ABSTRACT

Many applications track the movements of mobile users, especially in controlled (e.g., indoor) environments. This information can be combined with other data (e.g., user transaction records) for effective promotion marketing or item recommendation. In this paper, we define the concept of geo-footprint, a concise representation of the visits by mobile users in supervised indoor spaces, which summarizes the potential interest of users in nearby points of interest. Then, we define similarity measures between users based on their footprints, inspired by popular models from Information Retrieval. Finally, we propose and evaluate similarity search algorithms which can be used as modules in recommender systems or data mining tasks (e.g., clustering or nearest-neighbor classifiers).

1 INTRODUCTION

There has been a lot of previous work on managing the locations and movement of mobile objects. Most of it focuses on handling (outdoor) user trajectories [23]. This includes trajectory data analytics systems [9, 18], trajectory cleaning and transformation [5], trajectory similarity measures [20], trajectory retrieval [17, 25], and trajectory clustering [24] and classification [2]. Targeted applications include traffic monitoring and analysis [26], navigation [11], trip recommendation [16]. Location-based services for indoor spaces are gaining popularity [19], as indoor location tracking can be achieved with good accuracy, either with the help of wireless technology (e.g., WiFi, RFID, UWB, BLT devices, etc.) [27], computer vision [13], or by using the phone’s inertial sensors [14].

This work focuses on the capturing and use, for each mobile user, of the locations or regions that have been of interest to the user, such as a visit to the TVs section of a department store. Instead of storing the entire history of user locations (i.e., the complete trajectories), we focus on location information, which concisely captures and implies the interests or habits of users within a given context (e.g., in a department store). We call this information geo-footprint.

Example The manager of department store Acme is interested in tracking, for each registered customer and for each visit of the customer to Acme, the areas within the store where the customer spent at least 1 min. These regions may relate the customer to products or product categories that are exhibited/promoted nearby.

Motivation By comparing the geo-footprints of two users, we can infer whether two users have similar interests, behavior, or habits. Computing the geo-footprints of users and their similarity finds application in market analysis and recommender systems. For instance, customer segmentation divides the customers of a company into clusters based on the similarity of their features (characteristics). Geo-footprints can be considered as features. A clustering algorithm can be used to divide customers into groups based on the areas of the store they habitually visit. Targeted advertisements and promotions can then be addressed to each group, based on the exhibited products or services in the frequently visited areas by the group members. Geo-footprints can be used together with other features (e.g., age, transaction records) to define clusters based on multivariate information. Similarly, in recommender systems or advertisement, geo-footprints can be used as part of the user profiles. This is especially useful in situations where there is insufficient information about other feature types of the target user (i.e., cold-start users). Given a target user \( u \), to whom a recommendation should be provided, the recommender system can identify users with the most similar geo-footprints as \( u \) and recommend the products bought by them to \( u \). Another application is link recommendation in geo-social networks. The profiles of users of such networks include their location visits and their frequencies, which can be modeled as geo-footprints. In particular, frequently visited nearby places by a user \( u \) can be modeled as regions of interest in the geo-footprint of \( u \). Geo-footprint similarity can then be used to model the probability that two users meet and become socially connected.

Contributions In this paper, we define the concept of geo-footprint, which finds application in market-analysis and recommendation. We propose a measure for the similarity between geo-footprints, which naturally extends document similarity measures from information retrieval. We propose algorithms for efficient footprint-based similarity computation, which build upon plane-sweep and spatial join techniques. Finally, we investigate the indexing of geo-footprints for efficient similarity search.

2 RELATED WORK

There has been ample work on tracking, managing, and analyzing mobility data [2, 5, 9, 13, 14, 16–20, 23–27], as discussed in the introduction. Related to our work is the problem of identifying hot spots of moving vehicles [15]. Sample objects are used as “sensors” that track the density of vehicles around them, based
on their speed. Time-parameterized hot spots and regions are identified. As opposed to our geo-footprints, these hot spots are not linked to individual users and cannot be used as user features.

Related to our geo-footprints are the locations associated to documents or objects in location-based information retrieval [6, 7, 21]. The places mentioned in each document are extracted and indexed together with the text content, to facilitate location-based keyword search. (i.e., find documents containing biword "Chinese restaurant" near my location). The differences between these regions/locations and our geo-footprints is that, for a single object (or document), the same location cannot be in the object’s profile multiple times. Hence, our definitions and similarity function are essentially different.

