Spaceprint: a Mobility-based Fingerprinting Scheme for Public Spaces

Mitra Baratchi, Geert Heijenk, Maarten van Steen
University of Twente
Enschede, The Netherlands
{m.baratchi,geert.heijenk,m.r.vansteen}@utwente.nl

ABSTRACT

In this paper, we address the problem of how automated situation-awareness can be achieved by learning real-world situations from ubiquitously generated mobility data. Without semantic input about the time and space where situations take place, this turns out to be a fundamental challenging problem. Uncertainties also introduce technical challenges when data is generated in irregular time intervals, being mixed with noise, and errors.

Purely relying on temporal patterns observable in mobility data, in this paper, we propose *Spaceprint*, a fully automated algorithm for finding the repetitive pattern of similar situations in spaces. We evaluate this technique by showing how the latent variables describing the category, and the actual identity of a space can be discovered from the extracted situation patterns. Doing so, we use different real-world mobility datasets with data about the presence of mobile entities in a variety of spaces. We also evaluate the performance of this technique by showing its robustness against uncertainties.

KEYWORDS

Spatial profiling, WiFi scanning, Spatio-temporal analysis, Mobility pattern mining, Point of interest classification

ACM Reference format:
Mitra Baratchi, Geert Heijenk, Maarten van Steen. 2016. *Spaceprint*: a Mobility-based Fingerprinting Scheme for Public Spaces. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 9 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Many situational-aware decision-support systems rely on the capability of describing the situation in spaces. Ideally, these descriptions are updated automatically as the situation in a space changes. Typical examples include automatically identifying a bottleneck on a road, or a suspicious activity in an airport. A means for learning and comparing situations from the abundance of ubiquitously generated mobility data (GPS coordinates, check-in records, WiFi detections, etc.) can open the door to many applications that require such automated situational-awareness. As a first step towards this goal, in this paper, we investigate how mobility data can represent the repetitive pattern of situations in spaces.

In many cases, a specific space with a known category such as a library, a canteen, or a classroom, will exhibit repetitive visiting patterns characterizing a recurring situation. Such patterns effectively operate as a spatial *fingerprint* of situations. Moreover, we can expect that similar places will often have similar fingerprints. Although in many cases these fingerprints would seem to be static, it is really the usage of a space that determines its meaning, which at various occasions may differ from the location's original intended purpose. For example, in special situations an office space is used for throwing a party or, likewise, an apartment can be rented out as if it were a hotel room. We argue that to better understand or reason about the situation at hand, it is important to understand to what extent the situation in a space adheres to its regular fingerprint, and otherwise, to what extent it resembles any other well-known fingerprints.

In this light, we address the question to what extent we can automatically *measure* a location's fingerprint of situations from available mobility data. To realize situation-aware systems that are generally applicable, we focus on creating these fingerprints in a completely unsupervised manner. This implies that these fingerprints should be created from raw mobility data without additional human input of any kind. Therefore, unlike most previous related research in mobility data analysis, our method operates without a feature-engineering phase.

To this end, we study the presence pattern of devices by looking at when and how long they appear in a space. More specifically, we make the following contributions. (1) We propose a feature set that can generically characterize all possible presence patterns in a space. (2) We use such a feature set to extract the fingerprint of the repetitive situations in spaces (*Spaceprints*) in a fully unsupervised manner. (3) We evaluate the robustness of this fingerprinting scheme in the presence of common sources of uncertainty in ubiquitously collected mobility data sets. (4) We validate our method by showing its classification performance using a WiFi-based detection data set and a Foursquare check-in dataset.

The rest of this paper is organized as follows. Related work is presented in Section 2. We present our problem definition and a sketch of our proposed fingerprinting framework in Sections 3 and 4, respectively. The details of our fingerprinting scheme is presented in Section 5. The performance of this scheme is evaluated in Section 6. A number of remarks conclude this paper in Section 7.

2 RELATED WORK

There are two ways to study the movement of individuals in a space when dealing with mobility data (referred to as the Lagrangian and

Conference'17, Washington, DC, USA
© 2016 ACM. 978-x-xxxx-xxxx-x/YY/MM...$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

Eulerian approaches [1]). First, from the perspective of an individual, one may ask about the whereabouts of a person: what are the locations that someone visits? When do those visits take place, and for how long? The research in this direction concentrates on extracting mobility patterns that reflect an *individual mobility fingerprint* for frequent behavior [8, 10], periodic behavior [9], social behavior [14], etc.

Second, from the perspective of a specific location, one may ask about the visits to that location: When do they take place? How long do they last? Which visits happen again? In this case, one focuses on extracting a *spatial mobility fingerprint.* Previous related research in extracting these spatial fingerprints have either focused on improving the individual mobility prediction models [7, 15] or on bringing sense to raw location coordinates in terms of meaningful labels. Research in methods to describe the meaning of locations, primarily concentrates on how accurately trajectories can be segmented into sections with basic semantics such as *stop and move* areas [13], or *points of interest* [12]. With the prevalence of context-aware mobile applications which needed more than just such low-level semantics, further research has been performed to extract more detailed semantics about spaces interpreted in colloquial terms such as *home*, *work*, *cinema*, *restaurant*, etc. Using a single person's frequent trajectory patterns, semantics about few predefined places (e.g. *home*, *work*) have been extracted in [4, 11]. In a more general case, and when extracting semantics from data involving a large population of mobile entities, a common approach has been enriching data with higher level semantics using additional sources, or using common sense assumptions, for instance, presence at night for *home*, presence at working hours for *offices* or presence in weekends for *leisure related locations*. Some examples of additional sources of semantics are verbal terms used by people in social media such as twitter [6, 16], and third party geographical sources [17]. In [3] the authors use a number of selected mobility features (e.g., crowded hours, number of visitors per month) along with application usage, and proximity to other devices to label a group of known spaces. Knowing the semantic labels of spaces within a region, higher level regional semantics have also been extracted to label *areas* such as those used for *housing*, and *businesses* [20].

The spatial fingerprints made thus far are either meant for labeling location coordinates using *engineered* features in a supervised manner or use additional *semantic input* to enrich data with context from other sources. These approaches are not generic and cannot be taken further to realize automated situation-awareness in dynamically changing spaces purely using mobility data. To reach this goal, our approach in spatial fingerprinting from mobility data is different from all previous research as it performs in a fully unsupervised manner purely exploiting presence patterns in spaces. Specifically, instead of looking for features that characterize spaces based on their *semantic meaning*, we look for features that can characterize periods of time in a space based on its *dynamic situation.*

3 PROBLEM DEFINITION

We define a model based on data acquired from any system that allows for the collection of mobility data in terms of presence or detection of mobile entities in a well-defined region of space. A **detection** record is a tuple $\langle \mathrm{d}, \mathrm{s}, t\rangle$ with d being the identifier of the detected mobile entity, s being the identifier of the space where the entity d was detected, and t being the timestamp of the detection. A variety of mobility-data collection systems can result in such a dataset. These include, for example, WiFi detection of mobile devices near access points, GPS coordinates discretized in grid maps, and check-in records collected from location-based social networks.

