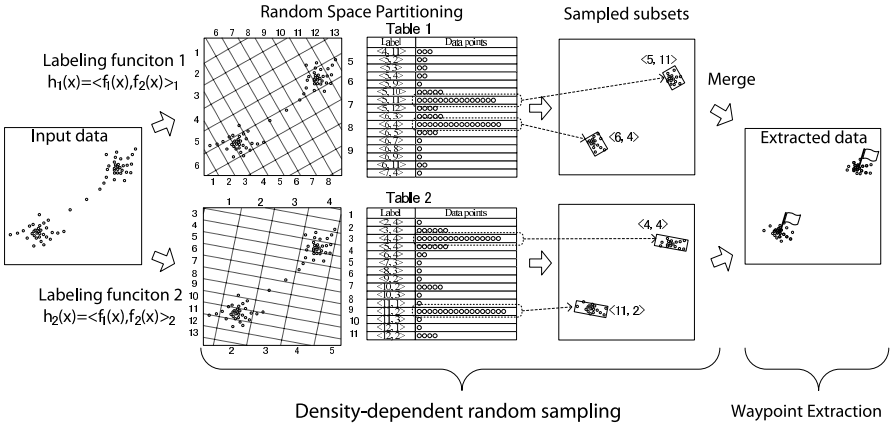
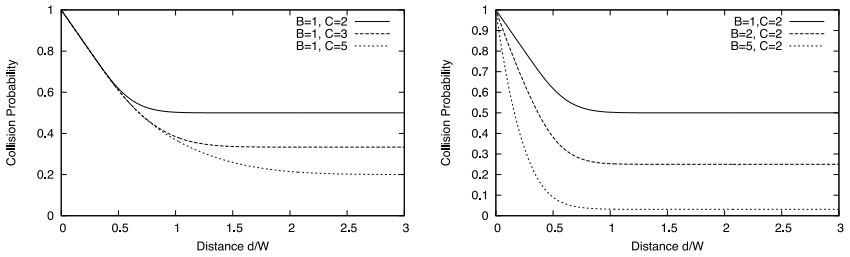


Fig. 1 Operation diagram. Input data points are indexed by performing random space partitioning using LSH for constructing subset that contains data points sampled from high-frequency bins (rectangles). *Waypoints* are extracted from subset of input data points as the most representative location.



(a) $B = 1$ and $C = 2, 3, 5$ (b) $B = 1, 3, 5$ and $C = 2$

Fig. 2 Collision probability of two vectors with distance d.

frequency bins. 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.

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 Chariker [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:

$$f(x) = \left\lfloor \frac{a \cdot x + u}{W} \right\rfloor \ mod\ C\ , \tag{1}$$

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 $\mathcal{N}\left(0, |x|^2\right)$, we obtain an analytical form of collision probability $p(d) = \mathbf{Pr}\left[f(p) = f(q)\right]$ for two vectors $p, q \in \mathbb{R}^D$ with distance $d = |p - q|$:

$$p(d) = \int_0^W dt \frac{2}{d} \sum_{k \in \mathbb{Z}} \phi\left(\frac{kCW + t}{d}\right)\left(1 - \frac{t}{W}\right)\ , \tag{2}$$

where $\mathbb{Z}$ 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 $\langle f_1(x), \cdots, f_B(x)\rangle$. **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 $P_{res} \sim C^{-B}$, 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

