Multi-resolution sketches and locality sensitive hashing for fast trajectory processing

Maria Astefanoaei[†], Paul Cesaretti[‡], Panagiota Katsikouli[*], Mayank Goswami[‡], Rik Sarkar[†]

[†] School of Informatics, University of Edinburgh. {m.s.astefanoaei@sms.ed.ac.uk, rsarkar@inf.ed.ac.uk}
[‡] Queens College, City University of New York (CUNY). {paul.cesaretti@qc.cuny.edu, mayank.goswami@qc.cuny.edu}
[*] University of Lyon, Inria, INSA-Lyon, CITI. {panagiota.katsikouli@inria.fr}

ABSTRACT

Searching for similar GPS trajectories is a fundamental problem that faces challenges of large data volume and intrinsic complexity of trajectory comparison. In this paper, we present a suite of sketches for trajectory data that drastically reduce the computation costs associated with near neighbor search, distance estimation, clustering and classification, and subtrajectory detection. Apart from summarizing the dataset, our sketches have two uses. First, we obtain simple provable locality sensitive hash families for both the Hausdorff and Fréchet distance measures, useful in near neighbour queries. Second, we build a data structure called MRTS (Multi Resolution Trajectory Sketch), which contains sketches of varying degrees of detail. The MRTS is a user-friendly, compact representation of the dataset that allows to efficiently answer various other types of queries. Moreover, MRTS can be used in a dynamic setting with fast insertions of trajectories into the database.

Experiments on real data show effective locality sensitive hashing substantially improves near neighbor search time. Distances defined on the skteches show good correlation with Fréchet and Hausdorff distances.

CCS CONCEPTS

• **Theory of computation** → **Data compression**; • **Networks** → *Location based services*; • **Information systems** → Global positioning systems;

KEYWORDS

Location, distance estimation, nearest neighbours search, clustering

ACM Reference Format:
Maria Astefanoaei[†], Paul Cesaretti[‡], Panagiota Katsikouli[*], Mayank Goswami[‡], Rik Sarkar[†]. 2018. Multi-resolution sketches and locality sensitive hashing for fast trajectory processing. In *26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18), November 6–9, 2018, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3274895.3274943

SIGSPATIAL '18, November 6–9, 2018, Seattle, WA, USA

ACM ISBN 978-1-4503-5889-7/18/11…$15.00
https://doi.org/10.1145/3274895.3274943

1 INTRODUCTION

Location analytics is a fundamental aspect of insights from GPS, sensor and mobile phone data, with applications in various domains ranging from social sciences, communication networks to smart cities and transport. Effective analysis depends on fast response to basic queries, for example, finding the nearest neighbor (or all near neighbors) of a given trajectory. Ability to efficiently update the query database – e.g., inserting a new trajectory or adding new points to an existing trajectory – helps one to handle streams of trajectory data continuously updated in real time.

Trajectories are usually compared using Hausdorff distance, Fréchet distance, or the dynamic time warping (DTW) distance [26]. The last two are most popular as they incorporate the intrinsic sequential nature of the trajectories in the comparison. However, these distances are computationally challenging to apply to large datasets. Dynamic programming algorithms to compute Fréchet and DTW between two trajectories of length m require $O(m^2)$ time per comparison, which makes them prohibitive in large datasets.

Nearest neighbor searching on points in the *exact* version require either linear query time or space exponential in the dimension d, popularly referred to as the *curse of dimensionality*. Recently it was proven [3] that a query time sublinear in n for the batched Hamming nearest neighbor problem would refute the Strong Exponential Time Hypothesis (SETH). With such barriers, researchers turned to *approximate* nearest neighbor search. Formally, a c-approximate nearest neighbor algorithm will return an item whose distance to the query is at most c times the true nearest neighbor of the query item. Indyk and Motwani [19] were the first to develop Locality sensitive hashing (LSH) for this approximation problem, and it was subsequently improved upon in various works to obtain a query time sublinear in n, and polynomial dependence in d in all parameters (see for example: [6, 11, 16, 17, 25]). The main idea in LSH is to construct a hash function that ensures that similar items are usually hashed to the same buckets, while dissimilar items are hashed to different buckets.

However, unlike point datasets, where one can exploit the bounded doubling dimension of the underlying metric space [7, 8], compressing trajectory data while preserving these distances is difficult in general: the space of trajectories under Fréchet distance does not have a bounded doubling dimension and does not easily admit a low dimensional embedding [13].

LSH for trajectories. Indyk [18] describes a version of locality sensitive hashing for product spaces that apply to Fréchet distances. However, this approach has an immense space requirement, which

makes it impractical in GPS datasets. A recent work by Driemel and Silvestri [14] gives simpler mechanisms for locality sensitive hashing of trajectories. This method works by snapping the vertices of trajectories to grids, so that trajectories with Fréchet distance much smaller than the grid size are likely to snap to the same vertices. The vertex order on the grid then serves as a locality sensitive hash.

Apart from near neighbor queries, other important research problems in trajectory datasets include quickly estimating the distance between two trajectories in the dataset, clustering trajectories, or detecting whether one trajectory is similar to a sub-trajectory of another, etc. The above works do not focus on such problems. Furthermore, it is highly desirable to obtain a data structure that can support fast insertions of new trajectories into the dataset, and also one where the user can specify a desired degree of detail.

Our contributions. We devise a hierarchy of sketches for trajectories, that not only perform the basic tasks of locality sensitive hashing and near neighbor search, but can also be used to directly estimate distances between trajectories, cluster trajectories, detect subtrajectories, etc. Our sketches and hash families have the added advantage of being simpler than existing techniques.

The structure of the sketch is as follows: we deploy a random set of disks in the plane, and for each trajectory, test its intersection with these disks. Our *basic* sketch is simply a bit vector representing the intersection of the trajectory with these disks. This bit vector can act as a simple locality sensitive hash with provable probability guarantee. The main insight is that when disks are deployed randomly, similar trajectories are likely to intersect a similar set of disks. A simple measure of distance using this sketch is the Hamming distance giving the number of bits where two sketches differ. The *ordered* version of this sketch is a sequence representing the order in which the trajectory enters and exits the disks. These ideas are shown in Figure 1. Measures such as edit distance can be used here to find how similar two sketches are. In the *hierarchic* version, we deploy disks of a different size at each level of the hierarchy, with the "highest" level consisting of the largest disks.

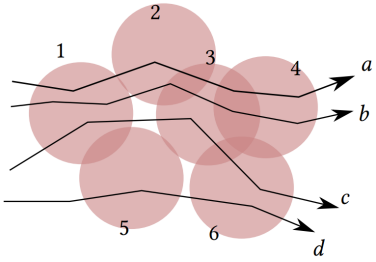


Figure 1: Trajectory skteches with respect to randomly deployed disks. Similar trajectories like a and b are likely to pass through same set/seqence of disks. For trajectory a, the basic unordered sketch is 111100**, the ordered sketch recording entry and exists is** $1; 2; 3; 2; 4; 3; 4 : : :$**.**

In real data analysis, our scales of interest, and therefore the size of the hierarchy are bounded by the application and the platform. For example, GPS localization measurements have errors up to a few meters, and thus *closeness* of GPS data is not significant at scales smaller than this size L. Similarly, beyond a certain distance, such as hundreds of kilometers, data points can simply be treated as *far*, since applications such as near neighbor search are not useful beyond such distances – let's call it this upper bound U.