Given a set of detection records **DT**, we are interested in a **spatial fingerprint** SP(s) which defines the core repeating temporal presence patterns of space s. Assuming that *latent* variables such as the unique *identity* of the space and its semantic *category* result in such a fingerprint, we demand that this scheme exhibit the following: (1) each space has a unique fingerprint, (2) spaces having the same category have similar fingerprints, and (3) spaces having different categories have different fingerprints.

4 FRAMEWORK OVERVIEW

Our goal is to define a spatial fingerprint that summarizes the situations in a space in terms of repeating presence patterns over time. One might think of creating a time series by measuring a feature from the detections over equally sized duration windows with a specific resolution, such as the number of detections (*feature*) during every hour (*resolution*) of a day (*duration*). By averaging the value of these features over many duration windows (e.g., over 100 days), the fingerprint can be extracted. If these features were enough to fingerprint a space, with a suitable classification algorithm and suitable distance function, we would also be able to classify different spaces from one another based on their fingerprint. However, there are many unknown factors that require attention. The challenge in our case is to identify (1) the **features**, (2) an appropriate **resolution** and **duration window**, and (3) a suitable **distance function**. Compared to these three, the choice of a classification or clustering algorithm is a trivial one. Typically, these challenges are addressed based on intuition. For instance, we may assume that a weekly pattern governs the visits to a space or that a resolution of one hour is enough to provide the necessary level of detail. This intuitive approach, however, limits the applicability of the fingerprinting scheme. The proposed fingerprinting scheme in this paper addresses these challenges through systematically finding appropriate parameter settings in an unsupervised manner. We define a spatial fingerprint as follows.

Definition 4.1. (Spatial fingerprint) The ***fingerprint for the space*** s *is a triplet* $\mathrm{SP}(\mathrm{s}) = \langle \mathbf{V}, FD, FR\rangle$, *with* ***feature vector*** $\mathbf{V} = [v_1, \ldots, v_n]$, *of which each element v_i represents the value of a specific feature. FD is the* ***fingerprint duration****, indicating the total time over which the fingerprint is configured. FR is the* ***fingerprint resolution****, indicating the minimum time interval over which detections are sampled to extract features. FD is a multiple of FR:* $\exists r \in \mathbb{N} : FD = r \cdot FR$.

Algorithm 1 summarizes the fingerprinting framework *Spaceprint* proposed in this paper. Let **DT** denote a set of detections and $t_{min}(\mathbf{DT}) = \min\{t|\langle \mathrm{d}, \mathrm{s}, t\rangle \in \mathbf{DT}\}$, i.e., the timestamp of the first detection. Likewise, we have $t_{max}(\mathbf{DT})$ for the timestamp of the last detection and $\tau(\mathbf{DT}) = t_{max}(\mathbf{DT}) - t_{min}(\mathbf{DT})$ for the duration of collecting **DT**. Denote by $\overline{\mathbf{DT}}$ the set of detections

$\{\langle \mathrm{d}, \mathrm{s}, t - t_{min} \rangle | \langle \mathrm{d}, \mathrm{s}, t \rangle \in \mathbf{DT}\}$, i.e., the set of same detections, but now transformed such that the first detection starts at time 0. Finally, we use the notation $\mathbf{DT}(\mathrm{s}) = \{\langle \mathrm{d}, \mathrm{s}, t \rangle | \langle \mathrm{d}, \mathrm{s}, t \rangle \in \mathbf{DT}\}$ to denote the set of detections by space s.

The spatial fingerprint is composed of three components. We have a separate procedure for extracting each of these components. We use the procedure *fingerprintParameters* for calculating the optimal fingerprint parameters, being the fingerprint duration (FD) and fingerprint resolution (FR). The procedure *vectorize* constructs the feature vector over a dataset spanning a duration of FD time units using resolution FR. The final procedure *vectorAverage* computes the average over multiple feature vectors. In the following sections, we will represent how we create the feature vector and determine the fingerprint duration and resolution.

```
Algorithm 1: Spaceprint
  Data: $\mathbf{DT}(\mathrm{s})$
  Result: $\mathrm{SP}(\mathrm{s}) = \langle \mathbf{V}, FD, FR \rangle$
  $(FD, FR) = fingerprintParameters(\mathbf{DT}(\mathrm{s}))$;
  for $(i = 0; i < \tau(\mathbf{DT}(\mathrm{s}))/FD; i = i + 1)$ do
      $\mathbf{DT}_i = \{\langle \mathrm{d}, \mathrm{s}, t \rangle \in \overline{\mathbf{DT}}(\mathrm{s}) | i \cdot FD \leq t < (i + 1) \cdot FD\}$;
      $\mathbf{V}_i = vectorize(\overline{\mathbf{DT}}_i, FD, FR)$;
  $\mathbf{V} = vectorAverage(\mathbf{V}_{i \in 1 \ldots \tau(\mathbf{DT}(\mathrm{s}))/FD})$;
  return $(\mathbf{V}, FD, FR)$;
```

5 METHODOLOGY

5.1 Presence patterns

As mentioned before, the most important step in fingerprinting spaces is identifying suitable (computable) features that represent the situation in spaces. Let us consider selecting features that may be relevant for such purpose and are observable from mobility data. For example, intuitively one may think of static features such as opening or closing hours, peak hours, group sizes, number of individuals, etc. However, features that can define the situation in a space are numerous and intuitively coming up with a comprehensive set of features that could characterize any thinkable situation in spaces is practically impossible.

Without any intuitive assumptions about features that define the situation in a space, the only measurable feature from detections is related to presence pattern of mobile entities. In reality, each space observes many of these patterns formed due to the variety of the intention of its visitors. For instance, consider the presence pattern of shopkeepers in a shop versus that of their clients. A shopkeeper enters the shop around opening time and leaves around closing time. The clients may appear during opening hours and stay for some time based on their intention (browsing or shopping). We assume that the situation in space is reflected in the overlapping visits of different groups of mobile entities. To consider this variety, we define a presence pattern such that it reflects **the synchronous presence of a group of mobile entities during a specific period of time**. Such a pattern represents a group of mobile entities entering a space, staying there for a specific amount of time, and then leaving it at the same time. Entering and leaving a space may be repeated multiple times as well. Extracting these patterns from a detection dataset can be achieved by counting the number of mobile entities in a window with a specific starting time, t_{start}, and duration, τ. As detections are registered in discrete time intervals, the presence should be detected in all sampling intervals of length T_s in τ. Correspondingly, we define presence features with the following *template* to quantify the intensity of such patterns.