Finally, geo-footprints are related to dwell or stay regions, where moving objects spend significant time [8, 12, 22, 28, 29]. Such regions are points or areas which remain close to the object for (at least) a time period [22, 28] and/or where the object exhibits stable (slow) movement [29]. Their identification can be done by geometric on-line or off-line algorithms (e.g., clustering [10]. The regions of interest (RoI) that we define in Sec. 3.1 are similar, but not identical, to stay/dwell regions. Specifically, for practical purposes, our regions are rectangular and enclose the locations relevant to them; this facilitates their efficient extraction, as we discuss in Sec. 3.2. Note that our definition of footprints (as sets of regions) and their similarity is orthogonal to how their constituent regions are defined.

3 GEO-FOOTPRINTS

This section presents definitions and an approach for extracting geo-footprints from user trajectories. We consider the scenario that the locations of users (e.g., customers in a department store) are tracked automatically and regularly and converted to trajectories, linked to the user identifiers. Formally:

**Definition 3.1 (user trajectory).** A user trajectory $T$ of length $|T|$ is a sequence of locations $\{l_1, l_2, \ldots, l_{|T|}\}$, such that each location $l_i = (x, y, t)$ is characterized by a spatial position $p$ (i.e., a pair of x, y coordinates) and a timestamp $t$. For regularly tracked locations, $l_{i+1}.t - l_i.t$ equals a fixed time difference $\Delta t$. for each $i \in [1, |T|]$.

For the same user $u$, we typically have a sequence of trajectories $u.T = (u.T_1, u.T_2, \ldots, u.T_y)$, where for each $i \in [1, |T|]$, $u.T_i \subset u.T_{i+1}$. t, i.e., the trajectories are **temporally disjoint**. We may also refer to a trajectory in $u.T$ as a session of user $u$. For example, a trajectory/session in $u.T$ corresponds to a continuous time period during which customer $u$ visited a store.

3.1 Regions of interest

Instead of the entire trajectories, we are interested in sub-trajectories, where the user is relatively immobile (e.g., the customer wanders around or stands near specific exhibition items). Two or more consecutive locations in a user trajectory belong to the same region of interest, if they are spatially close to each other. Formally:

**Definition 3.2 (region of interest (RoI)).** A region of interest for a given user $u$ is defined by the 3D minimum bounding box (MBB) that encloses the set of consecutive regions $R = \{l_i, l_{i+1}, \ldots, l_k\}$ in a trajectory $u.T$ of $u.T$, where $1 \leq s < e \leq |T|$, such that (i) $|l_i.p - l_j.p| \leq \epsilon$ for each $i, j \in [s, e]$ and (ii) $e - s > \tau$.

Parameter $\epsilon$ is a spatial extent constraint and $\tau$ is the minimum duration of a subtrajectory to qualify for a RoI. Quantity $|l_i.p - l_j.p|$ denotes the spatial distance between locations $l_i.p$ and $l_j.p$; typically measured using Euclidean distance ($L_2$). Intuitively, a RoI $R$ includes all consecutive user locations in a trajectory and their timestamps, the extent of $R$ does not exceed $\epsilon$, and $R$ includes a large enough number of locations, corresponding to a large enough time interval to indicate the user’s high interest to the region. For example, for a customer in a store who spends a long time near a particular category of items, we can infer that the spatial region around the items characterizes the interest of the customer. Figure 1(a) shows two regions of interest that can be extracted from a user trajectory $u.T$. For simplicity, we use the same symbol $R$ to denote the sequence of locations that define a region and their MBB.

3.2 Geo-footprint Extraction

To minimize the number of extracted RoIs from a user trajectory and at the same time unify temporally consecutive, overlapping RoIs, we propose Algorithm 1, which computes RoIs that are temporally maximal and temporally disjoint. Algorithm 1 is a greedy heuristic, which starts by verifying the conditions of Definition 3.2 on the first $\tau$ locations of the examined trajectory $T$ (i.e., $s = 1, e = \tau$). If the first $\tau$ locations form a RoI, we keep $s$ constant and expand $e$ until the maximality condition is violated and finalize the extracted maximal region. The next candidate to start searching for a maximal region is $[e + 1, e + \tau + 1]$. If the first $\tau$ locations do not form a region of interest, we repeat with $s = 2, e = \tau + 1$, and so on, until the condition is satisfied for a given $s$ and $e$. Algorithm 1 also includes an optimization in order to avoid some redundant checks. Starting from the first location in the input trajectory $T$, the algorithm adds to the current region $R$ the next location $l_i$, until condition $e$ is violated. When this happens, if $R$ has enough locations, it is added to the profile of the user corresponding to trajectory $R$ and a new $R$ is initialized to hold $l_i$ (Lines 6-8). If $R$ does not have enough locations, then a new region $newR$ is initialized with just $l_i$; then, we add to $newR$ as many points from $R$ as possible starting from the last point of $R$ and going backwards. This guarantees that the maximal region which includes $l_i$ will not be missed, while avoiding redundant operations.