We make use of these bounds to get theoretically rigorous results of practical importance. With trajectories d distance apart localized in a region of area A we can show the following:

- there is a constant factor approximation of the distance between them in time $O(\min(m; \log(U/d)\log(A/d^2)))$
- a new trajectory can be inserted into the sketch database in $O(m\log(A/L^2))$ time
- an $O(m)$ approximation to the nearest neighbor query can be computed in $O(m\log^2 n)$ time.

Experimental results. Experimental results on two real datasets show that the distance metric based on the basic unordered sketch is strongly correlated with Hausdorff distance, and the distance based on the ordered sketch is correlated with Fréchet and DTW distances. The correlations hold even when a large proportion of the data is missing, for example due to sensing/communication failure or lack of resources. The sketch-based distances can be computed orders of magnitude faster than Hausdorff, Fréchet , or DTW.

We found that a flat structure – about 50 disks of 2 km radius gives good practical results. Observe that with bit vector size of 50, Hamming distance or matching of hash can be made extremely fast. We used this feature for data pruning in nearest neighbor search. Using the sketch, we identified trajectories of same or similar sketch for comparison, and discarded everything else. Then applied Hausdorff and Fréchet distance computation on the small selected set. The sketch achieved a pruning ratio of about 80% (fraction of discarded items), while achieving accuracy of 80% – where the true nearest neighbor was found in the selected set. As a result of the high pruning ratio, on large datasets, this process runs an order of magnitude faster.

In other experiments we studied the effect of size and number of disks used. We also computed the error in distance to the nearest neighbor in the multi-resolution version of the sketch and found this to be usually low.

The rest of the paper is organized as follows. In Section 2 we present related works. In Section 3 we describe the type of queries that we are considering. In Sections 4 and 5 we describe our sketches and prove that they provide an LSH family (thus solving the near neighbor problems). In Section 6 we describe the MRTS (Multi Resolution Trajectory Sketch) data structure and show how to handle insertions, distance estimation, classification and clustering queries. Section 7 contains our detailed experimental results.

2 RELATED WORKS

Distances between curves are challenging to compute. For inputs of length m, Fréchet distance $F(\cdot)$ and DTW distance $DTW(\cdot)$ can be computed in $O(m^2)$ [5, 26]. Recent results show that assuming strong exponential time hypothesis, strongly subquadratic time algorithms are impossible [10] for Fréchet . Thus, in datasets of n trajectories. simple near neighbor search will run in $O(nm^2)$ time. Indyk's approach to LSH using product metrics [18] achieves an approximation of $O(\log m + \log\log n)$ and nearest neighbor query time of $O(m^{O(1)}\log n)$ using $O(|X|^{\sqrt{m}}(m^{\sqrt{m}}n)^2)$ storage, where X is the domain in question – loosely, the area A of the region containing the trajectories. This large data structure to be constructed

and stored, and the resultant query time polynomial in m are the main challenges of using this method in GPS datasets with large m.

The recent work by Dreimel and Silvestri [14] uses more intuitive methods to achieve locality sensitive hashing for Fréchet and DTW. The first strategy proposed is to snap trajectories to a grid placed with a random translation. The *hash* of a trajectory in this system is given by the sequence of vertices its snapped version traverses. For two trajectories P and Q, the probability of having identical hash depends on how the grid cell size δ compares to length m and the Fréchet distance $d_F(P;Q)$. In the plane, the probability of identical hash can be shown to be $1 - \frac{4md_F(P;Q)}{\delta}$. In this scheme, it is possible to construct an LSH with approximation factor linear in m for parameters suitably chosen for a pair of trajectories. A different scheme in [14] proposes to apply a fixed sequence of random perturbations to the vertices of a trajectory P before snapping them to nearest grid points. The query undergoes a different perturbation per vertex before snapping. This scheme achieves a constant approximation factor, but the probability of hashes matching for two similar trajectories can be shown only to be more than $\approx 2^{-4m}$, which is impractically small for datasets with long GPS trajectories. Since this scheme is meant mainly for answering near neighbor queries, it does not provide answers to various other types of queries one may be interested in, e.g., quickly estimating the distance between two trajectories, clustering, subtrajectory detection, etc.

Other works in managing large trajectory datasets include methods of simplification, where the goal is to construct an approximate curve that is close to the original. The Douglas-Peucker algorithm [12] is possibly the most popular algorithm of this type, which works well in many practical cases. A recent work uses topological persistence to simplify trajectories online, and find the significant turns at different resolutions [21]. Compact sketches based on topology of the domain have been developed in [15]. These methods treat trajectories individually, and do not provide any guarantees on distance computation or search in datasets. The power of large cluster computers has been exploited in [27] by developing indexing for segmented trajectories. In-network mining for popular subtrajectories is considered in [20].

In contrast to these works, our focus was to speedup trajectory processing by building highly compressed summaries. This enabled us to rapidly estimate distances between trajectories and prune large fractions of data when searching for similar trajectories.

3 PROBLEM DESCRIPTION AND BACKGROUND

We let A denote the area of the region (typically a square or a rectangle) of interest that contains all the trajectory data. Let n denote the number of trajectories in the dataset. When comparing two trajectories, we will use the term "distance between them" to mean either the Hausdorff or the Fréchet distance (defined in Section 3.1). We define a trajectory T of length m as a sequence of ordered embedded vertices $T = \{v_0; v_1; \cdots v_m\}$. We do not include other information than the order of the vertices, so essentially we treat trajectories as curves. The discrete representation simplifies the application of algorithms directly to the data without need for interpolation.

We assume an upper bound U and a lower bound L on the distances of interest. This means that we only care about the actual distance between a pair of trajectories if their distance is between L and U. For other pairs of trajectories, i.e., trajectories that are farther than U apart or within L of each other , we just want to detect if this happens. We partition the interval $[L;U]$ into subintervals, depending on the degree of detail desired by the user.

Let a be a user-specified constant larger than 1 (otherwise we set $a = 2$) and let $l = \log_a(U/L)$. Define $r_i = U/a^i$ for $i \in \{0; \cdots; l\}$. Thus $r_0 = U$, and the r_i's decrease geometrically until $r_l = L$. These r_i will serve as partitions of $[L;U]$ – the smaller the r_i, the greater the accuracy.

