Similarity Search based on Geo-footprints

Achilleas Michalopoulos
Department of Computer Science
and Engineering
University of Ioannina, Greece
amichalopoulos@cse.uoi.gr

Konstantinos Lampropoulos
Department of Computer Science
and Engineering
University of Ioannina, Greece
klampropoulos@cse.uoi.gr

George Kelantonakis
University of Crete and FORTH-ICS
Heraklion, Greece
kelantonag@ics.forth.gr

Chrysostomos Zeginis
University of Crete and FORTH-ICS
Heraklion, Greece
zegchris@ics.forth.gr

Kostas Magoutis
University of Crete and FORTH-ICS
Heraklion, Greece
magoutis@ics.forth.gr

Nikos Mamoulis
Department of Computer Science
and Engineering
University of Ioannina, Greece
nikos@cse.uoi.gr

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 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

Published in Proceedings of the 27th International Conference on Extending Database Technology (EDBT), 25th March-28th March, 2024, ISBN 978-3-89318-095-0 on OpenProceedings.org.

10.48786/edbt.2024.52

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 $l_i = \langle l.p, l.t \rangle$ 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 Δt, for each $i \in [1, |T|)$.

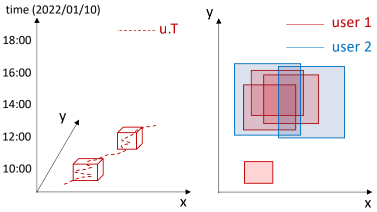
For the same user u, we typically have a sequence of trajectories $u.\mathcal{T} = \{u.T_1, u.T_2, \ldots u.T_{\mathcal{T}}\}$, where for each $i \in [1, |\mathcal{T}|)$, $u.T_i.l_{|T_i|}.t < u.T_{i+1}.l_1.t$, i.e., the trajectories are *temporally disjoint*. We may also refer to a trajectory in $u.\mathcal{T}$ as a *session* of user u. For example, a trajectory/session in $u.\mathcal{T}$ 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 locations $R = \{l_s, l_{s+1}, \ldots, l_e\}$ in a trajectory $u.T$ of $u.\mathcal{T}$, where $1 \le s < e \le |T|$, such that (i) $|l_i.p - l_j.p| \le \epsilon$ for each $i, j \in [s, e]$ and (ii) $e - s > \tau$.

Parameter ϵ is a spatial extent constraint and τ 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 ϵ, 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.



