Spaceprint: a Mobility-based Fingerprinting Scheme for 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 situational awareness in a specific location can be achieved by characterizing the fingerprint of recurrent situations from ubiquitously generated mobility data. Without semantic input about the time and space (location) where situations take place, this turns out to be a fundamental challenging problem. Uncertainties in data 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 actual identity of a space can be discovered from the extracted situation patterns.

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. 2017. *Spaceprint*: a Mobility-based Fingerprinting Scheme for Spaces. In *Proceedings of SIGSPATIAL'17, Los Angeles Area, CA, USA, November 7–10, 2017,* 4 pages.
DOI: 10.1145/3139958.3140009

1 INTRODUCTION

Full automation of decision making in smart cities of the future requires decision-support systems with the capability of describing the meaning of situations in different types of locations. A means for learning the patterns of such 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. In this paper, we investigate how mobility data can represent the general repetitive pattern of situations in spaces.

What we refer to as a situation is a series of events that happen in a space within a duration of time. Oftentimes, a specific space with a known category such as a library, or a classroom, will exhibit repetitive visiting patterns. Such patterns operate as a *fingerprint* of situations in that space that periodically repeat in the same manner. Moreover, we can expect that places with the same category 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 category. 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.

Implementing a situation-awareness system requires algorithms that are able to perform on raw data which is mixed with uncertainties and are generic enough to capture the significance of dynamic situations happening in any type of space. Previous related research in extracting the meaning of spaces do not provide the necessary capability needed for implementing such a system. Relevant research normally focus on automatic place labeling by considering that a space has a static meaning (or a known label such as "home", "office", etc.) [1–5]. This goal is achieved by extensive preprocessing and selecting a set of intuitive features (e.g. number of visitors, presence at nights or a specific day of the week) to classify spaces with certain categories.

We address the question to what extent we can automatically *measure* a location's unique fingerprint of situations purely using available mobility data. To realize situation-aware systems that are generally applicable, we focus on creating these fingerprints in a completely automated manner. This implies that these fingerprints should be created from raw mobility data without additional human input of any kind such as those spent on feature-engineering, parameter setting, and data cleaning. 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 (*Spaceprint*s) in a fully automated manner. (3) We validate our method by showing its classification performance using a WiFi-based detection dataset.

2 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. 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. 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 previously mentioned mobility-data collection systems can result in such a dataset.

SIGSPATIAL'17, Los Angeles Area, CA, USA
 978-1-4503-5490-5/17/11…$15.00
DOI: 10.1145/3139958.3140009

Given a set of detection records **DT**, we are interested in the **space 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 within the same category have similar fingerprints, and (3) spaces within different categories have different fingerprints.

3 FRAMEWORK OVERVIEW

Our goal is to define a fingerprint that summarizes the situations in a space in terms of repeating presence patterns over a duration of time. The challenge in our case is to identify (1) the **features**, (2) an appropriate **resolution** and **duration window**, and (3) a suitable **distinguishing classifier**. The fingerprint for spaces, therefore, is defined as:

Definition 3.1. (Space fingerprint) The ***fingerprint for the space*** s *is a triplet* $\mathbf{SP}(s) = \langle \mathbf{V}, FD, FR \rangle$*, 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.* $\exists r \in \mathbb{N} : FD = r \cdot FR$.

As seen, such a fingerprint is composed of three components which are a feature vector **V** and two fingeprint parameters *FD* and *FR*. We have a separate procedure for extracting each of these components as explained in the following sections.

4 METHODOLOGY

4.1 Presence patterns

To select features from mobility data one intuitively may think of static features such as opening or closing hours, peak hours, group sizes, number of individuals, etc. Without any intuitive assumptions about features that define the situation in a space, the only measurable feature from detections is related to the presence pattern of mobile entities. 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. Extracting the distinguishing group presence 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 4.1. A ***presence feature*** $PF(t_{\text{start}}, \tau, T_s)$ *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* 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. The feature vector **V** can be considered as a list of normalized presence features. The range of the parameters mentioned before can be determined 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., $T_s \leq \tau$.