Definition 5.1. (Presence feature) A ***presence feature*** *$PF(t_{\mathrm{start}}, \tau, T_s)$ over a space represents the number of mobile entities that were detected in all $\lceil \tau / T_s \rceil$ consecutive sampling intervals of length T_s within a measurement window, starting at time t_{start} and lasting for a duration of τ time units.*

By ranging over all possible values of the parameters t_{start}, τ, and T_s, the feature template mentioned above will lead to numerous presence features. Table 1 summarizes the possible range of these parameters for creating presence features as defined in Definition 5.1.

Table 1: The possible ranges for the parameters of a presence pattern, given a fingerprint duration FD and fingerprint resolution FR.

Variable Name	Range
t_{start}	$\{0 \leq k \cdot FD/FR < FD, k \in \mathbb{N}\}$
τ	$\{0 \leq k \cdot FD/FR < FD - t_{start}, k \in \mathbb{N}\}$
T_s	$FR \ll T_s \ll FD$

These parameter ranges are motivated as follows. Assume that we measure detections at a given location for a specific duration of time, FD, and that the mobile entities are detected at a frequency f_p (and period $T_p = 1/f_p$). For now, also assume that the fingerprinting resolution FR is equal to this period as well ($T_p = FR$). We later show how to extract the optimal value for FR which is possibly bigger than T_p. The basis of our approach is to sample the number of mobile entities within a specific **duration window** $W = \langle t_{start}, \tau \rangle$ with a **sampling frequency** f_s (with period $T_s = 1/f_s$). Both W and f_s can vary. The duration window can have any starting time and length as long as the window is smaller than FD. Therefore, we require that $\tau \leq FD$ and $t_{start} + \tau < FD$. To count the number of mobile entities, we need to sample detections with a period T_s. Obviously, as it does not make sense to sample with a speed faster than the mobile entity's detection generation speed, we require that $T_s \geq FR$ (or T_p). Additionally, T_s cannot be larger than the duration window, i.e., $T_s \leq \tau$. Note that the feature vector $\mathbf{V}$ can now be considered as an ordered list of normalized presence features.

5.2 Feature vector

As mentioned before, the presence features can be created by counting mobile entities based on every possible combination of starting time, stay duration, and sampling period, t_{start}, τ, T_s. Considering that we have n possible combinations by ranging over these parameters, we will have an n-dimensional vector composed from different presence features. Algorithm 2 (*vectorize*) represents the way of constructing a feature vector for a given space based on a collection of mobile entity detections.

The input of this algorithm is a set of detections $\mathbf{DT}(s)$ for a specific space s. If W is a duration window, we write $\mathbf{DT}[W]$ to denote the subset of detections that occurred inside W. If T_s is a sampling period, then $[\mathbf{DT}]_{T_s}$ denotes the list of $\lceil (t_{max}(\mathbf{DT}) - t_{min}(\mathbf{DT}))\rceil / T_s$ buckets, with the i^{th} bucket containing all detections that occurred during the i^{th} interval of length T_s.

The essence of *vectorize* is to count the number of mobile entities that were detected during an entire duration window, W, when sampled with the period T_s. We systematically explore every possible duration window and sampling period for a given fingerprint duration FD and resolution FR. There are three loops for covering all possible values for parameters t_{start}, τ and T_s. In each iteration, by counting the mobile entities that appeared in the intersection of all buckets of $[\overline{\mathbf{DT}}[W]]_{T_s}$, a presence feature is created.

Algorithm 2: vectorize

```
Data: $\overline{\mathbf{DT}}$, $FD$, $FR$
Result: $\mathbf{V}$
$\mathbf{V} = []$;
for ($t_{\text{start}} = 0; t_{\text{start}} < FD; t_{\text{start}} = t_{\text{start}} + FR$) do
    /* iterate over all durations                    */
    for ($\tau = FR; \tau \leq FD - t_{\text{start}}; \tau = \tau + FR$) do
        for ($T_s = FR; T_s \leq \tau; T_s = T_s + FR$) do
            /* iterate over all sampling periods   */
            if ($\tau\ mod\ T_s = 0$) then
                $W = \langle t_{start}, \tau \rangle$;
                $u = \bigcap([\overline{\mathbf{DT}}[W]]_{T_s})$; /* get the ID of
                    mobile entities present in all
                    buckets of window W                  */
                $append(\mathbf{V}, count(u))$; /* append to V the
                    total number of mobile entities  */
return($\mathbf{V}/\max(\mathbf{V})$);
```

The complexity of Algorithm 2 presented above is $O((\frac{FD}{FR})^4)$. This complexity comes from the three for loops and an intersection over all elements of $[\overline{\mathbf{DT}}[W]]_{T_s}$. By reusing the results of the intersection operation this complexity can be reduced to $O((\frac{FD}{FR})^3)$. A meaningful sampling period is the one that can break the duration window into its integer factors ($\tau\ mod\ T_s = 0$). In that case, the third loop will repeat only for integer multiples of τ, thus reducing complexity further. It should be noted that both FD and FR are fixed and do not depend on the size of detection dataset. Therefore, creating the feature vectors can be performed in a scalable manner.

Figure 1 represents an example feature vector $\mathbf{V} = [v_1, \ldots, v_n]$ calculated using Algorithm 2. This vector is acquired by vectorizing one week of data with a resolution of a day (i.e., $FD = 7$ days and $FR = 1$ day). It can be readily verified that there are $n = 57$ elements in $\mathbf{V}$. The first element, v_1, corresponds to the number of mobile entities that were detected during the first day: $W = \langle 0, 1 \rangle$, with resolution $T_s = 1$. Element v_2 counts the mobile entities that were present during both the first and the second day: $W = \langle 0, 2 \rangle$, with sampling period $T_s = 1$. Likewise, v_3 represents mobile entities during the either first or second day, and so on. In this example, v_{15} represents a window spanning over the entire week ($W = \langle 0, 7 \rangle$) and sampled with the sampling period $T_s = 1$. It thus counts the number of mobile entities that were present in all seven days. Typically, these encompass all static, that is, non-mobile entities. Also interesting is v_{16}, which represents a duration window spanning over all seven days ($W = \langle 0, 7 \rangle$), but with a sampling period $T_s = 7$ of also the entire week. As such, it counts the total number of mobile entities who showed up at least once during the entire week, regardless how long they stayed.