We run Algorithm 1 for all trajectories of a user $u$, to extract all RoIs of the user. If the user has been at the same area multiple times (e.g., the TV exhibition area of a store), then the weight of that should be higher in the user’s profile. Based on this, we define the geo-footprint $F(u)$ of a user as the collection of all RoIs of $u$, disregarding their temporal dimension. In other words, we consider the RoIs in a user footprint to be 2D MBRs, i.e., the 2D projections of the extracted 3D RoIs:

**Definition 3.3 (Geo-footprint).** The geo-footprint $F(u)$ of a user $u$ is the spatial projections of the RoIs that are extracted from the trajectories of $u$ using Algorithm 1; i.e., the RoIs that define $F(u)$
Algorithm 1 Regions of interest extraction

Require: trajectory T, bounds ε, τ
1. \( R := \text{null} \) \quad \text{Current region}
2. for each \( i \in T \) do
3. \( \text{if } R \cup \{i\} \text{ does not violate } \varepsilon \text{ then} \)
4. \( \text{add } i \text{ to } R \)
5. \( \text{else} \)
6. \( \text{if } |R| \geq \varepsilon \text{ then} \)
7. \( \text{add } R \text{ to user profile} \)
8. \( R := \{ i \} \) \quad \text{Initialize current region}
9. \( \text{else} \)
10. \( \text{newRegion := \{ } \)
11. \( \text{while newRegion } \cup R \text{ last does not violate } \varepsilon \text{ do} \)
12. \( \text{newRegion } \cup R \text{ last; delete } R \text{ last;} \)
13. \( \text{end while} \)
14. \( R := \text{newRegion} \) \quad \text{Initialize current region}
15. \( \text{end if} \)
16. \( \text{end if} \)
17. \( \text{end for} \)
18. \( \text{if } |R| \geq \varepsilon \text{ then} \)
19. \( \text{add } R \text{ to user profile} \)
20. \( \text{end if} \)

should be temporally disjoint and each of them cannot be enriched with additional locations without violating the constraint \( \varepsilon \) in Def. 3.2.

For example, in recommender systems for department stores, the time when a customer visited an area in the store may not be important. Examples of such user footprints are shown in Figure 1(b).

4 SIMILARITY BETWEEN GEO-FOOTPRINTS

To define the similarity between users based on their geo-footprints, we model the preference of a user \( u \) to a location \( l \) by the number of times \( l \) is in the RoIs of \( F(u) \), i.e., \( u \)'s geo-footprint. This is similar to the relevance of a text document to a term defined by the number of times the term appears in the document. However, since the spatial domain is continuous and infinite, as opposed to the domain of possible terms, we use the set of RoIs (instead of individual locations) in the user footprints to define and measure similarity.

Specifically, the footprint of a user can also be modeled as a set of disjoint spatial regions and their frequencies. Figure 2(a) shows an example of a footprint with three rectangular RoIs \( r_1, r_2, r_3 \) extracted by Algorithm 1. The RoIs divide the space into disjoint regions \( (A \text{ to } I) \) denoted by different colors. These disjoint regions are not necessarily rectangular and their union is the space covered by all three RoIs \( r_1, r_2, r_3 \). For each of the disjoint regions, Figure 2(a) shows in parentheses its frequency, i.e., the number of times it is included in the RoIs \( r_1, r_2, r_3 \) of the footprint.

Hence, the footprint \( F(r) \) of a user \( r \) can also be modeled as a set of \((X, f_X)\) pairs, where \( X \) is a continuous spatial region (not necessarily rectangular) and \( f_X \) is the frequency of \( X \). For example, the frequency of region \( C \) is 2 because it is included in \( r_1 \) and \( r_2 \), but not in \( r_3 \).

We define the similarity between two users \( r \) and \( s \), based on their footprints \( F(r) \) and \( F(s) \), inspired by the popular cosine similarity in IR, as follows:

\[
\text{sim}(F(r), F(s)) = \sum_{(X, f_X) \in F(r), (Y, f_Y) \in F(s)} |X \cap Y| \cdot f_X \cdot f_Y / (|F(r)| \cdot |F(s)|)
\]

(1)