Let T be the query trajectory. We provide a data structure that performs the following operations efficiently:

(1) $(c;r)$-**near Neighbor queries**: if there exists a trajectory in the dataset within distance r, return a trajectory in the dataset that is within cr of T. Our methods allow a linear approximation $c = O(m)$ answer with high probability in $O(m \log n)$ time.
(2) c-**approximate nearest neighbor queries**: return a trajectory whose distance to T is within c times the distance of T to its nearest trajectory in the dataset. We can answer the c-approximate nearest neighbor queries in $O(m \log^2 n)$ time.
(3) **Clustering/classification**: we can partition the dataset of n trajectories into $P_1; \cdots; P_l$, where trajectories in the set P_i are within r_i and at least r_{i+1} away from the query trajectory T. The entire partitioning takes $O(l \min(m; \log(A/L^2)))$ time. To compute only P_i for a given i, our data structure takes $O(\min(m; \log(A/L^2)))$ time.
(4) **Distance estimation**: Given another trajectory T', we can compute a constant factor approximation of the distance d between T and T' in $O(\min(m; \log(U/L)\log(A/d^2))$ time, where A is the area of the region where the trajectories are deployed.
(5) **Subtrajectory detection**: Given another trajectory T', we can identify if it is similar to a subtrajectory of trajectory T in time $O((\min(m; \log(A/L^2)))^2)$. Notice that a subcurve of a curve may have a large Fréchet or Hausdorff distance to the original curve, and so such queries are not merely distance queries.
(6) Finally, the data structure handles **fast insertions** of trajectories to the dataset. Inserting a new trajectory of length m into our data structure takes $O(m \log(A/L^2)\log(U/L))$ time (Lemma 5.1).

We now describe the proposed distance measures on the space of trajectories and the concept of LSH.

3.1 Preliminaries: distance measures and locality sensitive hashing

Distance Measures. As one baseline to compare trajectories, we use the Hausdorff distance, which can be defined as the maximum of distances from vertices on one trajectory to the nearest vertex on the other trajectory. For the following definitions we consider two trajectories $T_1 = \{u_1; u_2; \cdots u_{m_1}\}$ and $T_2 = \{v_1; v_2; \cdots v_{m_2}\}$, of length (number of vertices) m_1 and m_2, respectively.

Symbol	Description
T	trajectory (sequence of GPS locations)
m	length of trajectory
n	number of trajectories
A	area of the map
$H(T_1; T_2)$	Hausdorff distance between trajectories T_1 and T_2
$F(T_1; T_2)$	Frechet distance between trajectories T_1 and T_2
D	number of disks
r	radius of a disk
$S_{D;r}(T); S(T)$	binary sketch
$OS_{D;r}(T); OS(T)$	ordered sketch
$d_{D;r}$	distance between sketches (binary or ordered)
U, L	upper and lower bounds on the degree of detail
`	number of layers
L_i	layer i
D_i	number of disks on layer i
r_i	radius of disks on layer i
A_{dif}	area of the symmetric difference
EST_{kF}	estimator of kF

Table 1: Table of notations

Definition 3.1 (Hausdorff distance). The Hausdorff distance between T_1 and T_2 is:

$$H(T_1; T_2) = \max\{\max_{u \in T_1} \min_{v \in T_2} d(u; v); \max_{v \in T_2} \min_{u \in T_1} d(u; v)\}:$$

The complexity of computing Hausdorff distance between the two trajectories is $O((m_1 + m_2)\log(m_1 + m_2))$ [4], using Voronoi diagrams.

The Fréchet distance is another common distance measure, and it is generally believed to be more appropriate for comparing curves in a plane [4]. To define it we use the concept of traversal (as in [14]):

Definition 3.2 (Traversal). Given two trajectories $T_1; T_2$, of sizes $m_1; m_2$ respectively, a traversal $\tau = \{(i_1; j_1); (i_2; j_2); \cdots; (i_l; j_l)\}$ is a sequence of pairs of indices referring to a pairing of vertices from the two curves with the properties:

(1) $i_1 = 1, j_1 = 1, i_l = m_1$ and $j_l = m_2$
(2) $\forall (i_k; j_k) \in \tau : (i_{k+1} - i_k) \in \{0; 1\} \wedge (j_{k+1} - j_k) \in \{0; 1\}$
(3) $\forall (i_k; j_k) \in \tau : (i_{k+1} - i_k) + (j_{k+1} - j_k) \geq 1:$

Definition 3.3 (Discrete Fréchet distance). Let $\mathcal{T}$ be the set of all traversals between two trajectories T_1 and T_2. The discrete Fréchet distance $F(T_1; T_2)$ between them is:

$$F(T_1; T_2) = \min_{\tau \in \mathcal{T}} \max_{(i_k; j_k) \in \tau} \|u_{i_k} - v_{j_k}\|:$$

Locality-sensitive hashing. Solutions for approximate nearest neighbours often employ Locality-sensitive hashing (LSH) algorithms which ensure that the probability of two objects to be attributed to the same hash is high for similar objects and low for substantially different objects. A family of hash functions is a collection of mappings that are defined on the same sets (the set of all objects, which is of arbitrary size, and a set of fixed size).

Definition 3.4 (Locality-sensitive families). Let d be a distance measure defined on two trajectories T_1 and T_2. Given the values $r > 0$, $c < 1$, $0 \leq p_1; p_2 \leq 1$ with $p_1 > p_2$, a family $\mathcal{H}$ of hash functions is $(r; cr; p_1; p_2)$-sensitive if for every $f \in \mathcal{H}$:

(1) if $d(T_1; T_2) \leq r$, then $Pr(f(T_1) = f(T_2)) \geq p_1$;
(2) if $d(T_1; T_2) \geq cr$, then $Pr(f(T_1) = f(T_2)) \leq p_2:$

A $(r; cr; p_1; p_2)$-sensitive family of hash functions is useful when the collision probabilities $p_1; p_2$ satisfy $p_1 > p_2$. The LSH families we derive in this paper have $p_2 = 0$.

3.2 LSH families and the near-neighbor problem

Given a $(r; cr; p_1; p_2)$-sensitive hashing family, there is a standard framework for solving $(c; r)$-near neighbor queries. Using this framework, one can also solve the c-approximate nearest neighbor queries. We briefly describe it here, and refer the reader to more comprehensive descriptions [14, 19] for details. First, we construct a new family $\mathcal{H}'$ of hash functions by concatenating $` = \max\{1; \log_{p_2} 1/n\}$ hash functions. This decreases the collision probability to at most $1/n$. We then choose $k = (1/p_1^{`})$ hash functions from the family $\mathcal{H}'$, and insert each trajectory into k hash tables. This completes the preprocessing phase. Once the query arrives, we search among all trajectories that collide with the query in the k hash tables, and compute the distance of the query trajectory to such trajectories. We can stop as soon as a trajectory within cr distance is found, or in the reporting version, report all trajectories within cr. The space and query time of such a data structure is governed by the parameter $\rho = \log p_1 / \log p_2$. The space used is $O(n^{1+\rho} + nm)$ and the query time is $O((m \log m) n^{\rho})$ for Hausdorff distance and $O(m^2 n^{\rho})$ for Fréchet distance.

For the LSH families we derive in this paper, $p_2 = 0$, i.e., far away trajectories never hash to the same bucket. In this case, the query time is $O(m)$. These query times are for the algorithm to work with a constant probability. To get $(c; r)$-near neighbors with probability at least $1 - 1/n$, we can repeat the above process $\log n$ times, leading to an extra logarithmic overhead in the space and query time. Given a data structure to solve the $(c; r)$-near neighbor problem, one can use the concept of ring trees as described by Indyk and Motwani [19] to solve the c-approximate nearest neighbor problem.