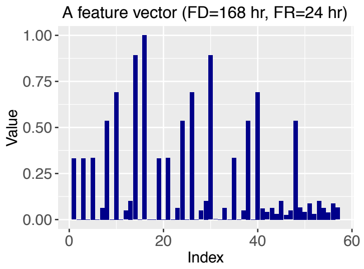


Figure 1: An example representation of a feature vector

Our goal is to use such feature vectors to compare spaces to each other based on visiting patterns of devices. In doing so, we need to take into account that the values in a single vector can vary widely, which is entirely due to the fact that we wish to include all possible values for duration windows and sampling periods into a single data structure. As a consequence, we need to avoid that high values (which are perfectly natural due to our method of counting) dominate our perspective of difference between two vectors. In order to take these natural differences between elements into account, we choose a distance metric based on the so-called *Canberra Distance* [5].

Definition 5.2. (Feature vector distance function) Given two feature vectors $\mathbf{V}$ *and* $\mathbf{V}^*$ *of equal length n, calculated using the same pair of fingerprint parameters FD and FR, their mutual distance is* $\Delta(\mathbf{V}, \mathbf{V}^*) = \frac{1}{n} \sum_{i=1}^{n} \frac{|v_i - v_i^*|}{|v_i| + |v_i^*|}$

5.3 Fingerprint parameters

We now concentrate on finding appropriate values for the fingerprint duration FD and the fingerprint resolution FR. Concerning the **fingerprint duration**, note that we are looking for the period (in the formal sense) of repetitive or self-similar situation. There are many ways of doing this, for example through Fourier analysis or computing autocorrelations. In our approach, we look for a series of consecutive fixed-length windows $W_1, W_2, W_3, \ldots$ such that for a given set of detections $\mathbf{DT}$, we have a minimal accumulated distance between all possible pairs of vectorized subsets of detections $\mathbf{DT}[W_i]$ and $\mathbf{DT}[W_j]$. Our only variable is the length of all such windows, and the length that *minimizes* the accumulated distance is our fingerprint duration.

Determining the best **fingerprint resolution** is a bit trickier. The resolution, as shown in Algorithm 2, determines the minimum sampling period and directly determines the number of features in the vector. Therefore, other than increasing the computational costs, a too detailed FR may also introduce the problem of over-fitting. It is desirable to choose the resolution such that all significant differences between feature vectors are preserved. Therefore, what we are looking for is a resolution that *maximizes* the distance between two vectorized datasets. The assumption is that we have already determined the periodicity *FD* in a series of detections. By looking at two consecutive datasets of duration *FD*, a resolution *FR* that maximizes the mutual distance of their vectorized versions effectively captures all differences that would have also been captured by a smaller resolution. At the same time, such a resolution will capture more differences than any larger resolution (which would show a smaller distance between the two vectorized datasets).

Lemma 5.3 tells us that such a distance-maximizing resolution actually exists.

LEMMA 5.3. *In case a space has a periodic fingerprint, there exists an optimal fingerprint resolution FR over which the distance between consecutive feature vectors is maximized.*

PROOF. We prove that having a constant fingerprint duration, by either increasing or decreasing the resolution, the distance between two features vectors $\mathbf{V}$,$\mathbf{V}^*$ approaches zero. Let $\delta_i = \frac{|v_i - v_i^*|}{|v_i| + |v_i^*|}$. When we increase the resolution, we will necessarily increase the length n of a constructed feature vector. As both v_i and v_j have positive values, regardless the changes in δ_i when increasing n, we will always see that $\delta_i \leq 1$, while the number of elements for which $\delta_i > 0$ will increase to a finite number M. This is due to the fact that elements acquired with a smaller sampling period ($T_s < T_p$) are meant to count the mobile entities that were detected with a speed much faster than the actual detection speed of mobile entities and there are hardly any of them. As a consequence,

$$\lim_{n \to \infty} \Delta(\mathbf{V}, \mathbf{V}^*) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \delta_i \leq \lim_{n \to \infty} \frac{1}{n} M = 0$$

Analogously, as the resolution decreases, the length of a feature vector decreases and will eventually be 1 when $FR = FD$. A vector of length one will have only one element, which after normalization, is equal to 1. Therefore,

$$\lim_{n \to 1} \Delta(\mathbf{V}, \mathbf{V}^*) = \frac{1}{1} \frac{|1 - 1|}{|1 + 1|} = 0$$

□

Algorithm 3 summarizes the procedure of extracting the fingerprint parameters.

```
Algorithm 3: fingerprintParameters
Data: DT(s), $r$ (such that $FD = r \cdot FR$)
Result: $FD$,$FR$
for ($i = 1; i < \tau(\mathbf{DT})/(2r); i = i + 1$) do
    $m = i \cdot r$;
    for ($j = 0; j < \tau(\mathbf{DT})/m; j = j + 1$) do
        $\mathbf{DT}_j = \{\langle \mathrm{d}, \mathrm{s}, t \rangle \in \overline{\mathbf{DT}}(\mathrm{s}) | j \cdot m \leq t < (j+1) \cdot m\}$;
        $\mathbf{V}_j^i$= vectorize($\overline{\mathbf{DT}}_j, m, i$);
$FD$= $r \cdot \arg\min_i \sum_{j,k} \Delta(\mathbf{V}_j^i, \mathbf{V}_k^i)$;
for ($i = 1; i \leq FD; i = i + 1$) do
    if ($FD$ mod $i = 0$) then
        for ($j = 0; j < \tau(\mathbf{DT})/FD; j = j + 1$) do
            $\mathbf{DT}_j = \{\langle \mathrm{d}, \mathrm{s}, t \rangle \in \overline{\mathbf{DT}}(\mathrm{s}) | j \cdot FD \leq t < (j+1) \cdot FD\}$;
            $\mathbf{V}_j^i$= vectorize($\overline{\mathbf{DT}}_j, FD, i$);
$FR$= $\arg\max_i \sum_{j,k} \Delta(\mathbf{V}_j^i, \mathbf{V}_k^i)$;
return($FD$,$FR$)
```

6 EVALUATION

In this section, we show how *Spaceprint* feature vectors can be used for finding repetitive situation patterns in spaces. We also evaluate the performance of *Spaceprint* in presence of uncertainties.

Evaluation approach: We expect that the fingerprint of situations in a space can reflect from which and what kind of space it is extracted. Therefore, we evaluate our method to see how the latent variables of the semantic category of a space and its unique identity are reflected in the fingerprint of the space. Our evaluations are on the basis of using the feature vectors mentioned before in unsupervised classification to infer these latent variables. Any unsupervised classification or clustering algorithm can be used for such purpose. In our experiments we have used *K-means* clustering algorithm using our defined distance function from Definition 5.2.