The numerator in Eq. 1 aggregates the common locations in the footprints \( F(r) \) and \( F(s) \) and multiplies them with their frequencies in the footprints. That is, for each pair \((X, Y)\) of regions that intersect and \( X \) is in \( F(r) \), \( Y \) is in \( F(s) \), the area \(|X \cap Y|\) of their intersection \( X \cap Y \) is computed and multiplied by \( f_X \) and \( f_Y \); the result is added to the numerator. The numerator is equivalent to the dot product of two frequency vectors, which include all locations in space and their frequencies in the footprints of users \( r \) and \( s \). Similarly, the denominator is the product of two quantities \(|F(r)|\) and \(|F(s)|\) which are equivalent to the Euclidean norms of the footprints, considering all locations in space. Specifically:

\[
|F(r)| = \sqrt{\sum_{(X, f_X) \in F(r)} |X| \cdot f_X^2},
\]

(2)

where \( |X| \) denotes the area of region \( X \). The denominator of Eq. 1 ensures that \( \text{sim}(F(r), F(s)) \) ranges from 0 to 1; two identical footprints have a similarity equal to 1 and two entirely disjoint ones have zero similarity. Figure 2(b) shows an example of two geo-footprints from two users \( r \) and \( s \). Footprint \( F(r) \) originally has two overlapping regions \( r_1 \) and \( r_2 \) which are modeled by three disjoint regions \( r_A, r_B, r_C \). Their frequencies shown in parentheses, i.e., \( r_A = r_1 - r_2, r_B = r_1 \cap r_2, r_C = r_2 - r_1 \). Hence, \( F(r) = \{(r_A, 1), (r_B, 2), (r_C, 1)\} \). Similarly, \( F(s) = \{(s_A, 1), (s_B, 2), (s_C, 1)\} \). The footprint similarity between \( r \) and \( s \) is:

\[
|r_A \cap s_A| \cdot 1 + |r_B \cap s_B| \cdot 1 + |r_C \cap s_C| \cdot 2 - |r_B \cap s_B| \cdot 2 - |r_B \cap s_C| \cdot 1 \geq \sqrt{2} \cdot 1^2 + 2 \cdot 2 \cdot 1^2 + 2 \cdot 2 \cdot 1^2 + 2 \cdot 2 \cdot 1^2 + 2 \cdot 2 \cdot 1^2 + 2 \cdot 2 \cdot 1^2.
\]

which amounts to \( \sqrt{\frac{2}{\sqrt{7}}} \approx 0.228 \). The numerator sums up the frequency products for all pairs of overlapping regions.

Figure 2: Disjoint regions and frequencies

5 SIMILARITY COMPUTATION

Given the footprints \( F(r) \) and \( F(s) \) of two users \( r \) and \( s \), we investigate how we can efficiently compute their similarity. We first study the efficient computation of the norm \(|F(r)|\) of a footprint \( F(r) \) and then suggest methods for similarity computation.

5.1 Norm computation

Algorithm 2 is a plane-sweep algorithm that computes the norm \(|F(r)|\) of a footprint \( F(r) \). Algorithm 2 can be used in a preprocessing phase, where the norms of all user footprints are computed and stored, so that they can readily be used in Eq. 1 whenever we need to compute the similarity between two footprints.

Algorithm 2 computes from the set of rectangular RoIs \( r_1, r_2, \ldots \) of \( F(r) \) a set of disjoint regions and their frequencies, i.e., the \((X, f_X)\) pairs discussed in Sec. 4. While doing so, it incrementally constructs the norm by summing the contribution of each such region \( X \). Initially, we pick a sorting dimension (e.g., the x-axis) and take the projection of each RoI \( r_i \) in \( F(r) \) on that dimension, which is an interval \([a, b]\). We create two triples \((a, r_i, \text{id}, \text{Start})\) and \((b, r_i, \text{id}, \text{End})\). All triples are then sorted using the first column. The algorithm then accesses the triples in order, which model the stops of a plane-sweep line. We initialize a data structure \( D \), which manages the disjoint regions at each stop of the
Algorithm 2 Norm computation algorithm