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_{res} 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 $h(x) = \langle f_1(x), \cdots, f_B(x)\rangle$ 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 $\mathcal{H} = \{h = \langle f_1, \cdots, f_B\rangle | f_i \in \mathcal{F}\}$ be a set of *labeling* functions and $\Lambda_{h\in\mathcal{H},l}(X) = \{x \in X | l = h(x)\}$ be a *bin* with *label* l, a set of data points mapped to l by h chosen randomly from $\mathcal{H}$. From a *table* (a set of bins) of X, $\Lambda_h(X) = \{\Lambda_{h,l}(X)\}_{l\in L}$, we can define a *random histogram* (*frequency distribution*) of X over L by setting a *frequency* $\lambda_l = |\Lambda_{h,l}|$ 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, $\mathscr{S}_{Q,h}[X]$, returns a set of bins with frequency being among the Q-highest in $\Lambda_h(X)$, *i.e.*, $\mathscr{S}_{Q,h}[X] = \{\Lambda_{h,l_i}(X) | i = 1, \cdots, Q\}$ where $L = \{l_1, \cdots, l_{|L|}\}$ is permuted in descending order of λ_l. Since histogram construction is a probabilistic operation, we need to repeat the same operation $\mathscr{S}_{Q,h_i}[X]$ independently N times with h_i chosen randomly from $\mathcal{H}$ and maintain a set of sampled bins, $\Xi_{\mathcal{H},Q,N}(X) = \{\mathscr{S}_{Q,h_1}[X], \cdots, \mathscr{S}_{Q,h_N}[X]\}$. Note that although 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 $\Lambda_{h,l_i}(X)$ in $\Xi_{\mathcal{H},Q,N}(X)$ 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 ψ are distributed according to a two-dimensional Gaussian noise distribution $\mathcal{N}(\psi, \Sigma)$, where $\Sigma = diag\{s^2, s^2\}$ 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 ψ inside and the detection accuracy can be measured using the expected Euclidean distance from the detected location r to the stay location ψ. Let $p(\xi|\omega)$ be a probability density function (pdf) of partitioning width ξ with average value of ω. Then, the expected value of the drift distance Δ_ω is given by a function of ω as:

$$E[\Delta_\omega] = \int_0^\infty d\xi p(\xi|\omega) g(\xi|s)$$
$$g(\xi|s) = \frac{\sqrt{2}s}{\sqrt{\pi}\xi_s} \int_0^{\xi_s} dt \frac{e^{-(t-\xi_s)^2} - e^{-(t+\xi_s)^2}}{\mathrm{erf}(t+\xi_s) + \mathrm{erf}(t-\xi_s)}, \quad (3)$$

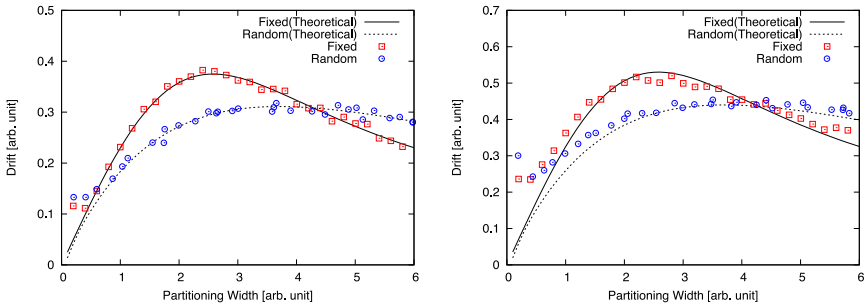
where $\mathrm{erf}(\cdot)$ is an error function and $\xi_s = \frac{\xi}{2\sqrt{2}s}$. If we use Eq. (1) for a partitioning function, the partitioning width ξ is distributed with a pdf:

$$p(\xi|\omega) = \frac{2\omega^2}{\pi\xi^3} e^{-\frac{\omega^2}{\pi\xi^2}}, \quad (4)$$

with the average value being $\omega = \sqrt{\frac{\pi}{2}}W$. For a general D-dimensional case, please see A.2.

Substituting Eq. (4) into Eq. (3), we obtain $E[\Delta_\omega]$ for the random space partitioning using Eq. (1) (denoted by "random-width partitioning" hereafter) in two-dimensional space. Note that if we alternatively use $p(\xi|\omega) = \delta(\xi - \omega)$, where $\delta(\cdot)$ is a delta function, we obtain $E[\Delta_\omega] = g(\omega|s)$ for "fixed-width partitioning" in which direction and zero-position are randomized uniformly at random but a partitioning width takes a fixed value, ω.

For a set of 10,000 artificially generated data points of two-dimensional Gaussian noise with $s = 1$ (i.e., $\Sigma = diag\{1, 1\}$), we measured the average drift from the extracted location to the origin using fixed-width partitioning and random-width partitioning over 500 independent trials. **Figure 3** (a) plots the theoretical



(a) 1-bit partitioning (b) 2-bit pratitioning

Fig. 3 Stay location detection accuracy of random-width partitioning ("Random") using (a) 1-bit *labeling* function and (b) 2-bit *labeling* function compared to that of fixed-width partitioning ("Fixed"). Accuracy is defined as average drift of detected location from actual stay location.