Baseline: To the best of our knowledge, there is no prior work in classifying or creating situation fingerprints for spaces purely based on presence patterns. However, a common approach in considering space-specific temporal features, is calculating hourly densities [7, 18]. Therefore, we compare *Spaceprint* with a *density-based* approach as baseline. The *density-based* feature vectors are represented by $\mathbf{V}_d = [d_0, ..., d_{\frac{FD}{FR}-1}]$ where each element d_i represents the number of mobile entities appearing in the window $W = \langle i \cdot FR, FR \rangle$. These vectors are extracted using the same fingerprint parameters (FD, FR).

6.1 Test with synthetic dataset

6.1.1 Synthetic dataset generation. Our goal of using a synthetic dataset is to test the robustness of the fingerprinting scheme against uncertainties, yet in a controlled fashion. We proceed as follows.

Generating virtual spaces: First, a total of *NS* different *virtual spaces* are created with presence patterns that are repeated over *FD* time units and mobile entities being detected with the same detection frequency ($T_p = 1$). A virtual space is characterized by a tuple $\langle \mathbf{P}, NP \rangle$ of presence patterns $\mathbf{P}$ each having size *NP*. Complying with the definition of presence patterns in Section 5.1, each presence pattern represents a group of mobile entities entering and leaving a space simultaneously. We denote a pattern by the tuple $\langle \mathbf{GS}, NG, t_{start}, \tau \rangle$ where **GS** is a set of mobile entity IDs of size *NG*. Parameter t_{start} is the start time of the pattern, and τ is its duration. We assume that each mobile entity generates a detection record at times $t_{start} + k$ for $0 \leq k < \tau$. A virtual space thus represents an actual space, such as a coffee corner, a class room, and so on, for which we assume that a fingerprint is known.

Generating instances of spaces: From each virtual space, *NI* number of instances are generated which will represent the presence patterns of the same space over multiple epochs of length *FD* with a modified situation. These instances are generated by varying different sensitivity test parameters as explained later.

Generating the mobility dataset: Note that each pattern implicitly defines a set of detections. Each mobile entity d ∈ **GS** is assumed to generate detections at times $t_{start}, t_{start}+1, \ldots$. As a consequence, $\langle \mathbf{GS}, NG, t_{start}, \tau \rangle$ for a space s gives rise to a set of detections $\mathbf{DT}(\mathrm{s}, \mathbf{GS}) = \{\langle \mathrm{d}, \mathrm{s}, t_{start}+k \rangle | \mathrm{d} \in \mathbf{GS}, 0 \leq k < \tau\}$. We construct a dataset by taking the union of sets $\overline{\mathbf{DT}}(\mathrm{s}, \mathbf{GS})$ for patterns generated for s.

Clustering: Each set of detections $\overline{\mathbf{DT}}(\mathrm{s})$ is vectorized using Algorithm 2 with a precomputed pair of *FD* and *FR* and the accuracy of clustering fingerprint instances to their correct cluster is presented. For the input *K* of the *K-means* algorithm, we use the number of original fingerprints as the number of clusters. The success of the algorithm in clustering is finding *NS* distinct clusters by mapping the instances of the same space to the same cluster.

6.1.2 *Sensitivity test parameters.*

Our goal is to test the clustering accuracy of the fingerprinting technique. There are in general two groups of parameters that affect the quality of clustering. The first group represents the inherent uncertainty present in presence patterns. That is, in real-world settings it is unlikely that a presence pattern repeats itself exactly the same way. The other group represents the noise introduced by data-collection instruments, such as, for example, missing detections due to collision. Below we specify how we apply the effects of these parameters on the synthetic dataset.

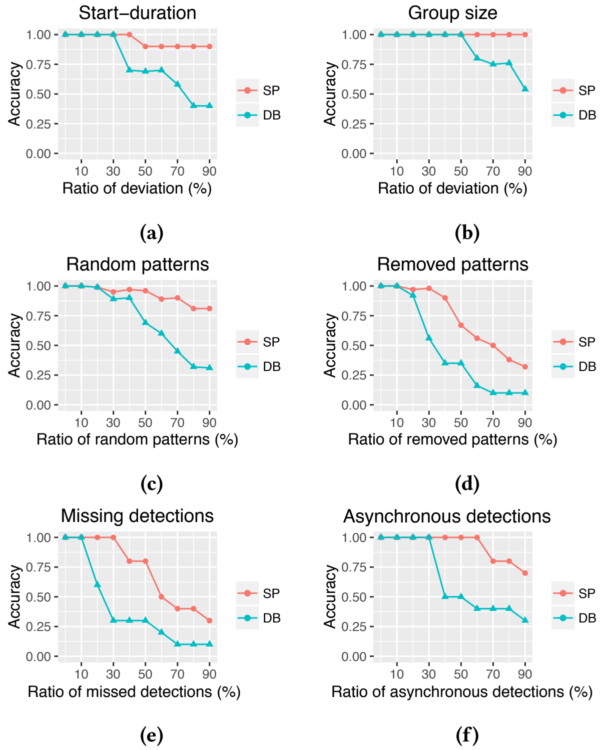
Mobility related sensitivity parameters

- **Variable start and duration**: We modify the start and duration of each presence pattern by $t^*_{start} \in N(t_{start}, \tau\alpha_{ts})$ and $\tau^* \in N(\tau, \tau\alpha_{td})$ such that $t^*_{start} + \tau^* < FD$. $N(\mu, \sigma)$ represents a normal distribution with mean μ and standard deviation σ.
- **Variable group size**: We modify the set of mobile entity IDs of each presence pattern to $\mathbf{GS}^*$ with a new size $NG^* \in N(NG, NG\alpha_{gs})$.
- **New random patterns**: For each space, we generate βNP number of new random patterns with the same procedure that we generated the presence patterns.
- **Removal of patterns**: We randomly remove γNP number of patterns from the original patterns and create a mobility dataset.

Instrument related sensitivity parameters

- **Asynchronous detection frequency**: In reality, the frequency of detections is very much device dependent. In order to show the effect of asynchronous detections being sent by mobile entities, we randomly choose ηNG number of mobile entities from each presence pattern and change their detection period by assigning a random number in the range $[2, 0.5\tau]$.
- **Missing detections**: After creating the detection dataset $\overline{\mathbf{DT}}(\mathrm{s})$, we randomly remove ρ percent of mobile entity IDs for each moment in $\overline{\mathbf{DT}}(\mathrm{s})$. (Recall that detections occur at discrete moments in time.)