Require: set of RoIs (footprint) \( F(r) \)
1: Sort endpoints \((i, r, id, type)\) of RoIs projections on the x-axis
2: \( D \leftarrow \{(0, 0)\}; sqs \leftarrow \{0\}; \) \( \text{pre} \leftarrow \infty \)
3: for each \((i, r, id, type)\) in \( \text{in-order} \) do
4: for each entry \( e \in D \) in \( \text{in-order} \) do
5: \( sqs \leftarrow sqs + (e.\text{next}\_\text{start} - e.\text{start} \cdot (e - \text{pre} \cdot e.\text{count})^2 \) \( \text{update norm square} \)
6: end for
7: if \( \text{type} = \text{Start} \) then \( \text{add entries to D} \)
8: \( e.\text{e} \in D \) with largest \( \text{start} \), such that \( e.\text{start} \leq r.\text{ylow} \)
9: \( e.\text{e} \leftarrow (r.\text{ylow}, e.\text{count} + 1); D.\text{add}(e) \)
10: while \( e.\text{next}\_\text{start} < r.\text{ylow} \) do
11: \( e.\text{next}\_\text{count} \rightarrow e.\text{next}\_\text{count} + 1 \)
12: \( e.\text{e} \) next
13: end while
14: \( D.\text{add}(r.\text{ylow}, e.\text{count} - 1); \) \( \text{type} = \text{End}; \text{remove entries from D} \)
15: else
16: \( e.\text{e} \in D \) with \( \text{start} = r.\text{ylow} \)
17: \( D.\text{remove}(e) \)
18: while \( e.\text{next}\_\text{start} < r.\text{ylow} \) do
19: \( e.\text{next}\_\text{count} \rightarrow e.\text{next}\_\text{count} - 1 \)
20: \( e.\text{e} \) next
21: end while
22: \( D.\text{remove}(e.\text{next}) \) \( \text{entry for r.\text{ylow}} \)
23: end if
24: \( \text{pre} \leftarrow e.\text{count} \)
25: end for
26: return \( \sqrt{\text{sqs}} \)

Complexity Analysis If there are \( n \) RoIs in \( F(r) \), the algorithm needs \( 2n \) steps. Each step scans \( D \) to update \( sqs \); \( D \) contains at most \( 2n \) entries. In addition, each step, updates \( D \) to add or remove 2 entries and these changes require a scan of \( D \) in the worst case. Hence, the time complexity is \( O(n^2) \). Regarding space, the algorithm has to maintain \( D \), which is \( O(n) \).

Extraction of Disjoint Regions Algorithm 2 can directly be used to extract a set of disjoint regions and their frequencies, which can be used as an alternative (to the set of RoIs) for representing a footprint. Specifically, when we update \( sqs \) (line 5), we can output each of the regions that contribute to \( sqs \) together with its frequency (equal to \( e.\text{count} \)). This will give us a set of disjoint rectangular regions that can be used to model the footprint.

5.2 Similarity computation
Algorithm 3 is a variant of Algorithm 2 that computes the similarity between two footprints \( F(r) \) and \( F(s) \). Besides them, Algorithm 3 takes as input their norms (assumed to have been pre-computed by Algorithm 2). Algorithm 3 sorts the endpoints of all RoI projections on a selected dimension (e.g., the x-axis) and accesses them in order, simulating a plane sweep line that stops at each endpoint. At each stop of the line it maintains in two data structures \( D_i \) and \( D_j \) the active intervals and their counts from \( F(r) \) and \( F(s) \). The main difference to Algorithm 2 is lines 5-17, where a routine merges the (ordered) contents of \( D_i \) and \( D_j \) to compute the contribution of a stripe (i.e., the area between the current line position \( v \) and the previous one \( \text{pre} \) ) to the numerator \( \text{sim} \) of the similarity function (Eq. 1). This routine computes the weighted-intersection between the disjoint regions for \( F(r) \) and \( F(s) \) in the stripe. After the merge-joint routine, \( D_r \) or \( D_s \) is updated depending on the source of the current endpoint, indicated by flag \( src \) in the representation of endpoints. After completing all steps, the algorithm divides \( \text{sim} \) by the product of the two norms and returns the similarity.

Algorithm 3 Similarity computation algorithm

Require: sets of RoIs (footprints) \( F(r) \), \( F(s) \); \( \text{norm}r, \text{norm}s \)
1: Sort endpoints \((i, r, id, src, type)\) of RoI projections on the x-axis
2: \( D_r \leftarrow \{(0, 0); D_s \leftarrow \{(0, 0)\}; \) \( \text{pre} \leftarrow \infty \)
3: \( \text{pre} \leftarrow \infty \text{min-value}; \text{sim} \leftarrow 0 \)
4: for each \((i, r, id, src, type)\) in \( \text{in-order} \) do
5: \( e_1 \leftarrow \text{first entry in } D_r \)
6: \( e_2 \leftarrow \text{second entry in } D_s \)
7: while \( e_1 \neq \text{null} \) and \( e_2 \neq \text{null} \) do
8: if \( e_1.\text{start} < e_2.\text{start} \) then
9: up \leftarrow \text{min}(e_1.\text{next}\_\text{start}, e_2.\text{start})
10: \text{sim} \leftarrow \text{sim} + (up - e_1.\text{start}) \cdot (e_2 - \text{pre} \cdot e_2.\text{count} \cdot e_2.\text{pre} \cdot \text{count} \cdot e_2.\text{count})
11: \( e_1 \leftarrow e_2.\text{next} \)
12: else
13: up \leftarrow \text{min}(e_1.\text{next}\_\text{start}, e_2.\text{start})
14: \text{sim} \leftarrow \text{sim} + (up - e_1.\text{start}) \cdot (e_2 - \text{pre} \cdot e_2.\text{count} \cdot e_2.\text{pre} \cdot \text{count} \cdot e_2.\text{count})
15: \( e_1 \leftarrow e_1.\text{next} \)
16: end if
17: \( \text{end while} \)
18: if \( src = 0 \) then \( \text{endpoint from an } F(s) \) RoI
19: update \( D_s \) by running lines 7-23 of Alg. 2, for \( D = D_s \)
20: else
21: update \( D_r \) by running lines 7-23 of Alg. 2, for \( D = D_r \)
22: end if
23: \( \text{pre} \leftarrow \text{null} \)
24: end for
25: return \( \text{sim}/(\text{norm}r \times \text{norm}s) \)