4 COMPACT SKETCHES AND LSH FOR NEAR NEIGHBORS

We propose a simple strategy that allows us to build LSH schemes for Hausdorff, Fréchet and DTW distances. The sketches are based on randomly deploying disks on a map and considering their intersections with trajectories. We then define metrics on these sketches that are related to the distances between the corresponding trajectories.

4.1 Binary sketches for Hausdorff distance

Our approach is based on the observation that if two trajectories $T_1; T_2$ are such that the Hausdorff distance $H(T_1; T_2)$ is small, then they both go through approximately the same neighbourhoods on the map.

Fix a radius $r > 0$, and let $D = \Theta(A/r^2)$. We define our *sketch* $S(T)$ of a trajectory T as the record of its intersection with D random disks on a map:

Definition 4.1 ($(D; r)$-Binary Sketch). The Binary Sketch of a trajectory T is the binary vector $S_{D;r}(T) = e_1 :: e_D$ of length D, defined in terms of a set of D random but fixed disks of radius

r. The vector element $e_i = 1$ if the disk i intersects trajectory T, otherwise it is 0.

We define the measure of distance between the sketches as the number of bits that are different:

Definition 4.2 ($(D; r)$- Binary Sketch Distance). The Binary Sketch Distance between two trajectories $T_1; T_2$ is the Hamming distance between their $(D; r)$-Binary Sketches. That is, the number of indices with different entries: $d_{D;r}(T_1; T_2) = \{i : e_i^1 \neq e_i^2\}$.

In other words, this is the L_1 norm of the binary difference of the two vectors and it represents the number of disks that intersect exactly one of the trajectories.

4.1.1 Data structures to speed up sketch computation. Let T be a trajectory of length m. In $O(mD)$ time we can test each disk for every point in the trajectory, and retrieve the set of disks intersected by the trajectory, i.e., the position of 1s in the sketch vector. However, when D is large (recall that $D = \Theta(A/r^2)$), we can actually do better.

Notice that for a given point p on the trajectory, we want to quickly determine which of the D disks of radius r contain p. This is essentially a range searching problem: Let Q denote the centers of the D disks, and B denote a ball of radius r around p. A *circular range query* for B determines which points in Q lie inside B. The disks corresponding to those points are exactly those that contain p (the center of B). Using known range searching data structures from computational geometry [2, 22], this can be accomplished in $O(\log D + k)$ time, where k is the number of disks containing p. In summary:

Observation 4.3. *Given a set of D disks of radius r and a trajectory T of length m, the set of k disks that intersect T can be computed in $O(m \log D + k)$ time.*

Computing the binary sketch distance. For large r such that $D = O(m)$, one can compute the Hamming distance between sketches of length D in the straightforward manner. For small r, D is larger than the number of disks intersected by the trajectory (which will never be more than m, but could be significantly smaller), resulting in sparse bit vectors. When D is large enough that the vectors are very sparse, we do not store the entire sketch but just the indices of the disks intersected by the trajectory. Assume that no trajectory intersects more than I disks, where $I << D$, and $I = O(m)$. Then we can actually compute the binary sketch distance in $O(I \log I)$ time using a set intersection query, where the two sets are the sets of disks intersected by each trajectory. Thus, computing the distance takes at most $O(\min(D; m \log m))$ time, and is typically significantly smaller; e.g. for uniformly sampled trajectories, this can be done in $O((m/r) \log(m/r))$ time.

4.1.2 Binary sketches provide an LSH family.

Theorem 4.4. *Let $T_1; T_2$ be two trajectories of lengths m_1 and m_2 respectively, intersecting at least one disk. Let $m = \min(m_1; m_2)$ and $c > 1$ a constant. Then the following are true:*

(1) *If $H(T_1; T_2) < \frac{2r}{c}$ then $P(S(T_1) = S(T_2)) > 1 - \frac{8\pi r^2 Dm}{cA}$*
(2) *If $H(T_1; T_2) > 2r$ then $P(S(T_1) = S(T_2)) = 0$:*

In order to prove the theorem, we need a geometric lemma.

Lemma 4.5. *The probability that a randomly placed disk separates two vertices at distance h is bounded by $4\pi rh/A$.*

Proof. The disk separates the vertices when the center lies in the symmetric difference of the disks of radius r centered at the two vertices. Let us call this the *symmetric difference area*. This is shown pictorially in Figure 2.

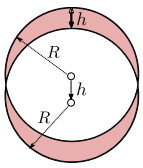


Figure 2: *Symmetric difference area:* A disk of radius r separates the two vertices if and only if its center lies in the symmetric difference of disks at those vertices given by the shaded area.

This area is bounded by $4\pi rh$. Thus the probability that a particular disk x separates the vertices, is at most $4\pi rh/A$.

Proof of Theorem 4.4. (1) For every $u \in T_1$ there is a $v \in T_2$ such that $d(u; v) \leq H(T_1; T_2) = H$, and vice versa. Let us refer to this set of pairs as the *closest pairs*. Any disk of radius r that contains u but not any vertex from T_2, must also not contain v, and therefore must have its center in the symmetric difference area for u and v. This applies for all closest pairs. Thus, any disk i such that $d_i^1 \neq d_i^2$ must lie in the symmetric difference of at least one closest pair. There are m closest pairs and any closest pair is separated by a distance at most H. By union bound, the probability that the center of a disk is in a symmetric difference area is bounded by $(4\pi mrH)/A$. Substituting $H < 2r/c$ gives the desired bound. To prove (2), we note that if the trajectories are more than $2r$ away then no disk can contain a point from each trajectory, and hence the sketches cannot be the same.

Observe that setting $D = \Theta(A/r^2)$ and $c = O(m)$ gives us a constant probability of collision, and thus by the standard framework of $(c; r)$-near neighbor data structure, we obtain the desired space and query bounds.

4.2 Ordered sketches for Fréchet distance

As opposed to Hausdorff, the Fréchet distance between trajectories takes into account the ordering of the vertices. Therefore, we propose sketches that maintain the order of the intersections with the disks.

Definition 4.6 (Ordered Sketch). Let T be a trajectory and $\mathcal{D}$ the set of disks spread on the map. We define by the *Ordered Sketch* the sequence of indices of the disks that T enters and exits.

Figure 1 shows a trajectory a passing through disks 1; 2; 3; 4. The trajectory first enters disk 1, exits 1, enters disk 2, enters disk 3 and so on.

We define a measure of distance (dissimilarity) and also a measure of similarity between two ordered sketches, and therefore, between their corresponding trajectories. These will help us answer distance related queries when we describe our Multi Resolution Trajectory Sketch (MRTS) data structure.

Definition 4.7 ((D; r)-Ordered Sketch Distance). The Ordered Sketch Distance between two trajectories $T_1; T_2$ is the edit distance between their $(D; r)$ Ordered Sketches. That is, the number of insertions, deletions and substitutions required to transform one ordered sketch into another.