(a) 3D regions (b) 2D footprints

Figure 1: Extracted regions and user footprints

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 τ locations of the examined trajectory T (i.e., $s = 1, e = \tau$). If the first τ 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 τ 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 ϵ 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 u 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 $\epsilon$, $\tau$
$R \leftarrow$ null                                   ▷ Current region
for each $l_i \in T$ do
    if $R \cup l_i.p$ does not violate $\epsilon$ then
        add $l_i$ to $R$
    else
        if $|R| \geq \tau$ then                        ▷ Current region has enough points
            add $R$ to user profile
            $R \leftarrow \{l_i\}$                     ▷ Initialize current region
        else
            $newR \leftarrow \{l_i\}$                  ▷ Initialize new region
            while $newR \cup R.last$ does not violate $\epsilon$ do
                $newR \cup R.last$; delete $R.last$;
            end while
            $R \leftarrow newR$                        ▷ Initialize current region
        end if
    end if
end for
if $|R| \geq \tau$ then                                ▷ Last region has enough points
    add $R$ to user profile
end if
```

should be temporally disjoint and each of them cannot be enriched with additional locations without violating the constraint ϵ 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 nine *disjoint* regions (A 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:

$$sim(F(r), F(s)) = \frac{\sum_{(X,f_X)\in F(r),(Y,f_Y)\in F(s)} |X \cap Y| \cdot f_X \cdot f_Y}{||F(r)|| \cdot ||F(s)||} \qquad (1)$$

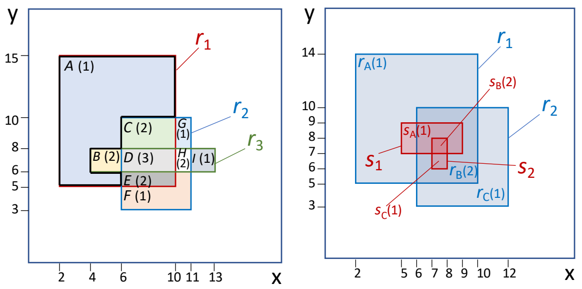
The numerator in Eq. 1 aggregates the common locations in the footprints $F(r)$ and $F(s)$ and multiplies them with their frequencies in the footprints. That is, for each pair (X, Y) of regions that intersect and X is in $F(r)$, Y is in $F(s)$, the area $|X \cap Y|$ of their intersection $X \cap Y$ is 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:

$$||F(r)|| = \sqrt{\sum_{(X,f_X)\in F(r)} |X| * f_X^2}, \qquad (2)$$

where $|X|$ denotes the area of region X. The denominator of Eq. 1 ensures that $sim(F(r), F(s))$ 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 with their frequencies shown in parentheses, i.e., $r_A = r_1 - r_2$, $r_B = r_1 \cap r_2$, $r_A = r_2 - r_1$. 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:

$$\frac{|r_A \cap s_A| \cdot 1 \cdot 1 + |r_B \cap s_A| \cdot 2 \cdot 1 + |r_B \cap s_B| \cdot 2 \cdot 2 + |r_B \cap s_C| \cdot 2 \cdot 1}{\sqrt{52 \cdot 1^2 + 20 \cdot 2^2 + 22 \cdot 1^2} \cdot \sqrt{7 \cdot 1^2 + 1 \cdot 2^2 + 1 \cdot 1^2}}$$

which amounts to $2/\sqrt{77} \approx 0.228$. The numerator sums up the frequency products for all pairs of overlapping regions.



(a) Disjoint-regions footprint (b) Regions of two footprints

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 $\langle a, r_i.id, Start\rangle$ and $\langle b, r_i.id, End\rangle$. 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 sweep line. D divides the line into intervals; each interval corresponds to a continuous empty space or occupied space by a disjoint region. D keeps a set of $(start, count)$ pairs ordered by $start$. For the first entry in D, $start$ is the smallest value of the non-sweep axis (e.g., y-axis). The $start$ value of each subsequent entry in D defines the end value of the interval of the previous entry. When the sweep line moves from one position to the next one, D is used to compute the areas of the disjoint regions and their contribution to the norm (lines 4-6); we keep in a variable ssq the sum of squares computed as the line progresses. We also keep in variable $prev$ the previous position of the sweep line (to compute the areas of the regions between the previous position of the line to the next one).

When the line moves from value $prev$ to v (i.e., the currently accessed projection endpoint is $\langle v, r_i.id, type\rangle$), the first thing to do is to compute the contribution of the disjoint regions in the stripe from x=$prev$ to x=v (lines 4-6). For this, we scan the entries of D in ascending $start$ order and, for each $e \in D$, compute the area of the region defined by y-range $[e.start, e.next.start]$ and x-range $[prev, v]$ and multiply them with $e.count^2$. The result is added to nqo. $e.next$ is the next entry to e in D (if there is no $e.next$, then $e.next.start = \infty$ and $e.count$ should be 0).

After updating ssq, the algorithm updates D to include the correct intervals and their counts. Specifically, if the line stops to the beginning of a RoI r_i, two new entries are added to D, otherwise (r_i ends), two entries that correspond to r_i are removed from D, updating the corresponding counters. In the end, Algorithm 2 returns the $\sqrt{ssq}$ which is the norm of the input set of RoIs.

Algorithm 2 Norm computation algorithm

```
Require: set of RoIs (footprint) $F(r)$;
 1: Sort endpoints $\langle v, r_i.id, type\rangle$ of RoIs projections on the x-axis
 2: $D \leftarrow [(0,0)]$; $ssq \leftarrow 0$; $prev \leftarrow$ min x-value;
 3: for each $\langle v, r_i.id, type\rangle$ in $v$-order do
 4:     for each entry $e$ in $D$ in $start$-order do        ▷ update norm square
 5:         $ssq \leftarrow ssq + (e.next.start - e.start) \cdot (v - prev) \cdot e.count^2$
 6:     end for
 7:     if $type = Start$ then        ▷ add entries to $D$
 8:         $e \leftarrow e \in D$ with largest $start$, such that $e.start \leq r_i.y_{low}$
 9:         $e \leftarrow (r_i.y_{low}, e.count + 1)$; $D.add(e)$
10:         while $e.next.start < r_i.y_{up}$ do
11:             $e.next.count \leftarrow e.next.count + 1$
12:             $e \leftarrow e.next$
13:         end while
14:         $D.add((r_i.y_{up}, e.count - 1))$;
15:     else        ▷ $type = End$: remove entries from $D$
16:         $e \leftarrow e \in D$ with $start = r_i.y_{low}$
17:         $D.remove(e)$
18:         while $e.next.start < r_i.y_{up}$ do
19:             $e.next.count \leftarrow e.next.count - 1$
20:             $e \leftarrow e.next$
21:         end while
22:         $D.remove(e.next)$        ▷ entry for $r_i.y_{up}$
23:     end if
24:     $prev = v$
25: end for
26: return $\sqrt{ssq}$
```

Complexity Analysis If there are n RoIs in $F(r)$, the algorithm needs $2n$ steps. Each step scans D to update ssq and 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 ssq (line 5), we can output each of the regions that contribute to ssq together with its frequency (equal to $e.count$). 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_r and D_s 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_r and D_s to compute the contribution of a stripe (i.e., the area between the current line position v and the previous one $prev$) to the numerator $simn$ 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-join 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 stops, the algorithm divides $simn$ by the product of the two norms and returns the similarity.

Algorithm 3 Similarity computation algorithm