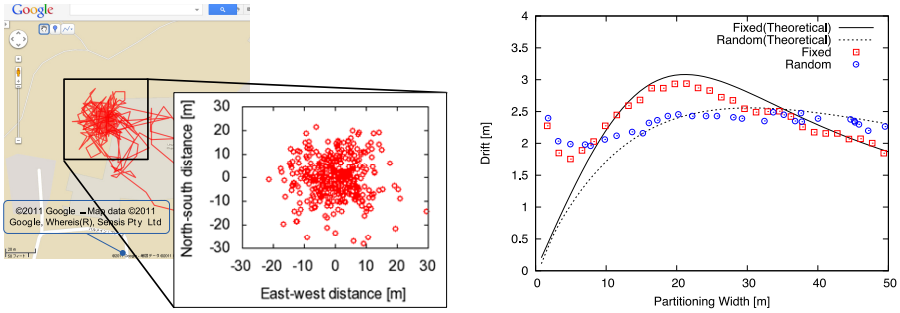


Fig. 4 Data points around actual stay location in actual GPS data (left). Enlargement is set of data points with average location shifted to origin and unit is rescaled in meters. Stay location detection accuracy for this dataset is also shown (right).

curves and simulation results of $E[\Delta_\omega]$ for both fixed-width partitioning and random-width partitioning with various (average) partitioning widths. One noticeable difference between these two partitioning methods is that $E[\Delta_\omega]$ for random-width partitioning clearly exhibits better detection accuracy than for fixed-width partitioning in a range of $\omega < 4$. The partitioning width of $\omega \sim 4s$ corresponds to the width that covers the two-sigma range of the distribution and would be a typical maximum value that one likely sets as a rough estimate of a stretch in noise distribution. For $B \leq D$, a theoretical curve for B-bit *labeling* function is approximated by $E[\Delta_\omega]_B \sim \sqrt{B} \cdot E[\Delta_\omega]$, which is also plotted in Fig. 3 (b) along with simulation results. For $B > D$, we have not derived a well matched theoretical curve, but we at least expect that $\sqrt{B} \cdot E[\Delta_\omega]$ will work as an estimate of the upper bound.

We also conducted the same experiments for an actual GPS dataset that possesses a single stay location and compared the detection accuracy of these two partitioning methods. **Figure 4** visualizes how data points are distributed around a stay location. The number of data points was 387 and the sample variance-covariance matrix was computed as $[\{57.6, -3.0\}, \{-3.0, 77.7\}]$. Figure 4 plots the drift distance $E[\Delta_\omega]$ for these two partitioning methods with a one-bit partitioning function. A similar tendency was observed with the artificial data, and the detection accuracy of random-width partitioning outperformed that of fixed-width partitioning for $\xi < 33 (\sim 4 \times \sqrt{(57.6 + 77.7)/2})$. Note that the deviation from the theoretical curves in small ξ comes from the small number of sampled data points (if the partitioning width is small, the total number of sampled data points also becomes small).

3.3 Waypoint Extraction

Once we receive a set of sampled bins $\Xi_{\mathcal{H},Q,N}(X) = \{\mathcal{S}_{Q,h_1}[X], \cdots, \mathcal{S}_{Q,h_N}[X]\}$, 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* ζ, 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 ζ, 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 ζ and the proposed algorithm uses ζ 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 $\Xi_{\mathcal{H},Q,N}(X)$. Although the total number of data points in $\Xi_{\mathcal{H},Q,N}(X)$ is significantly reduced compared to $|X|$, it may still take a long time. Observing that data points in each cluster $\Lambda_{h_i,l_j}(X) \in \mathcal{S}_{Q,h_i}[X]$ 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 ζ, 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 $\Xi'(X) = \{\Lambda_j\}$ and the positive integer K, a set of K waypoints $\Omega = \{(r_i, s_i) \in \mathbb{R}^D \times \mathbb{R}\}_{1,\cdots,K}$ is extracted by computing the most representative location $r_i = \arg\min_{x \in \Lambda_i} \sum_{y \in \Lambda_i \setminus \{x\}} |x - y|^2$ and a scoring metric $s_i = |\Lambda_i|$ for each Λ_i and selecting K-most important clusters $\{\Lambda_i\}$ from $\Xi'(X)$ in descending order of a scoring metric defined s_i.