4.2 Feature vector

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 of different presence features. Algorithm 1 (*vectorize*) represents the way of constructing a feature vector for a given space based on a collection of mobile entity detections. 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. 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 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 (lines 2-4). In each iteration, by counting the mobile entities that appeared in the intersection of all buckets of $[\overline{\mathbf{DT}}[W]]_{T_s}$ (line 7), a presence feature is created and appended to the feature list (line 8).

Algorithm 1: vectorize

```
Data: DT(s), FD, FR
Result: V
V = [];
for (t_start = 0; t_start < FD; t_start = t_start + FR) do
    /* iterate over all durations                                  */
    for (τ = FR; τ ≤ FD − t_start; τ = τ + FR) do
        for (T_s = FR; T_s ≤ τ; T_s = T_s + FR) do
            ;                /* iterate over all sampling periods */
            if (τ mod T_s = 0) then
                W = ⟨t_start, τ⟩;
                u = ⋂([DT[W]]_{T_s}) ;   /* get the ID of mobile
                 entities present in all buckets of window W */
                append(V, count(u)) ; /* append to V the total
                 number of mobile entities */
return(V/ max(V));
```

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 *Canberra Distance*.

Definition 4.2. 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* $\Delta(\mathbf{V}, \mathbf{V}^*) = \frac{1}{n}\sum_{i=1}^{n}\frac{|v_i - v_i^*|}{|v_i| + |v_i^*|}$

4.3 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 situations. 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 **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 1, 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, we are looking for 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). Algorithm 2 summarizes the procedure of extracting the fingerprinting parameters.

Algorithm 2: fingerprintParameters

```
Data: DT(s), r (such that FD = r · FR)
Result: FD,FR
for (i = 1; i < τ(DT)/(2r); i = i + 1) do
    m = i · r;
    for (j = 0; j < τ(DT)/m; j = j + 1) do
        DT_j = {⟨d, s, t⟩ ∈ DT(s) | j · m ≤ t < (j + 1) · m};
        V^i_j = vectorize(DT_j, m, i);
FD = r · arg min_i Σ_{j,k} Δ(V^i_j, V^i_k); ;        /* Optimal duration */
for (i = 1; i ≤ FD; i = i + 1) do
    if (FD mod i = 0) then
        for (j = 0; j < τ(DT)/FD; j = j + 1) do
            DT_j = {⟨d, s, t⟩ ∈ DT(s) | j · FD ≤ t < (j + 1) · FD};
            V^i_j = vectorize(DT_j, FD, i);
FR = arg max_i Σ_{j,k} Δ(V^i_j, V^i_k); ;        /* Optimal resolution */
return(FD,FR)
```

5 EVALUATION

We evaluate the performance of Spaceprint by comparing it with with a baseline. We show how *Spaceprint* feature vectors can be used for finding repetitive situation patterns in spaces using ia WiFi dataset. To the best of our knowledge, there is no prior work in classifying or creating situation fingerprints for spaces purely based on presence patterns. As a baseline, we consider a **density-based** approach that calculates hourly densities [4]. A *density-based* feature vector $\mathbf{V}_d = [d_0, \ldots, d_{\frac{FD}{FR}-1}]$ is extracted where each element d_i represents the number of mobile entities appearing in the window $W = \langle i \cdot FR, FR \rangle$.

5.1 Test on a WiFi dataset

In this section, we apply our fingerprinting framework on a dataset of WiFi detections. This dataset is collected from 8 different coffee corners for a period of 150 days. There are 95 million detections generated by more than seven hundred thousand devices.

