Distance and Time Sensitive Filters for Similarity Search in Trajectory Datasets
(Extended Abstract)

Madhav Narayan Bhat * Paul Cesaretti† Mayank Goswami‡ Prashant Pandey§

Abstract

Similarity searching is a well-studied problem in trajectory processing, where given a query trajectory, one wants to report all trajectories in the database similar to the query. Several works have focused on developing near neighbor data structures for trajectories under various metrics, e.g., Frechet distance or Hausdorff distance. A long-standing criticism of such measures is that they ignore the temporal aspect of time-stamped trajectories, with some recent works focusing on developing near neighbor methods that incorporate some of this temporal information.

Furthermore, all near neighbor data structures use space that is super-linear (and sometimes a high-degree polynomial) in the size of the trajectory database. With trajectory datasets getting larger everyday, these data structures may not fit in RAM, and likely stored on external disk. However, in order to avoid expensive disk-accesses for every query, a lightweight filter on RAM that reports YES when there is a trajectory in the database similar to the query, and NO otherwise (most of the time), is highly desirable.

This work focuses on optimizing near neighbor data structures by addressing the two issues above. We present a filter that answers YES or NO to similarity queries quickly, incorporates both spatial and temporal aspects of the trajectories, and can be configured to fit in RAM. Our filter can be used in conjunction with any near neighbor data structure on disk to provide a two-tiered system for fast similarity searches in trajectory databases.

Our main technique involves computing sketches from a suitable-scaled 3D representation of the trajectories, followed by building distance-sensitive filters for the sketches. Our experiments first demonstrate that the distance measure thus obtained is meaningful, both spatially and temporally. Furthermore, the experiments indicate a low false negative rate, fast filtering time, and a 95% space savings on the original dataset, making these filters a natural choice in a two-tiered system to reduce processing time in very large trajectory datasets.

*Columbia University
†Graduate Center, CUNY.
‡Queens College and Graduate Center, CUNY.
§VMware Research

1 Introduction

Preprocessing trajectory datasets have several applications, ranging from social sciences to smart cities and transport. Some examples are:

1. **Traffic Analysis:** The goal is to identify flow of traffic in a given, predefined area so as to locate common bottle neck points which would suggest areas which could be improved by additional road networks to alleviate traffic congestion.
2. **Contact Tracing:** Here we are looking to identify a collection of individuals who may have been in contact with a contagious individual by returning the set all those that have visited the same areas or traveled along a similar path.
3. **Web Analytics:** Here a users web traffic consists of a sequence of hyperlinks and the goal is to find users with similar browsing habits.
4. **Weather Prediction:** Here we are concerned with forecasting the weather by being able to track fronts according to a given path of interest.

Each of these applications amounts to searching for a set of similar trajectories from a large collection of possible trajectories. The nearest-neighbor search problem (NNS) asks, given a set S of n points in a metric space, preprocess them into a data structure so as to quickly answer queries of the form "given q, report the nearest neighbor of q in S", or "given q and $r > 0$, report any point in S within distance r of q".

There exist several data structures for the NNS problem for general metric spaces (Hamming [20] [6] [4], Euclidean [20] [1] [5] [6] [4], Angular [3], Edit [19], Jaccard [15], [2], [11]), and also for metrics on the space of trajectories ([17], [7],[10], [13], [14]).

For trajectory datasets, the most commonly studied metrics are Hausdorff [16], Fréchet [14] [7], [17], and the Dynamic Time Warping metrics (DTW). Su et al. [22] provide a thorough overview of trajectory data analysis results. However, these data structures suffer from two potential limitations:

- Most NNS data structures for very large trajectory datasets will not fit in RAM, as their space usage is typically (a high) polynomial of the space used to represent the trajectories, and
- NNS data structures for Fréchet, Hausdorff, and DTW on trajectories usually treat trajectories as curves lying in the plane, and therefore ignore the spatial aspect that is usually present in many applications.

The decision version of similarity search problem can be carried out through a distance-sensitive Bloom filter. A Bloom filter [9] is a succinct data structure approximating a set to support membership queries with tunable false positive rate. It is often used to construct a two-tiered data structure which keeps a set implicitly on RAM while explicitly storing the set on SSD. As Bloom filters have only one-sided errors, having no false negatives, they save the user from making unnecessary trips to disk which incur an incredible cost in any system application. The distance-sensitive version generalizes the functionality of Bloom filter to answer queries of the form "Is x close to an element in the database?", where closeness is measured under a suitable metric. Goswami et al. [18] have the such data structure but it is limited to similarity queries in Hamming space. We aim in this work to construct a such two-tiered data for trajectory data.