Complexity Analysis If there are \( n \) RoIs \( F(r) \) and \( m \) RoIs \( F(s) \), the algorithm needs \( 2(n + m) \) steps. Each step performs a concurrent scan to \( D_r \) and \( D_s \) to compute their contribution to \( \text{sim} \); \( D_r \) and \( D_s \) may include up to \( 2(n + m) \) entries. The update of either \( D_r \) or \( D_s \) takes at most \( O(n) \) and \( O(m) \), respectively. Hence, the overall time complexity is \( O((n + m)^2) \). The space complexity is \( O(n + m) \) as we only have to maintain the state of \( D_r \) and \( D_s \) at each step.

Computing Norms and Similarity Simultaneously With a minor modification, Algorithm 3 can also compute the norms of \( F(r) \) and \( F(s) \), in case they have not been precomputed. For this, we have to apply lines 4-6 of Algorithm 2 for \( D_r \) or \( D_s \) before the set is updated and also keep track of the previous steps in \( D_r \) and \( D_s \) (in place of \( \text{pre} \) of Algorithm 2), to update \( \text{norm}r \) or \( \text{norm}s \) at each step. Overall, (modified) Algorithm 3 is a general method that can compute the similarity between two sets of RoIs, regardless of whether their norms have been precomputed or not.
5.3 Computation based on spatial join

Interestingly, we can employ any spatial intersection join algorithm \cite{1, 3} to compute $sim(F(r), F(s))$, provided that the norms $||F(r)||$ and $||F(s)||$ are available. Each pair $(r_i, s_j)$ of RoIs, $r_i \in F(r), s_j \in F(s)$ that intersect contribute to the numerator $sim$ of Eq. 1 as much as the area of their intersection. Algorithm 4 describes a similarity computation algorithm based on this. Note that, unlike Algorithm 3, Algorithm 4 cannot be extended to compute the norms $||F(r)||$ and $||F(s)||$, so, it cannot compute similarity if the norms are not given.

**Algorithm 4** Join-based similarity computation

```
Require: sets of RoIs (footprints) $F(r), F(s)$, normer, norms
1: simm ← 0
2: for each pair $(r_i, s_j), r_i \in F(r), s_j \in F(s)$ that intersect do
3: simm ← simm + $|r_i \cap s_j|$ = add intersection area of pair
4: end for
5: return simm/normer * norms
```

Correctness Proof (sketch) According to Eq. 1, the intersecting area $|X \cap Y|$ of each pair of disjoint regions $X$ of $F(r)$ and $Y$ of $F(s)$ contributes $|X \cap Y| \cdot f_X \cdot f_Y$ to the similarity. Since $X$ is part of $f_X$ original RoIs in $F(r)$ and $Y$ is part of $f_Y$ original RoIs in $F(s)$, the intersecting area $|X \cap Y|$ will be part of $f_X \cdot f_Y$ spatial join pairs found by Algorithm 4, which totally contribute $|X \cap Y| \cdot f_X \cdot f_Y$ to $sim$.

Complexity Analysis Algorithm 4 is expected to be faster than plane-sweep-based spatial intersection join has a cost of $O(n \log n) + O(m \log m) + O(n + m + K)$, where $K$ is the size of the join output \cite{3}. For each join pair, we only have to compute the intersection area which takes $O(1)$ per pair.

6 INDEXING AND SIMILARITY SEARCH

Given the footprint of a query user $q$, we investigate methods for finding the $k$ users with the highest footprint-based similarity to $q$.

6.1 Using an R-tree

An intuitive approach is to index the rectangular RoIs in the geo-footprints of all users by an R-tree; each index entry is a (RoI, ID) pair, where ID identifies the user whose footprint contains RoI.