```
Require: sets of RoIs (footprints) $F(r), F(s)$; $normr, norms$
 1: Sort endpoints $\langle v, r_i.id, src, type\rangle$ of RoIs projections on the x-axis
 2: $D_r \leftarrow [(0,0)]$; $D_s \leftarrow [(0,0)]$
 3: $prev \leftarrow$ min x-value; $simn \leftarrow 0$
 4: for each $\langle v, r_i.id, src, type\rangle$ in $v$-order do
 5:     $e_r \leftarrow$first entry in $D_r$
 6:     $e_s \leftarrow$second entry in $D_s$
 7:     while $e_r \neq$ null and $e_s \neq$ null do
 8:         if $e_r.start < e_s.start$ then
 9:             $up \leftarrow \min\{e_r.next.start, e_s.start\}$
10:             $simn \leftarrow simn + (up - e_r.start) \cdot (v - prev) \cdot e_r.count \cdot e_s.prev.count$
11:             $e_r \leftarrow e_r.next$
12:         else        ▷ $e_r.start \geq e_s.start$
13:             $up \leftarrow \min\{e_s.next.start, e_r.start\}$
14:             $simn \leftarrow simn + (up - e_s.start) \cdot (v - prev) \cdot e_s.count \cdot e_r.prev.count$
15:             $e_s \leftarrow e_s.next$
16:         end if
17:     end while
18:     if $src = 0$ then        ▷ endpoint from an $F(r)$ RoI
19:         update $D_r$ by running lines 7-23 of Alg. 2, for $D = D_r$
20:     else        ▷ endpoint from an $F(s)$ RoI
21:         update $D_s$ by running lines 7-23 of Alg. 2, for $D = D_s$
22:     end if
23:     $prev = v$
24: end for
25: return $simn/(normr \cdot norms)$
```

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 $simn$; D_r and D_s may include up to $2(n+m)$ entries. The update of eiher 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 stops in D_r and D_s (in place of $prev$ of Algorithm 2), to update $normr$ or $norms$ 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 [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 *simn* 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) $F(r)$, $F(s)$; $normr, norms$
$simn \leftarrow 0$
for each pair $(r_i, s_j)$, $r_i \in F(r)$, $s_j \in F(s)$ that intersect do
    $simn \leftarrow simn + |r_i \cap s_j|$        ▷ add intersection area of pair
end for
return $simn/(normr \cdot norms)$
```

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 *simn*.

Complexity Analysis Algorithm 4 is expected to be faster than Algorithm 3, as 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 [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 [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 [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 1: Statistics of data and extracted RoIs

ATC Shopping Center	#users	avg. #regions	avg. relative	
			x-extent	y-extent
Part A	278K	16	0.020145	0.017232
Part B	236K	18	0.019387	0.016651
Part C	317K	20	0.019247	0.016606
Part D	377K	17	0.025416	0.022551

Footprint extraction. We implemented the footprint extraction algorithm we proposed in Sec. 3. We tuned the value of ϵ to 0.02 and the value of τ 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 resonable 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.