Definition 4.8 ((D; r)-LCS measure). The LCSM (Longest Common Subsequence Measure) between two trajectories $T_1; T_2$ is the length of the LCS between their $(D; r)$ Ordered sketches.

4.2.1 Ordered sketches provide an LSH family. We first find the probability of collision of two trajectories in terms of their Fréchet distance.

Lemma 4.9. *Given two curves $T_1; T_2$ with $m_1; m_2$ vertices, $m = \min(m_1; m_2)$ and D disks of radius r the probability that they hash to the same sequence is bounded by:*

$$P(OS(T_1) = OS(T_2)) \geq 1 - \frac{4\pi r D m}{A} F(T_1; T_2):$$

Proof. Suppose the trajectories do not hash to the same sequence. From Lemma 3 in [14] we know that an optimal traversal T will separate the trajectories in at most m components, each of them a star. Intuitively a star component has one vertex on one trajectory (traversal is static on this curve), and some vertices on the other trajectory (where the traversal is happening). We consider E_k to be the event that a disk separates a pair of vertices in the component k.

Since the component is a star, there must exist vertex v such that v is connected to all other vertices in the component. All the lengths are less than $F(T_1; T_2)$. The probability that any pair is separated is less than $4\pi r F(T_1; T_2)/A$ (this is just the symmetric difference area - similar to what we had before). There are at most m components in the traversal and D disks, therefore, by union bound, the probability that the hashes differ is bounded by $\frac{4\pi r D m}{A} F(T_1; T_2)$.

This provides us with the following locality sensitive hash family.

Theorem 4.10. *Let $T_1; T_2$ be two trajectories of lengths $m_1; m_2$, intersecting at least one disk. Let A be the area of the region, r the radius of the disk, D the number of disks, $m = \min(m_1; m_2)$ and c a constant. Then the following are true:*

1. *If $F(T_1; T_2) < \frac{2r}{c}$ then $P(OS(T_1) = OS(T_2)) > 1 - \frac{8\pi r^2 D m}{cA}$*
2. *If $F(T_1; T_2) > 2r$ then $P(OS(T_1) = OS(T_2)) = 0$:*

Proof. The second part is apparent as there is no disk which intersects both trajectories. For the first part we use Lemma 4.9.

4.3 Summary of LSH and near neighbor results

We have shown how the binary sketches and the ordered sketches provide a locality sensitive hash family for the Hausdorff and the Fréchet distances, respectively. In both families, setting the number of disks $D = \Theta(A/r^2)$ and $c = O(m)$ gives us a constant collision probability of nearby trajectories, and we have zero collision probability for far-away trajectories. From the framework described in Section 3.2, we obtain that

Theorem 4.11. *There exists a data structure using binary and ordered sketches, that takes space $O(n)$ memory words, and returns an $O(m)$ approximation (in the Hausdorff metric using the binary sketch, and in the Fréchet metric using the ordered sketch) to the*

1. *$(c; r)$-near neighbor problem in $O(m \log n)$ time.*
2. *c-approximate nearest neighbor problem in $O(m \log^2 n)$ time.*

Now we turn to the other kinds of queries, and the data structure we use to solve them.

5 MRTS: MULTI RESOLUTION TRAJECTORY SKETCH

In this section we describe the MRTS (Multi Resolution Trajectory Sketch), a layered data structure that we will use to solve distance estimation, classification, clustering and subtrajectory detection queries.

5.1 Description

Recall that U and L are upper and lower bounds on the degree of detail desired by the user, and $r_i = U/a^i$ for $0 \leq i \leq h$, with $l = \log_a(U/L)$ and $r_l = L$. Define $D_i = \Theta(A/r_i^2)$ for $0 \leq i \leq l$.

Our layered data structure has l layers, denoted by L_i, $0 \leq i \leq l$. Layer L_i stores

1. The binary sketches of all the n trajectories, for D_i random disks of radius r_i each. When $D_i = \omega(m)$, we store the set of disks intersecting a trajectory as opposed to the bit vector of length D_i.
2. The ordered sketches of all the n trajectories, for D_i random disks of radius r_i each.
3. In addition, there are forward and backward pointers between a trajectory in the dataset and its corresponding sketches at each layer. Each sketch also stores a counter with the number of trajectories pointing to it.

Insertions. When a new trajectory T is to be inserted in the dataset, we use the data structure referred to in Observation 4.3. Computing the sketch on layer i requires $O(m \log D_i + k_i)$ time, where k_i is the number of disks of radius r_i that intersect T. Since $D_i \leq D_l$, overall, this costs us at most $O(m \log D_l l + k)$, where k is the total number of disks intersecting T. Plugging in the values of D_l and l, we get

Lemma 5.1. *A new trajectory T can be inserted into MRTS in $O(m \log(A/L^2) \log(U/L) + k)$ time, where k is the total number of disks intersecting the trajectory.*

Space. Any point p on a trajectory T lies in $O(1)$ out of the D_i disks for every i. This is because D_i is the number of disks required to cover the region A and the disks are deployed uniformly at random in the domain. Thus the sketch of T on layer i requires at most $O(m)$ space, and so in total the sketches for the trajectory T in the entire MRTS data structure uses $O(ml)$ space. Since there are n trajectories,

Observation 5.2. *The MRTS uses at most $O(nm \log(U/L))$ words of memory in space.*

Notice that $O(nm)$ words of space are required to store the dataset. Moreover, the bound above is pessimistic in the sense that

we do not consider the fact that for densely sampled trajectories, many points of the trajectory will lie in the same disk. In fact, one can show that for uniformly sampled trajectories, $O(r)$ points lie in a disk of radius r. This reduces the space usage of our data structure to $O(\frac{nm}{L}\log(U/L))$ memory words.

5.2 MRTS and distance queries

Now we show how to handle other types of queries mentioned in our problem description. For the sake of brevity, we will restrict ourselves to the Fréchet case, which is arguably harder than the Hausdorff case, and more interesting. We point out that all of our results generalize to the Hausdorff case with the same (or better) bounds. However, before we describe our query procedure, we need some geometric lemmas.

Lemma 5.3. *Consider two disks of radius r, and let h be the distance between their centers. Let A_{dif} denote the area of their symmetric difference (see Figure 2). Then 1) if $h > 2r$, $A_{dif} = 2\pi r^2$, and 2) if $h \leq 2r$, $A_{dif} \geq 2rh$.*

Proof. Part 1) is obvious, as the disks are disjoint. For part 2), the area of the symmetric "lens", or the intersection of the two disks, is given by $2r^2\cos^{-1}(h/2r) - (h/2)\sqrt{4r^2-h^2}$. The area of the symmetric difference is therefore

$$\begin{aligned} A_{dif} &= 2(\pi r^2 - 2r^2\cos^{-1}(h/2r) + (h/2)\sqrt{4r^2-h^2}) \\ &= 2r^2(\pi - 2(\pi/2 - (h/2r) - O((h/2r)^3))) + (h/2r)\sqrt{1-(h/2r)^2} \\ &\geq 2r^2(h/r + O((h/2r)^3) \\ &\geq 2rh: \end{aligned}$$

This gives us the following crucial observation:

Observation 5.4. *Let T_1 and T_2 be two trajectories such that their Fréchet distance is F. Then the probability that a randomly chosen disk of radius r intersects one trajectory but not the other is at least $2rF/A$.*

This is because if we look at the component realizing the Fréchet distance (the maximum length of the leash), there must be a pair of vertices $u \in T_1$ and $v \in T_2$ that are F apart. A disk separates these two vertices with probability equal to the measure of their symmetric difference area, which by the above lemma is at least $2rF/A$.