4. Evaluation

4.1 Evaluation Using Artificially Generated Test Datasets

Using an artificially generated test 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 $\mathcal{N}\left(0, diag\{\sigma^2, \sigma^2\}\right)$, where σ 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 $\Omega_r = \{r_i\}_{i=1,\cdots,K}$ be a set of extracted *waypoint* locations and $\Psi = \{\psi_i\}_{i=1,\cdots,K}$ be a set of actual stay locations. We can then define two quality measures for Ω_r; *distance* $\delta(\Omega_r, \Psi)$ and *detection ratio* $\varrho(\Omega_r, \Psi)$. $\delta(\Omega_r, \Psi)$ quantifies the distance between Ω_r and Ψ, i.e., a set distance. Since each *waypoint* should correspond to each actual stay location, we should use a set distance for one-to-one matching defined by,

$$\delta(\Omega_r, \Psi) = \min \frac{1}{\Delta} \sum_{r_i \in \Omega_r} \sum_{\psi_i \in \Psi} a_{r_i,\psi_i} |r_i - \psi_i| \tag{5}$$

$$s.t. \quad \forall \psi_i \in \Psi, \quad \sum_{r_i \in \Omega_r} a_{r_i,\psi_i} \le 1 \ ,$$

$$\forall r_i \in \Omega_r, \quad \sum_{\psi_i \in \Psi} a_{r_i,\psi_i} \le 1 \ ,$$

$$\forall r_i \in \Omega_r, \forall \psi_i \in \Psi, \quad a_{r_i,\psi_i} \in \{0, 1\} \ ,$$

$$\Delta = \min\{|\Omega_r|, |\Psi|\} = \sum_{r_i \in \Omega_r} \sum_{\psi_i \in \Psi} a_{r_i,\psi_i} \ .$$

The coefficient $a_{r_i,\psi_i} = 1$ means that r_i is matched to ψ_i. Note that the number of matches is $\Delta = min\{|\Omega_r|, |\Psi|\}$, where each member in Ω_r and Ψ can be used at most once. The ratio of the number of matches to the number of actual stay locations denoted by $\varrho = \Delta/|\Psi|$ is another quality measure for indicating how many actual stay locations are detected.

4.1.2 Noise Tolerance in Parameter Setting

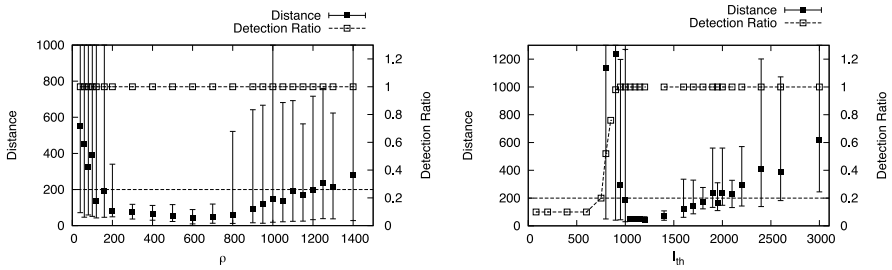
For comparison, we also implemented a fixed-threshold-based algorithm $\mathcal{A}_{FT}$, similar to the one described by Hariharan and Toyama [4], which is simple and intuitive but shows considerably good performance at least under low noise levels. It is a deterministic algorithm that has two threshold values; *roaming distance* l_{th} and *stay duration* t_{th}, where l_{th} represents the maximum distance that determines the region where one can roam in a "stay segment," and t_{th} 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 t_{th} and a diameter of a staying region less than l_{th}. The *waypoint* in each segment is extracted in the same way. Obviously, l_{th} 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_{th} is $\rho = \zeta/2$ (ζ: *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 ρ for $\mathcal{A}_{DDRS}$ and l_{th} for $\mathcal{A}_{FT}$. Here, using $\delta(\Omega_r, \Psi)$ and $\varrho(\Omega_r, \Psi)$, tolerance $\pi_p = [\pi_{p,l}, \pi_{p,u}]$ in a given parameter $p(= l_{th}, \rho)$ is defined by a range in p that achieves $\delta(\Omega_r, \Psi) \le \sigma$ and $\varrho = 1$, where σ 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 σ from each corresponding *waypoint*. Note that we only controlled ρ 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 $\delta(\Omega_{r,alg.}, \Psi)$ and $\varrho(\Omega_{r,alg.}, \Psi)$ over ten sets of independently generated X for $\sigma = 200$ and $\sigma = 500$, respectively. The notation Ψ indicates a set of actual stay locations in X, and $\Omega_{r,alg.}$ indicates the output of each algorithm $\mathcal{A}_{alg.}$ for $alg. = DDRS$ 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 $\delta(\Omega_{r,alg.}, \Psi)$ and $\varrho(\Omega_{r,alg.}, \Psi)$ 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 π_p for $p = \text{ł}_{th}$ and ρ.