Table 2: Times for footprint extraction & norm computation

Preprocessing Time (s)	Part A	Part B	Part C	Part D
Footprint extraction	60.09	56.9	82.15	90.33
Norm computation	6.91	7.84	7.97	11.98

Table 3: Times for similarity computation

Avg. cost (μs)	Part A	Part B	Part C	Part D
Algorithm 3	46.24	59.5	52.7	16.39
Algorithm 4	1.08	1.28	1.46	0.53

Table 4: Indexing time for R-tree methods

Indexing Time (s)	Part A	Part B	Part C	Part D
R-tree	3.54	3.68	5.64	5.57
User-centric R-tree	0.25	0.22	0.31	0.35

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.

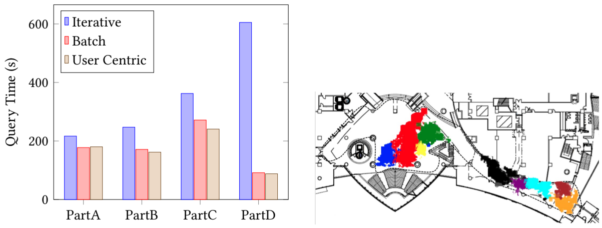
Similarity computation. Next, we investigated the performance of the user similarity algorithms we proposed in Sections 5.2 and 5.3. For each part, we picked 200 random user footprints from it and computed the similarity between these users and all other users in the part. Then, we took the average cost 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 to 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



(a) Similarity search performance

(b) Char. regions of clusters

Figure 3: Similarity search and footprint effectiveness

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 accomodate 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 foorprints, 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 (HFRI) under the "2nd Call for HFRI 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.

REFERENCES

[1] Lars Arge, Octavian Procopiuc, Sridhar Ramaswamy, Torsten Suel, and Jeffrey Scott Vitter. 1998. Scalable Sweeping-Based Spatial Join. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA.* Morgan Kaufmann, 570–581.

[2] Jiang Bian, Dayong Tian, Yuanyan Tang, and Dacheng Tao. 2019. Trajectory Data Classification: A Review. *ACM Trans. Intell. Syst. Technol.* 10, 4 (2019), 33:1–33:34. https://doi.org/10.1145/3330138

[3] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. 1993. Efficient Processing of Spatial Joins Using R-Trees. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993*, Peter Buneman and Sushil Jajodia (Eds.). ACM Press, 237–246.

[4] Drazen Brscic, Takayuki Kanda, Tetsushi Ikeda, and Takahiro Miyashita. 2013. Person Tracking in Large Public Spaces Using 3-D Range Sensors. *IEEE Trans. Hum. Mach. Syst.* 43, 6 (2013), 522–534.

[5] Maike Buchin, Anne Driemel, Marc J. van Kreveld, and Vera Sacristán. 2011. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *J. Spatial Inf. Sci.* 3, 1 (2011), 33–63. https://doi.org/10.5311/JOSIS.2011.3.66

[6] Yen-Yu Chen, Torsten Suel, and Alexander Markowetz. 2006. Efficient query processing in geographic web search engines. In *ACM SIGMOD*. ACM, 277–288.

[7] Gao Cong, Christian S. Jensen, and Dingming Wu. 2009. Efficient Retrieval of the Top-k Most Relevant Spatial Web Objects. *Proc. VLDB Endow.* 2, 1 (2009), 337–348.

[8] Maria Luisa Damiani, Hamza Issa, and Francesca Cagnacci. 2014. Extracting stay regions with uncertain boundaries from GPS trajectories: a case study in animal ecology. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas/Fort Worth, TX, USA, November 4-7, 2014.* ACM, 253–262.

[9] Xin Ding, Lu Chen, Yunjun Gao, Christian S. Jensen, and Hujun Bao. 2018. UlTraMan: A Unified Platform for Big Trajectory Data Management and Analytics. *Proc. VLDB Endow.* 11, 7 (2018), 787–799. https://doi.org/10.14778/3192965.3192970

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA.* AAAI Press, 226–231.

[11] Chenjuan Guo, Bin Yang, Jilin Hu, and Christian S. Jensen. 2018. Learning to Route with Sparse Trajectory Sets. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018.* IEEE Computer Society, 1073–1084. https://doi.org/10.1109/ICDE.2018.00100

[12] Tessai Hayama, Tsuyoshi Nariai, and Kazuya Nagatomo. 2019. Extracting Stay Regions from UWB Indoor Trajectory. In *8th International Congress on Advanced Applied Informatics, IIAI-AAI 2019, Toyama, Japan, July 7-11, 2019.* IEEE, 785–790.