1.1 Our Results

We show how to build a data structure to compute the decision version of the approximate nearest neighbor for trajectory data with an added temporal component under both the classical Hausdorff distance and recent SketchMin distance of [21] . This data structure reduces the space complexity of directly storing the trajectories outright by over 95% and has low mismatch when considering metrics like Hausdorff and SketchMin Therefore, we can construct an efficient two-tier system for performing similarity queries on trajectory data. Furthermore, we show a link between the Hamming space and two other metrics, namely Hausdorff and SkethcMin, which allows the possibility of constructing a two-tier system to perform similarity queries on a large dataset stored on SSD.

2 Algorithm

The construction of our data structure follows four steps:

1. Lift each trajectory to 3D by taking into account its temporal component.
2. Scale the time component, possibly incorporating a user-defined parameter indicating the importance of the temporal aspect.
3. Construct a binary sketch representation of each trajectory.
4. Store the binary sketches in a distance-sensitive Bloom filter data structure of Goswami et al. [18].

Step 1: Lifting Trajectories. Lift each trajectory $T = (v_1, t_1), ..., (v_l, t_l)$ to 3D by placing the first component in the xy-plane and the time in which the a trajectory arrives at this location on the z-axis.

Step 2: Scaling the Time Component Let s be the average speed of the trajectories in the dataset. Also, let $w \in [0, 1]$ denote a user defined parameter indicating the importance of the temporal aspect in the application. $w = 0$ corresponds to the temporal aspect being irrelevant, and the trajectory being just a 2D curve, as is treated in most of the literature. On the other hand, $w = 1$ corresponds to the temporal aspect being as important as the spatial aspect.

For each $T_i \in \mathcal{T}$ we scale its time component by sw; that is swt_i for all $i \in [n]$. For the remainder of this article we will assume $w = 1$.

Our time scaling means that one unit of time corresponds to the average speed s of the trajectories in $\mathcal{T}$. Consider points $p = (x, y, t), q = (x', y', t)$ and $r = (x, y, t + 1)$, where $d(p, q) = s$. Here p and q are equal temporally but are a distance s away. Similarly, p and r occupy the same space but at one unit time difference; see figure 1.

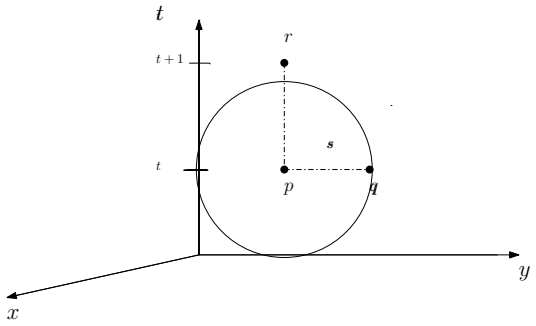

Figure 1: Scaling the Temporal Component

We would like the scaling to be such that these distances are the same; that is, $d(p, q) = d(p, r) = s$, where $d(.,.)$ is the standard Euclidean norm between two points.

Step 3: Computing Binary Sketches. Deploy and store a set of d random spheres in 3D $S = \{s_1, ..., s_d\}$, and for each trajectory, test for its intersection with these spheres; see Figure 1. The sketch $\sigma(T_j)$ for trajectory $T_j \in \mathcal{T}$ is now a d-dimensional bit vector $b_j[1, ..., d]$ where the i-th entry $b_j[i] = 1$ if and only if T_j contains point $p_k = (v_k, t_k)$ such that $p_k \in s_h$, for some $1 \leq k \leq l$ and some $1 \leq h \leq d$.

Step 4: Constructing Filter.

Given two sketches $\sigma(T_i)$ and $\sigma(T_j)$ for trajectories $T_i, T_j \in \mathcal{T}$, a simple measure of distance between them is the Hamming distance $d_H(\sigma(T_i), \sigma(T_j))$, which is defined as the number of positions in which b_i and b_j differ. As was noted in [8], when spheres are deployed randomly, similar trajectories will likely intersect a similar set of spheres which will make their respective sketches similar.