Table 2 represents the parameter ranges used for the tests in this section. The results of analysis with the synthetic dataset are shown in Figure 2. We use detections from a total of 10 different clusters. In each figure, we show the accuracy of assigning instances to the correct original cluster while varying a specific sensitivity test parameter. We note that with 10 clusters, simply assigning all instances to one cluster will lead to 10% accuracy. Therefore, an accuracy less than 10% is meaningless. In order to have a feeling of how good the accuracy of *Spaceprint* is, we compare it with a *density-based* approach. We extracted the feature vectors for *Spaceprint* using Algorithm 2 and an equivalent feature vector for the *density-based* approach with the fingerprint parameters ($FD = 1440, FR = 60$). The features extracted using these two methods are alternatively used as input to *K-means*. In the case of *Spaceprint*, the distance metric introduced in Definition 5.2 is used. For the density-based alternative we use the Euclidean distance.



(a) (b)

(c) (d)

(e) (f)

Figure 2: Tests with synthetic dataset. ***SP*** **and** ***DB*** **denote use of feature vectors extracted based on** ***Spaceprint*****, and** ***density-based*** **approach, respectively.**

Table 2: The parameters chosen for generating a synthetic dataset

Parameter	Value
NS	10
NI	100
FD	1440
FR	60
NG, NP	$\in [1, 100], \in \mathbb{N}$
t_{start}	$\in [1, 1440], \in \mathbb{N}$
τ	$\in [1, 1440 - t_{start}], \in \mathbb{N}$
$\alpha_{gs}, \alpha_{ts}, \alpha_{td}, \beta, \gamma, \eta, \rho$	$\in [0, 0.9], \in \mathbb{Q}$

The graphs presented in Figure 2 suggest that using the feature vectors extracted by *Spaceprint* results in a higher accuracy than using *density-based* feature vectors. Figures 2(a) and (b) show that the accuracy of *Spaceprint* is hardly affected by the changes in start, duration, and group size of random patterns. It is also seen in Figure 2(c) that introducing new random patterns will not degrade the accuracy of *Spaceprint* as the fixed underlying patterns are being reflected in various elements of the feature vector. By removing patterns that construct the original space from a generated instance of that space, the accuracy of *Spaceprint* degrades. However, *Spaceprint* is still much more robust in response to such changes than the *density-based* approach (Figure 2(d)). We see in Figures 2(e) and (f) that *Spaceprint* is also more robust to the noise introduced by instrument-related parameters than the density-based approach. Although missing detections and variable frequency of detections will distort parts of the feature vector representing presence patterns with a finer period, the effect of patterns will still be present in elements which represent coarser sampling periods.

6.2 Real datasets

In this section, we apply our fingerprinting framework on two datasets collected from real-world public spaces. Both of these datasets conform to our model in Section 3. However, due to having different data collection mechanisms, they have subtle differences in terms of sparsity of detections and variety of spaces (summarized in Table 3). The first dataset is a set of WiFi detections very rich in terms of the number of detections collected per space but contains data from a limited number of spaces. This dataset is collected by WiFi scanners placed in all **coffee corners** at our university campus for a period of 5 months[1]. The second one, which is a dataset of Foursquare [19] check-ins, is very rich in terms of diversity of spaces while being much sparser in terms of the number of detections available per location. We chose locations within the top 100 location categories with data from more than 531 days.

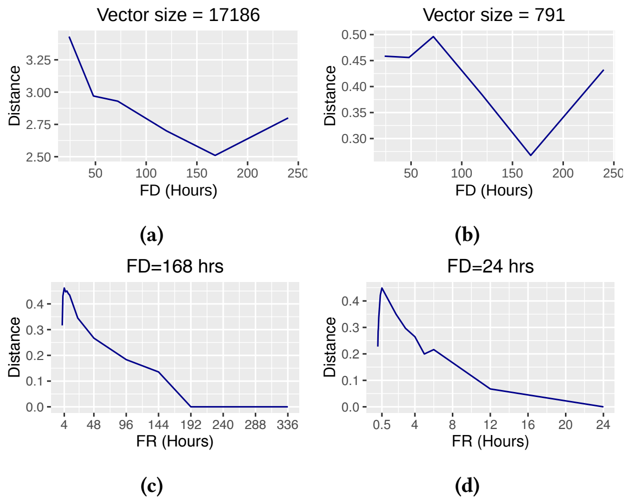
Table 3: Datasets

	WiFi DB	Foursquare DB
#Spaces	8	10,000
#Categories	1	100
#Mobile entities	700,000	201,132
Duration (days)	150	531
#Detections	95,000,000	24,474,738
#Detections per space per day	**79,166**	**2.3**

6.3 Case study with WiFi dataset

In what follows, we demonstrate the procedure of extracting fingerprinting parameters and feature vectors using the WiFi dataset.

6.3.1 Extracting fingerprint parameters. In order to calculate the feature vectors, it is required that the optimal fingerprinting parameters, *FD* and *FR*, are extracted for each space separately. We show how we find these values for one of the coffee corners using Algorithm 3. Figures 3(a) and (b) illustrate how the optimal fingerprint duration can be extracted. What is shown in these graphs is the average pairwise distance of feature vectors calculated by varying the parameter, *FD*. It should be noted that the comparison of fingerprint durations is only fair if it is performed based on the pairwise distance of vectors of the same length (vectors of longer length will have more elements equal to zero and therefore, their distance will be smaller). To have feature vectors of the same length, we changed the fingerprinting resolution, *FR*, based on the fingerprint duration such that the size of the resulting feature vector stays constant. This is achieved by setting $\frac{FD}{FR}$ to a constant value. We calculated these distances for vectors of length 17186 and 791 features, respectively. The optimal fingerprint duration is the one that *minimizes* the distance between two feature vectors, and thus maximizing similarity. For both graphs shown in Figure 3(a) and (b), this value is acquired at a duration equivalent to one week (168 hours). In Figure 3(b), it is also seen that a fingerprint duration equivalent to 3 days is the worst fingerprinting choice, leading to maximum dissimilarity between vectors.



(a) **(b)**

(c) **(d)**

Figure 3: (a-b) Choosing optimal fingerprinting duration (c-d) choosing optimal fingerprinting resolution

Figures 3(c) and (d) show how the optimal fingerprint resolution can be chosen. The optimal fingerprint resolution is the one that *maximizes* the distance between feature vectors revealing more detail about the space. We have looked at the optimal resolution when the fingerprint duration is equal to the optimal fingerprint duration (1 week time). The results suggest that a resolution of 4 hours can still reveal the differences between feature vectors. As most of the weekdays are similar, we also looked at the spaces (only over weekdays) with a fingerprinting duration of 24 hours. The figures suggest that a resolution of 30 minutes suffices to reveal the necessary level of detail when the fingerprint is only extracted from weekdays. This is in fact the minimal resolution that still captures detections from static devices. Any finer resolution will result only in more zero-valued entries in feature vectors. Note that deriving two optimal resolutions does not contradict Lemma 5.3, as the daily resolution is extracted only from weekdays.