[13] Niklas Karlsson, Enrico Di Bernardo, James P. Ostrowski, Luis Goncalves, Paolo Pirjanian, and Mario E. Munich. 2005. The vSLAM Algorithm for Robust Localization and Mapping. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, April 18-22, 2005, Barcelona, Spain.* IEEE, 24–29. https://doi.org/10.1109/ROBOT.2005.1570091

[14] Fan Li, Chunshui Zhao, Guanzhong Ding, Jian Gong, Chenxing Liu, and Feng Zhao. 2012. A reliable and accurate indoor localization method using phone inertial sensors. In *The 2012 ACM Conference on Ubiquitous Computing, Ubicomp '12, Pittsburgh, PA, USA, September 5-8, 2012*, Anind K. Dey, Hao-Hua Chu, and Gillian R. Hayes (Eds.). ACM, 421–430. https://doi.org/10.1145/2370216.2370280

[15] Siyuan Liu, Yunhuai Liu, Lionel M. Ni, Jianping Fan, and Minglu Li. 2010. Towards mobility-based clustering. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010.* ACM, 919–928.

[16] Xin Lu, Changhu Wang, Jiang-Ming Yang, Yanwei Pang, and Lei Zhang. 2010. Photo2Trip: generating travel routes from geo-tagged photos for trip planning. In *Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010.* ACM, 143–152. https://doi.org/10.1145/1873951.1873972

[17] Shuyao Qi, Panagiotis Bouros, Dimitris Sacharidis, and Nikos Mamoulis. 2015. Efficient Point-Based Trajectory Search. In *Advances in Spatial and Temporal Databases - 14th International Symposium, SSTD 2015, Hong Kong, China, August 26-28, 2015. Proceedings (Lecture Notes in Computer Science)*, Vol. 9239. Springer, 179–196. https://doi.org/10.1007/978-3-319-22363-6_10

[18] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. DITA: Distributed In-Memory Trajectory Analytics. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 725–740. https://doi.org/10.1145/3183713.3183743

[19] Zhou Shao, Muhammad Aamir Cheema, David Taniar, and Hua Lu. 2016. VIP-Tree: An Effective Index for Indoor Spatial Queries. *Proc. VLDB Endow.* 10, 4 (2016), 325–336. https://doi.org/10.14778/3025111.3025115

[20] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. 2020. A survey of trajectory distance measures and performance evaluation. *VLDB J.* 29, 1 (2020), 3–32. https://doi.org/10.1007/s00778-019-00574-9

[21] Artur Titkov and Panagiotis Bouros. 2022. Spatially Combined Keyword Searches. In *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022.* OpenProceedings.org, 2:403–2:407.

[22] Reaz Uddin, Mehnaz Tabassum Mahin, Payas Rajan, Chinya V. Ravishankar, and Vassilis J. Tsotras. 2023. Dwell Regions: Generalized Stay Regions for Streaming and Archival Trajectory Data. *ACM Trans. Spatial Algorithms Syst.* 9, 2 (2023), 9:1–9:35.

[23] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, and Gao Cong. 2021. A Survey on Trajectory Data Management, Analytics, and Learning. *ACM Comput. Surv.* 54, 2 (2021), 39:1–39:36. https://doi.org/10.1145/3440207

[24] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Timos Sellis, and Xiaolin Qin. 2019. Fast Large-Scale Trajectory Clustering. *Proc. VLDB Endow.* 13, 1 (2019), 29–42. https://doi.org/10.14778/3357377.3357380

[25] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Zizhe Xie, Qizhi Liu, and Xiaolin Qin. 2018. Torch: A Search Engine for Trajectory Data. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018.* ACM, 535–544. https://doi.org/10.1145/3209978.3209989

[26] Zuchao Wang, Min Lu, Xiaoru Yuan, Junping Zhang, and Huub van de Wetering. 2013. Visual Traffic Jam Analysis Based on Trajectory Data. *IEEE Trans. Vis. Comput. Graph.* 19, 12 (2013), 2159–2168. https://doi.org/10.1109/TVCG.2013.228

[27] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. 2019. A Survey of Indoor Localization Systems and Technologies. *IEEE Commun. Surv. Tutorials* 21, 3 (2019), 2568–2599. https://doi.org/10.1109/COMST.2019.2911558

[28] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009.* ACM, 791–800.

[29] Yang Zhou, Yi Chen, and Dechang Pi. 2021. Discovery of stay area in indoor trajectories of moving objects. *Expert Syst. Appl.* 170 (2021), 114501.