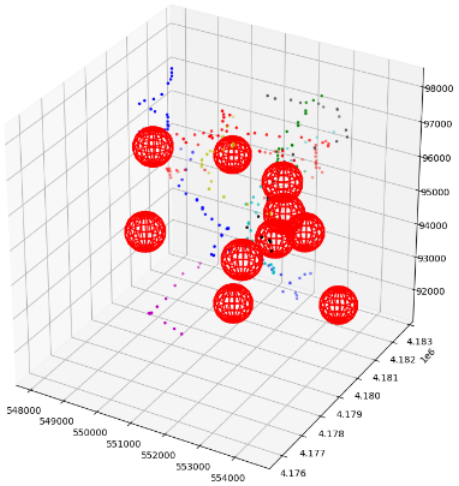

Figure 2: Trajectory sketches with respect to its intersection with 10 randomly deployed spheres.

We now store this collection of binary signatures in the (r, c, ε) distance-sensitive Bloom filter data structure by Goswami et al. [18].

Such a data structure stores a set $S \subset \{0, 1\}^d$ of points from d-dimensional Hamming space, such that given a query $q \in \{0, 1\}^d$, the data structure reports:

- YES, if $q \in \{x \in \{0, 1\}^d : D(x, S) \leq r\}$.
- NO, if $q \in \{x \in \{0, 1\}^d : D(x, S) > cr\}$, with probability at least $1 - \varepsilon$.

where $D(q, S) = \min_{p \in S} d(q, p)$ is the minimum Hamming distance between q and any point in S. The following guarantee is provided:

Theorem 2.1. *[18] There exists a (r, c, ε)-distance sensitive approximate membership filter with pointwise error which requires*

$$O\Big(n\Big(\frac{r}{(c-1)} + \Big(\frac{c}{c-1}\Big)^2 \log(\frac{n}{\varepsilon})\Big)\Big)$$

bits for any $c > 1$ on a set S of n points. When $c \geq 2$, the filter uses $O(n(\frac{r}{c} + \log(\frac{n}{\varepsilon})))$ bits, and it is optimal if $r/c \leq \log(n\varepsilon)$ or $\varepsilon \leq 1/n^{1+o(1)}$.

Query: Given a query trajectory Q and user defined similarity range of interest $0 \leq r \leq d$, compute its sketch $\sigma(Q)$ by testing its intersection with the set of spheres S. Now given its d-dimensional bit representation b_Q, feed it to the (r, c, ε)-DSBF D. If D answers YES, then return YES, otherwise return NO.

3 Experiments

The experiments were broken into two phases. In phase 1 we explore the correlation between Hamming distance and three other distance measures, namely that of Hausdorff, SketchMin of [21], and the Spatio-Temporal distance of [12]. If the correlation is sufficiently high to establish a link between the two spaces, we investigate affect of the number of spheres and size of radius on the correlation between Hamming space and the metric in question. We additionally look into the affect the Hamming weight of each vector has on this space correlation. Phase 1 determines the parameter values, namely the number of spheres and the size of a radius, which will remain constant through the second set of experiments. In phase II we look into the rate of mismatch between the two spaces in correctly labeling trajectories as being within a user prescribed range before and after they are stored in the filter. Lastly, we look at the space savings of using the filter.

3.1 Defining Testing Measures

The metric of interest in analyzing our data structure is the level of mismatch in labeling between the two spaces. We define such events to be a positive (negative) metric mismatch. Here is the formal definition of both:

Definition 3.1. A **Positive Metric Mismatch (PMM)** for a metric space $M = (d, X)$ on query trajectory Q occurs if for $r, R, c > 0$ there is a signature for a trajectory T such that $d_H(\sigma(T), \sigma(Q)) \leq r$ but for all trajectories T' in our dataset $d(T', Q) > cR$.

Definition 3.2. A **Negative Metric Mismatch (NMM)** for a metric space $M = (d, X)$ on query trajectory Q occurs if for $r, R, c > 0$, on all signatures $\sigma(T)$, $d_H(\sigma(T), \sigma(Q)) > r$ but there exists a trajectory T' in our dataset such that $d(T', Q) \leq cR$.

The efficacy of our data structure is dependant on how often these events arise and whether or not they can be reduced. Hence, we are concerned with the rate at which this occurs over a collection of queries.