[1] Anonymous WiFi scanning can be performed by hashing MAC addresses on the fly and providing an opt-out option for visitors.

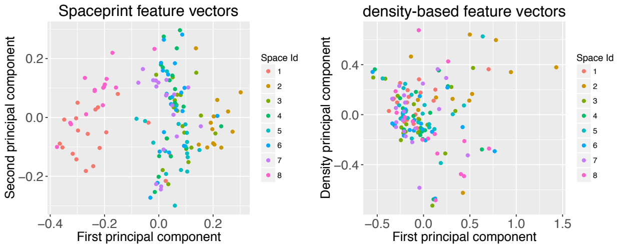
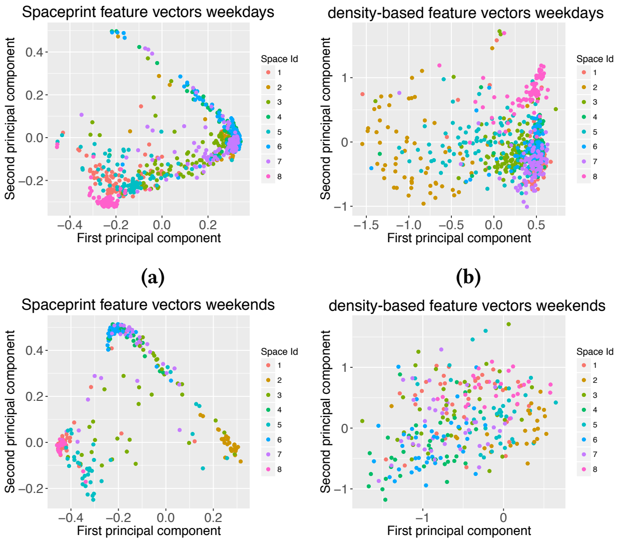


Figure 4: Two-dimensional representation of feature vectors of *Spaceprint* and *density-based* approach. Each point represents one week of data. $FD = 168$ **hours and** $FR = 1$ **hour.**



(a) **(b)**

(c) **(d)**

Figure 5: Two-dimensional representation of feature vectors of *Spaceprint* and *density-based* approach. Each point represents one day of data. $FD = 24$ **hours and** $FR = 1$ **hour.**

6.3.2 Two-dimensional representation of feature vectors: To further see how *Spaceprint* represents the similarities between the situation in these coffee corners, we also visualize the extracted feature vectors from the whole dataset in Figures 4 and 5. The feature vectors extracted have n elements (e.g., with $FD = 168$ hours and $FR = 1$ hour, $n = 23355$) and can be represented as points in an n-dimensional coordinate system. In order to represent such points, we map them to a two-dimensional space using multi-dimensional scaling [2]. This method takes a dissimilarity matrix composed of the pair-wise distance between all vectors. By applying principal component analysis on such a matrix a coordinate matrix is generated whose configuration minimizes a loss function. Using the dissimilarity matrix calculated based on the distance function from Definition 5.2, multi-dimensional scaling can capture the effects of the nonuniform size of the elements in our feature vectors.

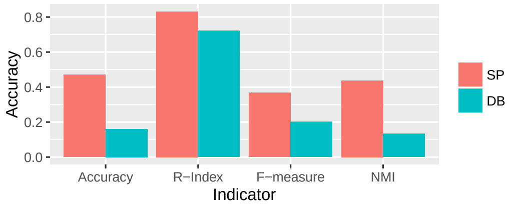


Figure 6: Performance of clustering for $FD = 168$ **hours.** SP **and** DB **denote use of *Spaceprint* and *density-based* features.**

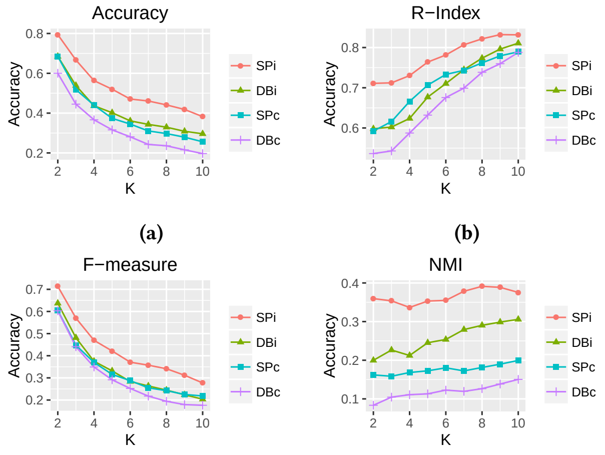
The results are presented in Figures 4 and 5. We compare the result of vectorizing using *Spaceprint* and the *density-based* approach. In Figure 4, we have vectorized each week of data ($FD = 168$ hours and $FR = 1$ hour). As seen, *Spaceprint* results in a clearer distinction between points of the same color. In other words, the identity of the location is reflected in the similarity between weeks of data from the same space. In Figure 5, using the parameters $FD = 24$ hours and $FR = 1$ hour, each day is vectorized separately. We also present the weekdays and weekends in separate graphs. Again, *Spaceprint* provides a better distinction between the situation in spaces by placing points representing days in different spaces further from each other. This is specifically visible in the case of weekends (Figure 5(c)-(d)). The data presented here includes occasional changes in normal presence patterns, due to holidays, special events such as conferences, etc. Therefore, there are naturally outliers, yet the identity of locations is evident.

6.3.3 Clustering performance (Latent variable of identity). To further evaluate how such feature vectors can be used to create a unique fingerprint for spaces, we cluster them using *K-means* algorithm. The goal is to see if we can distinguish from *which* space they have been extracted. Each space in this dataset has a **space id**. We cluster feature vectors extracted from 150 days and look for 8 different clusters representing 8 different space ids. This is equivalent of assigning points of the same color (in Figure 5) to the same cluster. Performance of the clustering task in terms of Accuracy, Random Index, F-measure, and Normalized Mutual Information (NMI) is presented in Figure 6. As seen, the results are in favor of *Spaceprint* for all of these indicators.

6.4 Case study with the Foursquare dataset

In this section, we perform evaluations on a dataset collected from Foursquare location-based social network. Each space in this dataset has a **space id** and a **space category**. Taking each of these two labels as ground truth for determining the clustering performance, gives us the opportunity to perform two types of evaluations. The first one, similar to evaluations on the WiFi dataset, is to classify feature vectors to know from *which* space they were extracted. The second one, is to classify feature vectors of a group of spaces to know from *what type* of space they were collected. Performance is evaluated based on classification of spaces with category labels such as *home*, *office*, *airport*, *restaurant*, *Chinese restaurant*, *road*, etc. (Full list is omitted due to lack of space).