Lastly, we show that the ordered sketches already provide an upper bound on the Fréchet distance.

Lemma 5.5. *Let T_1 and T_2 be two trajectories with Fréchet distance F. Consider their ordered sketches with D disks of radius r, and assume all vertices overlap at least one disk. If the ordered Sketches coincide, then $F < 2r$.*

Proof. If the ordered sketches coincide then for every vertex in T_1 there is at least one vertex in T_2 such that the distance between them is lower than $2r$, because they should both belong to the same set of disks. There is a traversal that groups every pair of such vertices, giving a cost of at most $2r$. This means than an optimal traversal has a lower cost, therefore the Fréchet distance is less than $2r$.

5.2.1 Querying the MRTS. We now describe how we query the MRTS for different kinds of queries.

Distance Estimation. Given the indices of two trajectories T_1 and T_2 in the dataset, we want to approximate the Fréchet distance between them. Note that precomputing the distance of a newly inserted trajectory to all the existing trajectories in the dataset trivially solves this problem in $O(1)$ time, but the cost is the high insertion time, at least $O(m^2n)$. The MRTS has low insertion time, and is still flexible enough to answer such queries faster.

Let us consider the optimal traversal of the two trajectories that realizes their Fréchet distance F. As observed, this traversal can be broken down into at most m components, each of which is a star. In some of these components the distance is $O(F)$, whereas in others it is significantly lower. Let k be the number of components in which the maximum distance between the vertices of the component (the center of the star and the other vertices on the second trajectory) is $\Theta(F)$. We will first estimate the product kF.

The main reason in estimating kF is that the symmetric difference area of the two trajectories depends linearly on k. If $k = 1$, then the two trajectories travel very close together until they reach this component; in this case the area of the symmetric difference is $O(rF)$. On the other extreme, two trajectories could be traveling at a constant distance F from each other (thus giving the same Fréchet distance between the trajectories), but the area of the symmetric difference is $O(rFk)$.

Let $\grave{}_i$ denote the fraction of disks on layer i that intersect one trajectory but not the other. Clearly, $\grave{}_i$ is an unbiased estimator for the symmetric difference area (for disks of radius r_i). We compute $A_{dif} = \sum_{i=1}^{h} \grave{}_i A/r_i$, and let $\text{Est}_{kF} = A_{dif}/h$. We can show that this is an unbiased estimator of the true value of kF, and by increasing the number of disks at each layer, we can get higher concentration for this estimator. The expected value of $\grave{}_i$ on the other hand equals $O(r_i kF/A)$, as the probability that one disk separates the two trajectories equals $r_i kF/A$.

Query Algorithm: Report r_j, where j is the smallest value of i such that $\grave{}_i \geq r_i \text{EST}_{kF}/A$.

We conjecture that this value of r_j reported is a constant factor approximation of the Fréchet distance between the trajectories. Note that if one just wants an upper bound, the r_i corresponding to the last layer i such that $S_1^i = S_2^i$ (the sketches coincide) gives us an upper bound,as by Lemma 5.5, we know that the Fréchet distance between the two trajectories is at most $2r_i$, since the sketches coincide.

This procedure requires us to examine the sketches at all levels in the worst case, requiring $O(\min(m; \log(U/L)\log(A/d^2))$ time. To get an upper bound we can stop at the first layer where we discover the sketches are different.

Classification/clustering. Our classification procedure is simple: for a given interval $[r_{i+1}; r_i]$, we define the set P_i to be all the trajectories that have the same sketch on layer i of the MRTS as the query trajectory.

Using the pointers, we can report the trajectories in P_i, or, if one just needs the count, we can read the counter value of the bucket corresponding to the sketch of the query trajectory. Running this

procedure for all $1 \leq i \leq l$ gives us the partition of the dataset. Procedures such as [23] can be used to select representative elements as centers for exemplar based clustering. Clustering of trajectories can be applied to anonymize locations traces [28].

Subtrajectory Detection. Given the query "Is trajectory T_1 r_i-close to a subtrajectory of T_2?", we first insert T_1 and T_2 in the MRTS if they haven't been inserted yet. Then we locate the sketches of T_1 and T_2 on layer $i-1$. We compute the LCS between these sketches, and answer yes if the sketch for T_1 appears as a subsequence, and no otherwise. To get accurate results with high probability, we can increase the number of disks on layer $i-1$, and take the majority answer.

6 EXPERIMENTS

We tested the performance of binary and ordered sketches on two real datasets. The experiments show that:

- there is a correlation between the proposed distace measures and Hausdorff (strong correlation), Fréchet and DTW (Section 6.1);
- the correlation holds even with a small number of disks or with incomplete data (Sections 6.1, 6.3);
- the sketches achieve high pruning ratio with good accuracy, less storage space and better running time in nearest neighbours queries (Sections 6.2, 6.4, 6.5).

We ran all experiments on both the CRAWDAD dataset of taxis in Rome [9] and the ECML/PKDD dataset of Taxi Trajectories in Porto [1].

Data preparation. The ECML/PKDD (Porto) dataset describes the mobility of 442 taxis driving in the city of Porto (Portugal) for 1 year and is composed of 1:7 million datapoints. The GPS locations (latitude, longitude) are recorded every 15 seconds with mobile data terminals installed in the vehicles. Each trajectory represents a complete taxi trip taken by a passenger. The lengths vary from $1km$ to $15km$. We randomly selected 1000 trajectories in a $5km \times 10km$ area of the map.

The CRAWDAD (Rome) dataset [9] contains the GPS locations of 320 taxi drivers working in the city centre of Rome (Italy). Each trace is for one driver for one day at about 7 second intervals, giving roughly 22 million datapoints. To obtain trajectories similar to individual trips, as in the Porto dataset, we split each trajectory into trips, such that the length of each is at most a 3 times the distance between source and destination. This gives trajectories with lengths roughly between $2km$ and $5km$. We randomly selected 1000 trajectories in a $7km \times 2km$ area of the map.

Choice of parameters. For all experiments with a fixed number of disks we picked uniformly at random 50 disks of radius $2km$. In general, the choice of radius depends on the scale of the problem. Suppose we consider two $1km$ trajectories to be close if their distance is less than $10m$. We choose enough disks to cover the map. Based on results in Section 4, we want to maximise $1 - \frac{8\pi r^2}{c}$ while $\frac{2r}{c} = 10m$. For $r = 2km$, the probability $1 - \frac{8\pi r^2}{c}$ becomes 0:75, which is a lower bound for the probability that the sketches are the same when the trajectories are less than $10m$ apart.