Definition 3.3. Let $r, R \geq 0$, $c > 0$ and $\mathcal{Q}$ be a collection of query trajectories. Define the following sets:

- $C^p_{\mathcal{Q}} = \{T_i \in \mathcal{T} : d_H(\sigma(Q), \sigma(T_i)) \leq r \wedge d(T_i, Q) \leq cR \wedge Q \in \mathcal{Q}\}$
- $C^n_{\mathcal{Q}} = \{T_i \in \mathcal{T} : d_H(\sigma(Q), \sigma(T_i)) > r \wedge d(T_i, Q) > cR \wedge Q \in \mathcal{Q}\}$
- $I^p_{\mathcal{Q}} = \{T_i \in \mathcal{T} : d_H(\sigma(Q), \sigma(T_i)) \leq r \wedge d(T_i, Q) > cR \wedge Q \in \mathcal{Q}\}$
- $I^n_{\mathcal{Q}} = \{T_i \in \mathcal{T} : d_H(\sigma(Q), \sigma(T_i)) > r \wedge d(T_i, Q) \leq cR \wedge Q \in \mathcal{Q}\}$

The **Positive Metric Mismatch Rate (PMMR)** for a collection of queries $\mathcal{Q}$ is defined as $|I^p_{\mathcal{Q}}|/(|I^p_{\mathcal{Q}}| + |C^n_{\mathcal{Q}}|)$. Similarly, the **Negative Metric Mismatch Rate (NMMR)** for a collection of queries is defined as $|I^n_{\mathcal{Q}}|/(|I^n_{\mathcal{Q}}| + |C^p_{\mathcal{Q}}|)$.

3.2 Results

Space Correlation: For both Hausdroff and SketchMin the coorrelation is very high, 94.3% and 98.4%, respectively, though the standard deviation differs widely, with Hausdorff being more than 4 times that of Sketchmin. However, the correlation between Spatio-temporal distance was nearly non-existent at 17.6% and had no exploitable structure from which we could learn a function mapping between the two spaces. The results show that there is a point at which this correlation is maximized and a range in which it remains within the 90-th percentile, but the appropriate radii seems mildly dependent on

the dataset itself.

Metric Mismatch Rate of the Pipeline: We show that the PMMR is between 0 and 12% for Hausdorff and between 0 and 3% for SketchMin. This increases by roughly 1% when we involve the filter, where the increase is due to compression function used in its construction. The NMMR is negligible maxing out at 0.08%. Though not zero, such a low negative metric mismatch rate makes the data structure report similarity queries with high degree of accuracy which closely imitates the main feature of a Bloom Filter. Also, we show that DSBF of [18] actually decreases the NMMR. This is due to the Las Vegas guarantee of the filter which ensures that if signatures of binary vectors when mapped by the compression function are within a certain threshold then the Hamming distance between these binary vectors is with r. Lastly, the results indicate that the pairing of Hamming distance and Sketchmin provides the lowest PMMR and would provide the best overall performance if used in a two-tier system to reduce the processing time in very large trajectory databases where the underlying metric of interest is SketchMin.

Space Saving: To provide the possibility of constructing a two-tiered system with a succinct representation of the trajectories on RAM and a nearest-neighbor data structure on SSD, we need to ensure that the filter is indeed light-weight and reduces the space complexity of storing the trajectories outright. For both datasets we set $m = 500$ which corresponds to the number of rows in random matrix used in the construction of the individual signatures stored in the filter. This produce m-dimensional bit vectors. The total size of the filter storing $10,000$ is only 625 kilobytes. For the T-Drive dataset, at 216 megabytes, the space savings is 99.7%. Similarly, for the Rome dataset, at 46.6 megabytes, the space savings is 98.6%. We can reduce the space of the filter further by lowing our value of m but we do so at the cost of increasing the PMMR.

References

[1] Thomas Dybdahl Ahle. Optimal las vegas locality sensitive data structures. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 938–949. IEEE, 2017.

[2] Thomas Dybdahl Ahle. Subsets and supermajorities: Unifying hashing-based set similarity search. *CoRR*, abs/1904.04045, 2019.

[3] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *Advances in neural information processing systems*, pages 1225–1233, 2015.

[4] Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya P. Razenshteyn. Beyond locality-sensitive hashing. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1018–1028. SIAM, 2014.