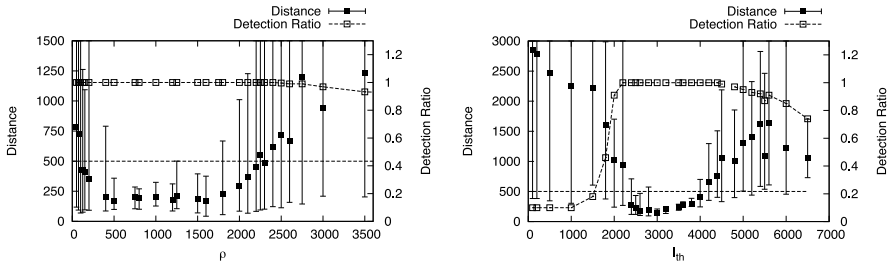
Although both l_{th} and ρ depend on the stretch in spatial noise distribution, finding good ρ in $\mathcal{A}_{DDRS}$ is not as difficult as finding the appropriate l_{th} in $\mathcal{A}_{FT}$. Basically, the optimal points ρ^* and l^*_{th}, which are around the center of the tolerance π_p for $p = \rho$ and l_{th}, respectively, increase in response to the increase in σ, 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 ρ works even

Table 1 Parameters used for evaluation.

$\lvert X\rvert = 2000$	# of data points (total steps) for X
$K = 10$	# of stay locations for X
$N = 10$	# of *random histograms*
$B = 5$	bit length
$C = 21$	base number
$W = \rho$	window size
$Q = 10$	# of bins for sampling
$t_{th} = 10$ min	stay duration for $\mathcal{A}_{FT}$ (corresponding 40 data points)

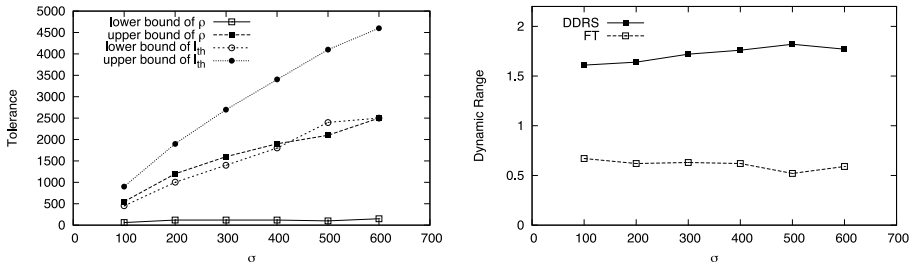


(a) $\mathcal{A}_{DDRS}$ (left) and $\mathcal{A}_{FT}$ (right) for $\sigma = 200$



(b) $\mathcal{A}_{DDRS}$ (left) and $\mathcal{A}_{FT}$ (right) for $\sigma = 500$

Fig. 5 Performance comparison of distance $\delta(\Omega_{r,alg.}, \Psi)$ and $\varrho(\Omega_{r,alg.}, \Psi)$ between $alg. = \mathcal{A}_{DDRS}$ and $alg. = \mathcal{A}_{FT}$ 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.



(a) Tolerance (b) Dynamic range

Fig. 6 Tolerance π_p (a) and dynamic range DR (b) for $p = \rho$ in $\mathcal{A}_{DDRS}$ and $p = l_{th}$ in $\mathcal{A}_{FT}$ for various σ.