6.1.1 Iterative search. The first (baseline) algorithm searches the R-tree for each RoI $q_j$ of the query user $q$, to find users $r$ who have regions in their footprints $F(r)$ that overlap with $q_j$. For each such region $r_i$, the algorithm updates the similarity of $r$ w.r.t. $q$ by adding to the numerator of $sim(F(r), F(q))$ the area of the spatial intersection $r_i \cap q_j$. The denominator of $sim(F(r), F(q))$ can be computed once, given that we have access to $||F(r)||$ and that $||F(q)||$ is computed in the beginning of query processing, using Algorithm 2. While finding users whose footprints overlap with $q_j$, we keep track of the set of $k$ most similar users to $q$ so far. After finishing this iterative search (i.e., one spatial search for each $q_j \in F(q)$), the $k$ most similar users become the query result.

6.1.2 Batch search. An improved approach is to perform search for all $q_j \in F(q)$ simultaneously. Specifically, we access the R-tree in search for the R-tree leaf nodes that have non-zero spatial overlap with $F(q)$. For this, we use the MBR of $F(q)$ to guide search. Whenever we visit a leaf node $L$ of the tree, we conduct a spatial join between $F(q)$ and the contents of $L$. For each pair $(r_i, q_j)$ of intersecting RoIs, where $r_i \in L$ and $q_j \in F(q)$, we update $sim(F(r), F(q))$. The $L \bowtie F(q)$ join is optimized as follows \cite{3}. Before the join, we access all contents of $L$ and remove from consideration all $r_i \in L$ which do not intersect with MBR($F(q)$). In addition, we ignore all $q_j \in F(q)$ which do not intersect MBR($L$). Then, we do a plane-sweep join between the non-eliminated RoIs in $L$ and $F(q)$. We expect this batch search approach to be significantly faster than the iterative baseline because it avoids accessing the same R-tree nodes more than once per query.

6.2 User-centric R-tree

Another way to organize the data is to index by an R-tree, for each user $r$, the MBR of $F(r)$ and the ID of $r$, which can be used to access $F(r)$. That is, the RoIs in a user footprint $F(r)$ are not stored/indexed independently, but as a single entry. We denote this user-centric R-tree index by $R^U$. We use the tree to find the user footprint MBRs that intersect the query footprint $F(q)$. For each such footprint $F(r)$, in a refinement step, we use Alg. 4 to compute $sim(F(r), F(q))$. $R^U$ indexes the regions of each user together and computes the similarity of each accessed user in one step, instead of accumulating it during search (as in Sec. 6.1); hence, it has an advantage if the RoIs in each geo-footprint are close to each other.

7 EXPERIMENTS

In this section we present our experimental analysis by first describing our setup and then presenting our experiments, which evaluate the efficiency and effectiveness of footprint extraction, norm computation, similarity computation, and similarity search.

Setup We compiled all codes in g++ 9.4.0 with flag -O3 and ran experiments on a 32GB Ubuntu 20.04.3 LTS machine with Intel Core i9-10900K CPU @3.70GHz.

Datasets. We used with the publicly available ATC shopping center dataset \cite{4}, which includes user trajectories which are exported from raw sensor measurements. The dataset is divided in different parts, each of which corresponds to a 8-10 day period of continuous recordings of user movements. The spatial coordinates of trajectory points were normalized to take values in $[0,1]$. In our experiments, we used four of the available parts as different datasets, denoted by Part A, B, C, D, respectively; Table 1 summarizes their statistics.

<table>
<thead>
<tr>
<th>Table 1: Statistics of data and extracted RoIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATC Shopping Center</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Part A</td>
</tr>
<tr>
<td>Part B</td>
</tr>
<tr>
<td>Part C</td>
</tr>
<tr>
<td>Part D</td>
</tr>
</tbody>
</table>

Footprint extraction. We implemented the footprint extraction algorithm we proposed in Sec. 3. We tuned the value of $e$ to 0.02 and the value of $\tau$ to 30 which correspond to 2 meters and 3 sec., respectively. For this, we experimented with different values and used the ones that led to a reasonable number of RoIs for each user. Table 1 summarizes, for each dataset, the average number of RoIs per user footprint and the average extent of the RoIs in each dimension. As Table 2 shows, Algorithm 1 is fast and scalable, extracting about 4.5K footprints per second.

Norm computation. After extracting the footprints for each of the four parts, we applied the norm computation algorithm (Alg.
2) on each part, to compute the norms of all footprints in it, which are to be used in similarity computations. The computation of norms for all parts (also reported in Table 2) is very efficient.