Experimental setup. The experiments were implemented in Python 3.4.9 and ran on an Intel Core i5-4570 CPU @ 3.20GHz × 4 processor. Time comparisons were done by running 20 parallel jobs.

6.1 Correlation with Hausdorff, Fréchet and DTW

We checked how the proposed distance measures compare to Hausdorff, Fréchet and DTW. Figures 3 (a), (b), (c), (e), (f) show the results for the Porto and Rome dataseta. For each pair of trajectories we compute the distance: with the binary unordered sketches for Hausdorff and with the ordered version for Fréchet and DTW. We grouped the possible values in 30 bins and show in the plots the median value (white line) and the 5-95, 10-90, and 25-75 percentiles (the shaded areas). We see that the sketch distance measure bears a clear correlation to the traditional measures. For Hausdorff the correlation is high ($\approx$ 0:8). For Fréchet and DTW, while the correlation is lower, there is a clear distinction between close and distant trajectories, showing the utility of locality sensitive hashes.

6.1.1 Effect of radius and number of disks. This correlation naturally raises a question of how it is influenced by parameters such as the number and radii of disks. Figure 4 shows the effect of disk radius; each subplot corresponds to results with a fixed number of disks and a varying radius between $500m$ and $4km$. The shaded area corresponds to the 5 to 95 percentiles based on 30 trials. For each trial we computed the Pearson correlation coefficient between our distance measure and Hausdorff distance.

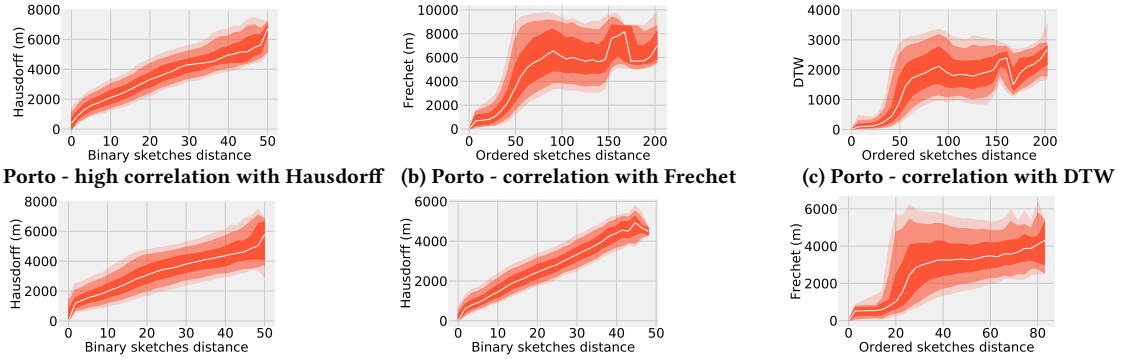
In Figure 4(a) a radius size of around $2{:}5km$ achieves the maximum correlation, beyond which the disks become less discriminatory and the correlation decreases. We also observe that too small a radius does not perform as well – to use a smaller radius we would need to consider more disks.

The correlation also increases with the number of disks. A higher number also assures low variability in performance between trials. Even with a low number, such as 10, we get a good correlation of $\approx$ 0:7 or more as long as the radius is in the right range ($1{:}5km$-$3km$). Beyond a certain number, increasing the number of disks does not improve performance. For a good correlation we need to choose enough disks to get a cover of the map, but having more disks offers a more fine grained distance measure at the cost of greater storage and computation, as we discuss in relation to nearest neighbours queries in Section 6.2.

6.2 Nearest neighbour search

For a variable number of disks (10, 30, 50) and fixed $r = 2km$, we computed the nearest neighbour for each trajectory in the dataset using Hausdorff and Fréchet distances. To prune the search space using our sketches we selected only those trajectories that had sketch distance equal to 0, 1, 2 etc. We measured in what proportion of cases the true nearest neighour is in the remaining set (accuracy), and how large the eliminated set is compared to the total dataset (pruning ratio). In Figure 5 we show how the pruning ratio changes with accuracy.

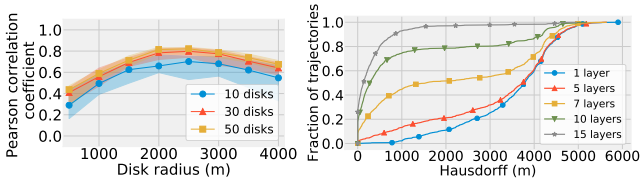
There is little variation in the accuracy for a specific number of disks. Interestingly, changing the number of disks does not influence the results too much for nearest neighbor search with LSH; with a small number we can achieve a high pruning ratio and high



(a) Porto - high correlation with Hausdorff (b) Porto - correlation with Frechet (c) Porto - correlation with DTW

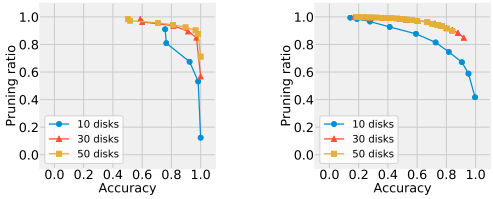
(d) Porto - 90% of locations removed (e) Rome - high correlation with Hausdorff (f) Rome - correlation with Fréchet

Figure 3: Correlation of binary and ordered sketch distances with Hausdorff, Fréchet , and DTW for the Porto and Rome datasets (the 5-95, 10-90, and 25-75 percentiles).



(a) (b)

Figure 4: (a) Correlation coefficient with Hausdorff distance depending on the disk radius and the number of disks. Variance decreases when increasing the number of disks. (b) CDF of the Hausdorff distance error between true and found nearest neighbours. With more layers the error decreases.



(a) Hausdorff (b) Fréchet

Figure 5: The performance for nearest neighbours queries inscreases with the number of disks. A small number (30) suffices.

accuracy. However, larger number of disks provides better resolution when using the sketches as an estimate of distance.

Figures 5 show that accuracy can be as high as about 80% at a pruning ratio of over 80% for both Hausdorff and Fréchet distances, even with a small number of disks. Thus, this method allows us to restrict our attention to a small fraction of the trajectories and still obtain the correct nearest neighbor in most cases.

Multiresolution trajectory sketches. In Figure 4 we show how the distance error to the nearest neighbour changes with the different number of layers in the multiresolution sketches. We considered 1, 5, 7, 10 and 15 layers. At the first level the disks have radius $r = 2km$; this decreases by a factor $\varepsilon = 0.1$ at each level. The number of disks at level i is given by $A/r_i^2 \log(A/r_i^2)$, where r_i is radius at level i.

To find the nearest neighbours given a query trajectory we compare the sketches at every layer, starting from the top, and only advance to the next one if the sketches match. When we reach the last layer, the set of the remaining trajectories is searched for the nearest neighbour. The difference in distance between the true nearest neighbour and the one we found gives the error. Figure 4 shows the CDF of the error.

At the first level the number of disks is low (13). Performance increases quickly with the number of layers. In practice, the performance can be further improved by checking for approximate matches between sketches instead of exact matches (corresponding to distance 0).