6.4.1 Clustering performance (Latent variable of identity). For the first experiment, performance of *Spaceprint* and *density-based* method (SP_i and DB_i) is compared based on classification between K randomly chosen space ids ($K \in [2, 10]$) and feature vectors extracted from 531 days. The accuracy of clustering algorithm is



(a)

(b)

(c)

(d)

Figure 7: Tests with Foursquare dataset. ***SP*** **and** ***DB*** **denote use of feature vectors extracted based on** ***Spaceprint*****, and** ***density-based*** **approach, respectively. Subscripts "i" and "c" refer to classification based on the latent variable of identity and category, respectively.** ***K*** **is the number of clusters.**

calculated on correctly clustering feature vectors of different spaces based on their original space id.

6.4.2 Clustering performance (Latent variable of category). For the second experiment, we chose K randomly chosen categories ($K \in [2, 10]$) and further selected 10 spaces per category. We similarly extracted the feature vectors from 531 days. The accuracy of *Spaceprint* and *density-based* method (SP_c and DB_c), is compared based on correctly clustering the feature vectors of a group of spaces based on their correct category. The results presented in Figure 7, are the mean value acquired from 100 runs of experiment per K with $FD = 168$ and $FR = 1$ hour. Generally, regardless of the high sparsity level of this dataset, comparisons shown in Figure 7 (a)-(d) are in favor of *Spaceprint* for both experiments. Higher performance in terms of NMI shows that even misclassification of spaces based on category yields more information about the similarity of spaces in different clusters. An example will be misclassifying a space with the category label of *restaurant* to the category of *Chinese restaurant*.

7 DISCUSSION AND CONCLUSIONS

In this paper, we presented *Spaceprint*, a technique for creating spatial fingerprints for repetitive situations in public spaces. What makes *Spaceprint* unique is its fully automatic operation with minimal input from anyone who operates it. Our evaluations show that the automated fingerprinting of spaces is indeed possible, opening the path to more sophisticated approaches for automated situation-awareness. We also conclude that *Spaceprint* is relatively insensitive to parameters that can degrade the classification accuracy. By automatically extracting fingerprint parameters, *Spaceprint* allows embedding privacy by design in data collection by anonymizing (e.g. hashing) data with timely hashes based on fingerprint duration parameter such that the accuracy of the spatial fingerprint is also not affected. In this paper, we looked at the possibility of fingerprinting repetitive situations in a single space. Our future work entails refining this method to consider interaction between multiple spaces in creating these fingerprints.

REFERENCES

[1] M. Baratchi, N. Meratnia, P. J. M. Havinga, A. Skidmore, and A.G. Toxopeus. 2013. Sensing solutions for collecting spatio-temporal data for wildlife monitoring applications: a review. *Sensors* 13, 5 (2013), 6054–6088.

[2] I. Borg and P. J. F. Groenen. 2005. *Modern multidimensional scaling: Theory and applications.* Springer Science & Business Media.

[3] T. M. T. Do and D. Gatica-Perez. 2014. The Places of Our Lives: Visiting Patterns and Automatic Labeling from Longitudinal Smartphone Data. *IEEE Transactions on Mobile Computing* 13, 3 (2014), 638–648.

[4] N. Eagle and A. S. Pentland. 2009. Eigenbehaviors: identifying structure in routine. *Behavioral Ecology and Sociobiology* 63, 7 (2009), 1057–1066.

[5] S. M. Emran and N. Ye. 2002. Robustness of Chi-square and Canberra distance metrics for computer intrusion detection. *Quality and Reliability Engineering International* 18, 1 (2002), 19–28.

[6] D. Falcone, C. Mascolo, C. Comito, D. Talia, and J. Crowcroft. 2014. What is this place? Inferring place categories through user patterns identification in geo-tagged tweets. In *6th International Conference on Mobile Computing, Applications and Services.* 10–19.

[7] H. Gao, J. Tang, and H. Liu. 2012. Mobile location prediction in spatio-temporal context. In *Nokia mobile data challenge workshop*, Vol. 41. 44.

[8] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. 2007. Trajectory Pattern Mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07).* ACM, New York, NY, USA, 330–339. DOI: http://dx.doi.org/10.1145/1281192.1281230

[9] Z. Li, J. Wang, and J. Han. 2012. Mining Event Periodicity from Incomplete Observations. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12).* 444–452.

[10] D. Lian, X. Xie, V. W. Zheng, N. J. Yuan, F. Zhang, and E. Chen. 2015. CEPR: A Collaborative Exploration and Periodically Returning Model for Location Prediction. *ACM Trans. Intell. Syst. Technol.* 6, 1, Article 8 (April 2015), 27 pages.

[11] M. Lv, L. Chen, Z. Xu, Y. Li, and G.i Chen. 2016. The discovery of personally semantic places based on trajectory data mining. *Neurocomputing* 173 (2016), 1142–1153.

[12] R. Montoliu, J. Blom, and D. Gatica-Perez. 2013. Discovering places of interest in everyday life from smartphone data. In *Proceedings of the Multimedia Tools and Applications.* 179–207.

[13] S. Spaccapietra, C. Parent, M. L. Damiani, J. A. de Macedo, F. Porto, and C. Vangenot. 2008. A Conceptual View on Trajectories. *Data Knowl. Eng.* 65, 1 (April 2008), 126–146.

[14] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A. Barabasi. 2011. Human Mobility, Social Ties, and Link Prediction. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11).* 1100–1108.

[15] Y. Wang, N. J. Yuan, D. Lian, Linli X., X. Xie, E. Chen, and Y. Rui. 2015. Regularity and Conformity: Location Prediction Using Heterogeneous Mobility Data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15).* 1275–1284.

[16] F. Wu, H. Wang, Z. Li, W. C. Lee, and Z. Huang. 2015. SemMobi: A Semantic Annotation System for Mobility Data. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15 Companion).* 255–258.

[17] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. 2013. Semantic Trajectories: Mobility Data Computation and Annotation. *ACM Trans. Int. Syst. Tech.* 4, 3, Article 49 (July 2013), 38 pages.

[18] D. Yang, B. Li, and P. Cudré-Mauroux. 2016. POIsketch: Semantic Place Labeling over User Activity Streams. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16).*

[19] D. Yang, D. Zhang, Z. Yu, and Z. Yu. 2013. Fine-grained preference-aware location search leveraging crowdsourced digital footprints from LBSNs. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing, (Ubicomp '13).* 479–488.

[20] N. J. Yuan, Y. Zheng, X. Xie, Y. Wang, K. Zheng, and H. Xiong. 2015. Discovering Urban Functional Zones Using Latent Activity Trajectories. *IEEE Trans. Knowl. and Data Eng.* 27, 3 (2015), 712–725.