**Similarity computation.** Next, we investigated the performance of the user similarity algorithms we proposed in Sections 5.2 and 5.3. For each part, we picked 200 random user footprints from it and computed the similarity between these users and all other users in the part. Then, we took the average cost of each similarity computation using Alg. 3 and Alg. 4. Table 3 shows the results. On average, Alg. 4 is 1-2 orders of magnitude faster than Alg. 3, because it aggregates the similarity from each pair of intersecting RoIs instead of doing so by first decomposing them into disjoint regions. Hence, Alg. 4 is the best option, provided that the norms have been pre-computed and they are available, which is consistent to our complexity analysis.

**Similarity search.** Next, we compare the three indexing and search approaches suggested in Sec. 6; namely, (i) using an R-tree and performing iterative search for each region in the query user footprint (Sec. 6.1.1) or (ii) performing batch search (Sec. 6.1.2), or (iii) using a user-centric R-tree, which indexes footprint MBRs (Sec. 6.2). Figure 3(a) shows the total runtime for 1000 random top-K similarity queries (sampled from the data) on each part, for $K = 5$ (time is not affected by $K$). The user-centric R-tree approach typically outperforms the other methods, because it uses a smaller tree, which is fast to search and construct (as shown in Table 4). However, we observed that for queries having very large MBRs, the user-centric approach can become slower than the other methods, because it computes $sim(F(r), F(q))$ for numerous users $r$, although their RoIs do not actually overlap with the RoIs of $q$.

**Utility.** To assess the utility of our geo-footprints, we applied agglomerative clustering to the footprints of a sample of 4000 users from Part A, using the average-link approach for merging clusters. For each of the nine produced clusters, Figure 3(b) shows in different colors, one for each cluster, characteristic regions on the map which were in the footprints of users of one cluster and not in the footprints of users in other clusters. This experiment indicates that our geo-footprints can be used to identify effectively groups of users that visit different areas; this information can be used for marketing purposes, as discussed in the introduction.

### 8 EXTENDABILITY

In this work, we have considered 3D RoIs and 2D spatial projections of them; still, our work can directly be applied for the case where objects move in the 3D space. In this case, the original RoIs are 4D (i.e., 3 dimensions for space and one for time) and the footprints comprise the 3D spatial projections of RoIs. Our similarity measure algorithms and search techniques are still applicable; however, the time complexity of Algorithms 2 and 3 increases to $O(n^3)$, as the “active intervals” at each stop of the sweep line (now sweep plane) become rectangles and we need to find their intersections by a spatial join algorithm that has quadratic cost in the worst case.

In addition, note that by keeping only the spatial projections of RoIs in each footprint, as per Def. 3.3, we neglect the temporal durations of extracted RoIs. However, in some applications it may make sense to use the duration of visits at each region as an importance weight. For example, if user $u_1$ spends double the time at the TVs section compared to user $u_2$ on a particular day, this might mean that $u_1$ is more interested in TVs than $u_2$. Our approach can easily consider duration of each region in the footprint as a weight. That is, the 2D regions of interest (RoIs) that a user footprint includes can be weighted based on the duration of the user’s stay there. Similarity and norm computation can be easily adapted to accommodate duration weights: (i) each frequency $f_X$ (or $f_Y$) in Eq. 1 and Eq. 2 is replaced by the sum of duration weights in the individual visits, (ii) in Alg. 4, the intersection of two regions is weighted by the product of their weights in the corresponding footprints, and (iii) the weights can be included in the spatial index (R-tree) together with the MBRs of the RoIs to facilitate top-k similarity retrieval.

### 9 CONCLUSIONS

In this work, we defined the concept of geo-footprints, which comprise the spatial regions in indoor spaces whereabout users stay for long time periods and model the user interests to nearby items. In addition, we defined the similarity between geo-footprints of different users which can be used in location-based market analysis and recommendations (e.g., in e-commerce). We presented efficient algorithms for similarity computation and for similarity-based search, based on geo-footprints. In the future, we plan to investigate the enrichment of footprints with temporal information (i.e., duration) and the adaptation of similarity definition and computation, accordingly. In addition, we plan to investigate the effectiveness of geo-footprints when used as features by recommenders.

### ACKNOWLEDGMENTS

Funded by the Hellenic Foundation for Research and Innovation (HFRl) under the “2nd Call for HFRl Research Projects to support Faculty Members & Researchers” (Project No. 2757) and by the Greek Research Technology Development and Innovation Action “RESEARCH - CREATE - INNOVATE”, Operational Programme on Competitiveness, Entrepreneurship and Innovation 2014–2020, Grant T1EDK-04810.