5.1.1 Extracting fingerprint parameters. Figures 1(a) and (b) illustrate how the optimal fingerprint duration can be extracted. The average pairwise distance of feature vectors calculated by varying the parameter, *FD* is shown. The comparison of fingerprint durations is fair only if it is performed based on the pairwise distance of vectors of the same length. To have feature vectors of the same length, we changed the fingerprinting resolution, *FR*, (17186 and 791) based on the fingerprint duration such that the size of the resulting feature vector stays constant. For both graphs shown in Figure 1(a) and (b), the optimal fingerprint duration (minimum) is acquired at a duration equivalent to one week (168 hours). Figures 1(c) and (d) show how the optimal fingerprint resolution can

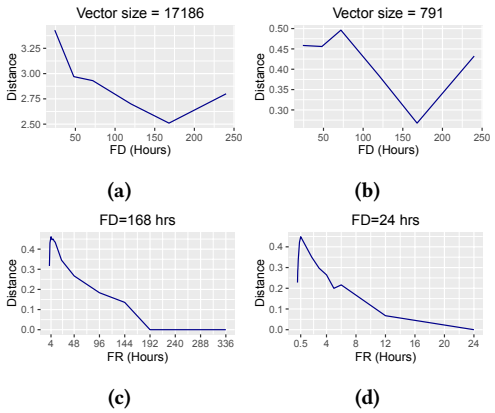

Figure 1: Choosing the optimal fingerprinting (a-b) duration (c-d) resolution

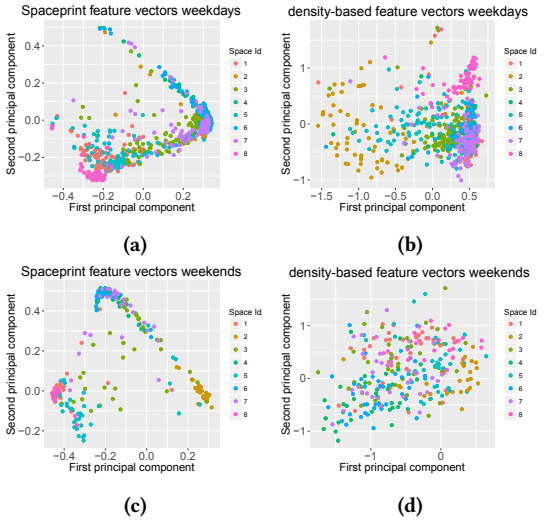

Figure 2: Two-dimensional representation of feature vectors of *Spaceprint* and *density-based* approach. Each point represents one day of data. $FD = 24$ hours and $FR = 1$ hour.

Figure 3: Performance of clustering for $FD = 168$ hours. *SP*, and *DB* denote use of *Spaceprint* and *density-based* features.

be chosen. The optimal fingerprint resolution is the one that *maximizes* the distance between feature vectors. We have looked at the optimal resolution when the fingerprint duration is equal to the optimal fingerprint duration (1 week time). The results in Figure 1(c) 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. Figure 1(d) suggests that a resolution of 30 minutes suffices to reveal the necessary level of detail when the fingerprint is only extracted from weekdays.

5.1.2 Two-dimensional representation of feature vectors: The feature vectors extracted have n elements that 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. In Figure 2, we compare the result of vectorizing using *Spaceprint* and the *density-based* approach. Using the parameters $FD = 24$ hours and $FR = 1$ hour, each day is vectorized separately and represented as a point on the image. We also present the weekdays and weekends in separate graphs. 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 days of data from the same space. *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 2(c)-(d)).

5.1.3 Classification accuracy: To evaluate how feature vectors can 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 2) 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 3. As seen, the results are in favor of *Spaceprint* for all of these indicators.

6 CONCLUSIONS

In this paper, we presented *Spaceprint*, a technique for creating 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 possible, opening the path to more sophisticated approaches for automated situational awareness. Our future work entails extending such fingerprints to situations spanning over multiple spaces.

REFERENCES

[1] Trinh Minh Tri Do and Daniel Gatica-Perez. 2014. The places of our lives: Visiting patterns and automatic labeling from longitudinal smartphone data. *IEEE Trans. on Mobile Computing* 13, 3 (2014), 638–648.
[2] 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.
[3] 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.
[4] 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)*.
[5] Mao et al. Ye. 2011. On the semantic annotation of places in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 520–528.