[5] Alexandr Andoni, Huy L. Nguyen, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Approximate near neighbors for general symmetric norms. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, page 902–913, New York, NY, USA, 2017. Association for Computing Machinery.

[6] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 793–801, New York, NY, USA, 2015. Association for Computing Machinery.

[7] Boris Aronov, Omrit Filtser, Michael Horton, Matthew J. Katz, and Khadijeh Sheikhan. Efficient nearest-neighbor query and clustering of planar curves. In Zachary Friggstad, Jörg-Rüdiger Sack, and Mohammad R. Salavatipour, editors, *Algorithms and Data Structures - 16th International Symposium, WADS 2019, Edmon-*

ton, AB, Canada, August 5-7, 2019, Proceedings, volume 11646 of *Lecture Notes in Computer Science*, pages 28–42. Springer, 2019.

[8] Maria Astefanoaei, Paul Cesaretti, Panagiota Katsikouli, Mayank Goswami, and Rik Sarkar. Multi-resolution sketches and locality sensitive hashing for fast trajectory processing. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '18, page 279–288, New York, NY, USA, 2018. Association for Computing Machinery.

[9] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[10] Matteo Ceccarello, Anne Driemel, and Francesco Silvestri. FRESH: fréchet similarity with hashing. *CoRR*, abs/1809.02350, 2018.

[11] Tobias Christiani and Rasmus Pagh. Set similarity search beyond minhash. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1094–1107, 2017.

[12] Anne Driemel, Petra Mutzel, and Lutz Oettershagen. Spatio-temporal top-k similarity search for trajectories in graphs. *CoRR*, abs/2009.06778, 2020.

[13] Anne Driemel and Ioannis Psarros. Ann for time series under the fréchet distance. In Anna Lubiw and Mohammad Salavatipour, editors, *Algorithms and Data Structures*, pages 315–328, Cham, 2021. Springer International Publishing.

[14] Anne Driemel, Ioannis Psarros, and Melanie Schmidt. Sublinear data structures for short fréchet queries, 2019.

[15] Otmar Ertl. Probminhash–a class of locality-sensitive hash algorithms for the (probability) jaccard similarity. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[16] Martin Farach-Colton and Piotr Indyk. Approximate nearest neighbor algorithms for hausdorff metrics via embeddings. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 171–179. IEEE, 1999.

[17] Arnold Filtser, Omrit Filtser, and Matthew J. Katz. Approximate Nearest Neighbor for Curves - Simple, Efficient, and Deterministic. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 48:1–48:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[18] Mayank Goswami, Rasmus Pagh, Francesco Silvestri, and Johan Sivertsen. Distance sensitive bloom filters without false negatives. *CoRR*, abs/1607.05451, 2016.

[19] Piotr Indyk. Approximate nearest neighbor under edit distance via product metrics. In *SODA*, volume 4, pages 646–650, 2004.

[20] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 604–613, New York, NY, USA, 1998. Association for Computing Machinery.

[21] Jeff M. Phillips and Pingfan Tang. Simple distances for trajectories via landmarks. In Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam, editors, *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, pages 468–471. ACM, 2019.

[22] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. A survey of trajectory distance measures and performance evaluation. *The VLDB Journal*, 29(1):3–32, jan 2020.

Distance and Time Sensitive Filters for Similarity Search in Trajectory Datasets

Madhav Narayan Bhat * Paul Cesaretti† Mayank Goswami‡ Prashant Pandey§

Abstract

Similarity searching is a well-studied problem in trajectory processing, where given a query trajectory, one wants to report all trajectories in the database similar to the query. Several works have focused on developing near neighbor data structures for trajectories under various metrics, e.g., Frechet distance or Hausdorff distance. A long-standing criticism of such measures is that they ignore the temporal aspect of time-stamped trajectories, with some recent works focusing on developing near neighbor methods that incorporate some of this temporal information.

Furthermore, all near neighbor data structures use space that is super-linear (and sometimes a high-degree polynomial) in the size of the trajectory database. With trajectory datasets getting larger everyday, these data structures may not fit in RAM, and likely stored on external disk. However, in order to avoid expensive disk-accesses for every query, a lightweight filter on RAM that reports YES when there is a trajectory in the database similar to the query, and NO otherwise (most of the time), is highly desirable.