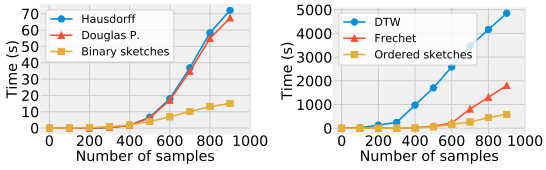
6.3 Robustness to data loss and privacy of locations

Trajectory data can be subject to sensing errors and noise and often has missing data. In other cases, a lower frequency of location sampling is preferred to maintain the user's privacy in between check-ins [24]. The disk distance performs well even when large parts of the trajectories are missing. Interestingly, when only 10% of the datapoints in each trajectory are retained, the disk distance is still highly correlated with the Hausdorff distance, as can be seen in Figure 3 (d). In this experiment we computed the disk distance on only a 10% sample of the location points in each trajectory, and plotted against their true initial Hausdorff distance.

Beyond the natural robustness to unreliable sensing, this result implies efficiency of storage and computations as only a small random sample needs to be stored for useful comparison.

6.4 Time efficiency

We compared the running times of our method using pruning with Hausdorff, Fréchet and DTW when computing distance matrices.



(a) Binary sketches (b) Ordered sketches

Figure 6: Better performance in terms of running time.

In the first experiment shown in Figure 6 (a) we used the standard Hausdorff implementation between point sets and computed the time for sampled sets of up to 1000 trajectories. Using the Douglas-Peucker algorithm to first simplify the trajectories did not bring a large improvement in the computation time. The time taken to compute all pairwise sketch distances is more than 5 times faster

than computing Hausdorff distance for 1000 trajectories. This also includes the preprocessing time of picking the disks and computing the sketches.

For the second experiment in Figure 6 (b) we computed Fréchet and DTW distances efficiently with Python libraries. The results show that the sketch distance computation is 2 times faster than Fréchet for 1000 trajectories when using 50 disks - enough to get a 90% accuracy for nearest neighbours search with 80% pruning ratio, as can be seen in Figure 5.

6.5 Comparison with existing LSH method

We compared the performance of the ordered sketches with the grid-based sketches proposed by Driemel and Silvestri [14]. In their work the hashes are constructed in the following way:

- consider a grid that covers all trajectories;
- each vertex in the trajectory is replaced by the closest node in the grid;
- from the resulting sequence of nodes the consecutive duplicates are removed.

In experiments we considered various sizes of grids, ranging from 100m to 15km. For every size s we built a grid that covers the map, where the distance between any two nodes is equal to s. We denote the number of resulting nodes by n_{grid}. To compare with our proposed method, we deployed $n_{grid}/2$ disks of diameter s. Notice that we are using the ordered sketches (not the fixed size binary sketches) because the grid approach gives a LHSs scheme for Fréchet . Two ordered sketches are considered the same if the edit distance is at most 2. Figure 7 shows the results: the two methods perform approximately the same in terms of pruning ratio and accuracy (slightly better for the disks version), but the sizes of our sketches are almost half the size of the grid sketches.

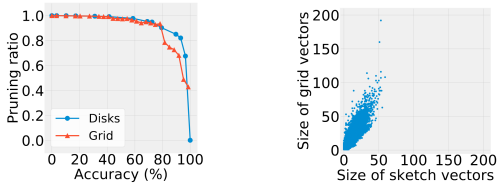


Figure 7: Better performance in terms of (a) accuracy and (b) storage size.

7 CONCLUSION

We presented a randomized sketching scheme to compactly represent trajectories, and established its effectiveness in both theory and practice of trajectory processing. Planar networks and trajectories are common in robotics, biological systems and many other domains. Adaptation of this basic randomized scheme in these areas remain to be investigated, as well as in trajectories embedded in higher dimensions.

REFERENCES

[1] Ecml/pkdd porto taxi data. https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i.

[2] A. Aggarwal, M. Hansen, and T. Leighton. Solving query-retrieval problems by compacting voronoi diagrams. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 331–340. ACM, 1990.

[3] J. Alman and R. Williams. Probabilistic polynomials and hamming nearest neighbors. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 136–150. IEEE, 2015.

[4] H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13(3-4):251–265, 1995.

[5] H. Alt and M. Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995.

[6] A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 793–801. ACM, 2015.

[7] S. Arya, D. M. Mount, A. Vigneron, and J. Xia. Space-time tradeoffs for proximity searching in doubling spaces. In *European Symposium on Algorithms*, pages 112–123. Springer, 2008.

[8] A. Backurs and A. Sidiropoulos. Constant-distortion embeddings of hausdorff metrics into constant-dimensional l_p spaces. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 60. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[9] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from http://crawdad.org/roma/taxi/20140717, July 2014.

[10] K. Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless seth fails. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 661–670. IEEE, 2014.

[11] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[12] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 11(2):112–122, 1973.

[13] A. Driemel, A. Krivošija, and C. Sohler. Clustering time series under the fréachet distance. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 766–785. Society for Industrial and Applied Mathematics, 2016.

[14] A. Driemel and F. Silvestri. Locality-Sensitive Hashing of Curves. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77, pages 37:1–37:16, 2017.

[15] A. Ghosh, B. Rozemberczki, S. Ramamoorthy, and R. Sarkar. Topological signatures for fast mobility analysis. In *26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL)*. ACM, 2018.

[16] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[17] P. Indyk. On approximate nearest neighbors under lâĹđ norm. *Journal of Computer and System Sciences*, 63(4):627–638, 2001.

[18] P. Indyk. Approximate nearest neighbor algorithms for fréchet distance via product metrics. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 102–106. ACM, 2002.

[19] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. *Proc. 30th ACM Symposium on Theory of Computing (STOC)*, 8:321–350, 1998.

[20] P. Katsikouli, M. Astefanoaei, and R. Sarkar. Distributed mining of popular paths in road networks. In *DCOSS 2018-International Conference on Distributed Computing in Sensor Systems*, pages 1–8. IEEE, 2018.

[21] P. Katsikouli, R. Sarkar, and J. Gao. Persistence based online signal and trajectory simplification for mobile devices. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 371–380. ACM, 2014.

[22] J. Matousek. Reporting points in halfspaces. *Computational Geometry*, 2(3):169–186, 1992.

[23] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause. Distributed submodular maximization. *The Journal of Machine Learning Research*, 17(1):8330–8373, 2016.

[24] J. Niedermayer, A. Züfle, T. Emrich, M. Renz, N. Mamoulis, L. Chen, and H.-P. Kriegel. Probabilistic nearest neighbor queries on uncertain moving object trajectories. *Proc. VLDB Endow.*, 7(3):205–216, Nov. 2013.

[25] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1186–1195. Society for Industrial and Applied Mathematics, 2006.

[26] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49, 1978.

[27] D. Xie, F. Li, and J. M. Phillips. Distributed trajectory similarity search. *Proceedings of the VLDB Endowment*, 10(11):1478–1489, 2017.

[28] J. Zeng, G. Telang, M. P. Johnson, R. Sarkar, J. Gao, E. M. Arkin, and J. S. Mitchell. Mobile r-gather: Distributed and geographic clustering for location anonymity. In *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2017.