under high noise levels, and the crosstalk between data points coming from two neighboring stay locations is also suppressed. For all σ, small l_{th} degrades both $\delta(\Omega_{r,FT}, \Psi)$ and $\varrho(\Omega_{r,FT}, \Psi)$, and the lower bound $\pi_{p,l}$ for $p = l_{th}$ is relatively large. This is because l_{th} that is too small detects only few stay locations, as indicated by $\varrho(\Omega_{r,FT}, \Psi) \sim 0.1$, where $\delta(\Omega_{r,FT}, \Psi)$ for such l_{th} is too large to be displayed in the plot. On the other hand, $\mathcal{A}_{DDRS}$ shows good performance even for small ρ 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 σ ranging $[100, 600]$ for parameter $p = \rho$ and $p = l_{th}$, where DR is the ratio of tolerance width $(\pi_{p,u} - \pi_{p,l})$ for p to the optimal point p^*: $DR = (\pi_{p,u} - \pi_{p,l})/p^*$. 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 $\left[\max\{\pi_{\rho,l}(\sigma)\}, \min\{\pi_{\rho,u}(\sigma)\}\right]$, in which ρ works for all σ, whereas there is no l_{th} that is universally valid for all σ 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 $\mathcal{A}_{DDRS}$ accordingly shows a large DR value due to a small lower bound $\pi_{\rho,l}$ even for large σ.

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 $\mathcal{A}_{FT}$ requires a long computation time as expected when l_{th} was large, whereas $\mathcal{A}_{DDRS}$ did not show a noticeable difference in response to an increase in ρ. This is because $\mathcal{A}_{FT}$ 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_{th} is large, $\mathcal{A}_{DDRS}$ that requires no distance computation among data points but computes hash values (*labels*) individually for each data point has an advantage over $\mathcal{A}_{FT}$ in terms of computation time reduction. In comparison to existing

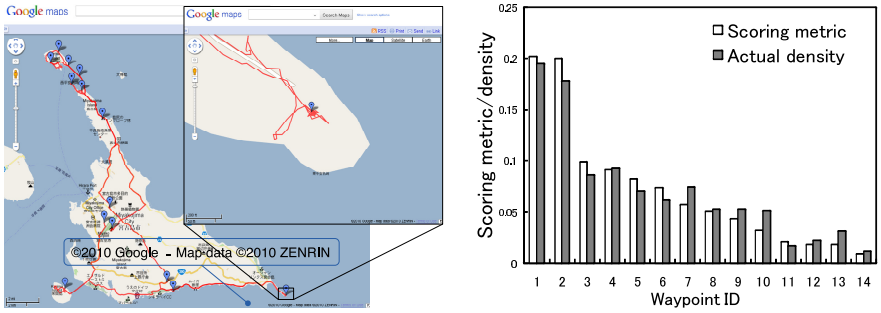


Fig. 7 Trajectory and extracted *waypoints* for actual travel data on *Miyako* island (left). Enlargement is an *waypoint* designating Higashi-Hennazaki. Comparison between normalized scoring metric of each *waypoint* and normalized actually measured density (defined by number of data points in circle with radius of 40 meters around measured point) is also shown (right). All maps are displayed using Google Maps [19].

deterministic histgram-based algorithms that do not require distance computation either, $\mathcal{A}_{DDRS}$ requires an additional computation cost resulting from randomization for easy parameter setting: merging sampled bins for extracting *waypoints*. The merging operation requires distance computation among centroids of sampled bins. This is, however, often negligible because this centroid distance computation depends on constant parameters N and Q, which are the repetition number of random space partitioning trials (the number of *tables*) and the number of sampled bins for each *table*, respectively. Another advantage is high affinity for distributed computation. An input dataset needs to be divided into several segments for distributed computation. Since distance computation between data points in different segments requires considerable computation cost, algorithms that require distance computation among those data points may seriously experience long computation latency. Thus, this segmentation process should be carefully conducted, for example, in such a way that data points coming from a same stay location should not be separated into different segments. In contrast, since $\mathcal{A}_{DDRS}$ only requires individual *label* computation for each data point and frequency of data points having the same *label*, it is easily implemented by taking advantage of many available software frameworks for distributed computing, such as MapReduce [18], in a very straightforward way without any concern for performance degradation due to segmentation.

4.2 Case study: Extracting *Waypoints* from Actual Travel Data

We applied the proposed algorithm to actual GPS data recorded when a person traveled throughout Miyako island, in Okinawa prefecture, Japan. The traveler basically traveled by car and stopped at several locations, such as sightseeing spots, distributed around the island. We examined how well the algorithm detected the spots that the traveler actually visited. The GPS data contained 1,617 data points, each of which was recorded at about 15-second intervals. **Figure 7** shows the trajectory of the GPS data and the fourteen top-ranked *waypoints* extracted with the algorithm. The algorithm successfully returned *waypoints* that corresponded well to all major locations that the traveler actually visited; sightseeing spots, shops, a gas station, a hotel, an airport, *etc*. The enlargement in Fig. 7 shows the region around one *waypoint* that encompasses *Higashi-Hennazaki*, the most eastern cape on Miyako 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 *waypoint* is located around the center of the region. For extracting 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 density at each location. By observing excellent matching between s 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 histograms* constructed using LSH-based mapping for computing a *label* of bins. Owing to the randomization effects on space partitioning, 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 responsiveness to an increase in the number of data points. Evaluations with artificially generated datasets with various noise levels revealed that our algorithm possesses competitive *waypoint* extraction ability as well as very wide tolerance in parameter setting compared to typical fixed-threshold-based algorithms. This result implies that the proposed algorithm greatly reduces the difficulty 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 consistency between extracted *waypoints* and actually visited locations. 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 beucase the computation process is easily distributed in principle and hence programmed to run on many available software 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

[1] Microsoft Research: GeoLife Project, available from ⟨http://research.microsoft.com/en-us/projects/geolife/⟩ (accessed 2011-10-17).
[2] Kami, N., Enomoto, N., Baba, T. and Yoshikawa, T.: Algorithm for Detecting Significant Locations from Raw GPS Data, *Proc. 13th Intl. Conf. Discovery Science*, pp.221–235 (2010).
[3] Ashbrook, D. and Starner, T.: Using GPS to Learn Significant Locations and Predict Movement across Multiple Users, *Personal and Ubiquitous Computing*, Vol.7, No.5, pp.275–286 (2003).
[4] Hariharan, R. and Toyama, K.: Project Lachesis: Parsing and Modeling Location Histories, *GIScience*, pp.106–124 (2004).
[5] Liao, L., Fox, D. and Kautz, H.: Location-based Activity Recognition using Relational Markov networks, *Proc. 19th Intl. Joint Conf. Artifcial Intelligence*, pp.773–778 (2005).
[6] Liao, L., Patterson, D.J., Fox, D. and Kautz, H.: Building Personal Maps from GPS Data, *Ann. New York Academy of Sciences*, pp.249–265 (2006).
[7] Zheng, Y., Liu, L., Wang, L. and Xie, X.: Learning Transportation Mode from Raw GPS Data for Geographic Applications on the Web, *Proc. 17th Intl. Conf. World Wide Web*, pp.247–256 (2008).
[8] Zheng, Y., Li, Q., Chen, Y., Xie, X. and Ma, W.-Y.: Understanding Mobility Based on GPS Data, *Proc. 10th Intl. Conf. Ubiquitous Computing*, pp.312–321 (2008).
[9] Zheng, Y., Zhang, L., Xie, X. and Ma, W.-Y.: Mining Interesting Locations and Travel Sequences from GPS Trajectories, *Proc. 18th Intl. Conf. World Wide Web*, pp.791–800 (2009).
[10] Agamennoni, G., Nieto, J. and Nebot, E.M.: Mining GPS Data for Extracting Signifcant Places, *Proc. 2009 IEEE Intl. Conf. Robotics and Automation*, pp.855–862 (2009).
[11] Samet., H. and Kochut, A.: Octree Approximation and Compression Methods, *Proc. 1st Intl. Symp. 3D Data Processing Visualization and Transimission*, pp.460–469 (2002).
[12] Ward, J.H.: Hierarchical Grouping to Optimize an Objective Function, *American Statistical Association*, Vol.58, No.301, pp.236–244 (1963).
[13] Byers, S. and Raftery, A.E.: Nearest-neighbor Clutter Removal for Estimating Features in Spatial Point Processes, *J. American Statistical Association*, Vol.93, No.442, pp.596–604 (1998).
[14] Datar, M., Immorlica, N., Indyk, P. and Mirrokni, V.S.: Locality-sensitive Hashing Scheme Based on p-stable Distributions, *Proc. 12th Annual Symp. Computational Geometry*, pp.253–262 (2004).
[15] Charikar, M.S.: Similarity Estimation Techniques from Rounding Algorithms, *Proc. 34th Annual ACM Symp. Theory of Computing*, pp.380–388 (2002).
[16] Dong, W., Wang, Z., Charikar, M. and Li, K.: Efficiently Matching Sets of Features with Random Histograms, *Proc. 16th ACM Intl. Conf. Multimedia*, pp.179–188 (2008).
[17] Dong, W., Charikar, M. and Li, K.: Asymmetric Distance Estimation with Sketches for Similarity Search in High-Dimensional Spaces, *Proc. 31st Annual Intl. ACM SIGIR Conf. Research and Development on Information Retrieval*, pp.123–130 (2008).
[18] Dean, J. and Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters, *Proc. 6th Symp. Operating Systems Design and Implementation* (*OSDI'04*), San Francisco, USA, pp.137–150 (2004).
[19] Google: Google Maps, available from ⟨http://maps.google.com/⟩ (accessed 2011-10-17).

Appendix

A.1 Collision Probability of LSH to Base *C*

Given $p, q \in \mathbb{R}^D$ with distance $d = |p - d|$, assume that $f(p) = f(q) = j$, i.e., $a \cdot p + u = jW + \xi$ for some $j \in \mathbb{Z}$ and $0 \le \xi < W$. Since $a \cdot (p - q)$ follows a normal distribution $\mathcal{N}(0, |p-q|^2)$, we can denote $a \cdot q = jW + u + \xi + \eta$ using a random variable $\eta \sim \mathcal{N}(0, d^2)$. Because $f(p) = f(q)$, η 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 $\mathbf{Pr}[a \cdot p + u - f(p) = \xi] = 1/W$ independently of j, so ξ is distributed uniformly at random in the range $[0, W)$. Therefore, $p(d)$ is given by:

$$\begin{aligned} p(d) &= \int_0^W d\xi \frac{1}{W} \sum_{k\in\mathbb{Z}} \int_{kCW-\xi}^{kCW-\xi+W} d\eta \frac{1}{d}\phi(\frac{\eta}{d}) \\ &= \int_0^W dt \frac{2}{d} \sum_{k\in\mathbb{Z}} \phi\left(\frac{kCW+t}{d}\right)\left(1 - \frac{t}{W}\right), \end{aligned} \tag{A.1}$$

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

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

The one-bit *labeling* function using Eq. (1) partitions the entire space with parallel linear boundary lines with the partitioning width $\xi = W/|a|$. Let Λ be a partition defined by a set $\Lambda = \left\{x \in \mathbb{R}^D | f(x) = f(r)\right\}$, 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 Λ 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 $y_{\min} + \xi$. Let t be $(y_{\max} + y_{\min})/2$. Since x is a random variable drawn from a normal Gaussian distribution $N(0, \Sigma)$, where $\Sigma = diag(s^2, \cdots, s^2)$, a drift distance from an expected value of $x \in \Lambda$ to the origin is given by a function of ξ:

$$g(\xi|s) = \frac{1}{\xi} \int_{-\xi/2}^{\xi/2} dt \left| \frac{\int_{t-\xi/2}^{t+\xi/2} dy\, y\, \theta(y)}{\int_{t-\xi/2}^{t+\xi/2} dy\, \theta(y)} \right|, \tag{A.2}$$

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

$$g(\xi|s) = \frac{\sqrt{2}s}{\sqrt{\pi}\xi_s} \int_0^{\xi_s} dt \frac{e^{-(t-\xi_s)^2} - e^{-(t+\xi_s)^2}}{\mathrm{erf}(t+\xi_s) + \mathrm{erf}(t-\xi_s)}, \tag{A.3}$$

where $\mathrm{erf}(\cdot)$ is an error function and $\xi_s = \frac{\xi}{2\sqrt{2}s}$. Note that $|a|$ is distributed according to a chi distribution with D degrees of freedom. A pdf of ξ is then given by:

$$p_D(\xi) = \frac{2^{1-\frac{D}{2}}}{\Gamma\left(\frac{D}{2}\right)} \frac{W^D}{\xi^{D+1}} e^{-\frac{W^2}{2\xi^2}}, \tag{A.4}$$

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

$$\omega = E[\xi] = \frac{\Gamma\left(\frac{D-1}{2}\right)}{\sqrt{2}\,\Gamma\left(\frac{D}{2}\right)} W. \tag{A.5}$$

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 τ_i drawn from the Poisson distribution with average τ_s and randomly allocate these slots without any overlap in X with the total number of time steps being T. Each time slot τ_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 $\delta\Delta\psi_m(t)$ such that $\psi(t+1) = \psi(t) + \delta\Delta\psi_m(t)$. The step width δ is drawn from the Poisson distribution with average δ_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 σ 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.