This work focuses on optimizing near neighbor data structures by addressing the two issues above. We present a filter that answers YES or NO to similarity queries quickly, incorporates both spatial and temporal aspects of the trajectories, and can be configured to fit in RAM. Our filter can be used in conjunction with any near neighbor data structure on disk to provide a two-tiered system for fast similarity searches in trajectory databases.

Our main technique involves computing sketches from a suitable-scaled 3D representation of the trajectories, followed by building distance-sensitive filters for the sketches. Our experiments first demonstrate that the distance measure thus obtained is meaningful, both spatially and temporally. Furthermore, the experiments indicate a low false negative rate, fast filtering time, and a 95% space savings on the original dataset, making these filters a natural choice in a two-tiered system to reduce processing time in very large trajectory datasets.

*Columbia University
†Graduate Center, CUNY.
‡Queens College and Graduate Center, CUNY.
§VMware Research

1 Introduction

Preprocessing trajectory datasets have several applications, ranging from social sciences to smart cities and transport. Some examples are:

1. **Traffic Analysis:** The goal is to identify flow of traffic in a given, predefined area so as to locate common bottle neck points which would suggest areas which could be improved by additional road networks to alleviate traffic congestion.
2. **Contact Tracing:** Here we are looking to identify a collection of individuals who may have been in contact with a contagious individual by returning the set all those that have visited the same areas or traveled along a similar path.
3. **Web Analytics:** Here a users web traffic consists of a sequence of hyperlinks and the goal is to find users with similar browsing habits.
4. **Weather Prediction:** Here we are concerned with forecasting the weather by being able to track fronts according to a given path of interest.

Each of these applications amounts to searching for a set of similar trajectories from a large collection of possible trajectories. The nearest-neighbor search problem (NNS) asks, given a set S of n points in a metric space, preprocess them into a data structure so as to quickly answer queries of the form "given q, report the nearest neighbor of q in S", or "given q and $r > 0$, report any point in S within distance r of q".

There exist several data structures for the NNS problem for general metric spaces (Hamming [21] [6] [4], Euclidean [21] [1] [5] [6] [4], Angular [3], Edit [20], Jaccard [16], [2], [12]), and also for metrics on the space of trajectories ([18], [7],[11], [14], [15]).

For trajectory datasets, the most commonly studied metrics are Hausdorff [17], Fréchet [15] [7], [18], and the Dynamic Time Warping metrics (DTW). Su et al. [24] provide a thorough overview of trajectory data analysis results. However, these data structures suffer from two potential limitations:

- Most NNS data structures for very large trajectory datasets will not fit in RAM, as their space usage is typically (a high) polynomial of the space used to represent the trajectories, and
- NNS data structures for Fréchet, Hausdorff, and DTW on trajectories usually treat trajectories as curves lying in the plane, and therefore ignore the spatial aspect that is usually present in many applications.

The decision version of similarity search problem can be carried out through a distance-sensitive Bloom filter. A Bloom filter [9] is a succinct data structure approximating a set to support membership queries with tunable false positive rate. It is often used to construct a two-tiered data structure which keeps a set implicitly on RAM while explicitly storing the set on SSD. As Bloom filters have only one-sided errors, having no false negatives, they save the user from making unnecessary trips to disk which incur an incredible cost in any system application. The distance-sensitive version generalizes the functionality of Bloom filter to answer queries of the form "Is x close to an element in the database?", where closeness is measured under a suitable metric. Goswami et al. [19] have the such data structure but it is limited to similarity queries in Hamming space. We aim in this work to construct a such two-tiered data for trajectory data.

1.1 Our Results

We show how to build a data structure to compute the decision version of the approximate nearest neighbor for trajectory data with an added temporal component under both the classical Hausdorff distance and recent SketchMin distance of [22] . This data structure reduces the space complexity of directly storing the trajectories outright by over 95% and has low mismatch when considering metrics like Hausdorff and SketchMin Therefore, we can construct an efficient two-tier system for performing similarity queries on trajectory data.

1.2 Preliminaries

Let A be a closed region from which a collection of trajectories are taken, and let $\mathcal{T} = \{T_1, ..., T_n\}$ denote the dataset consisting of $|\mathcal{T}| = n$ trajectories. Here we define a trajectory as follows:

Definition 1.1 (Trajectory)**.** Let $G = (V, E, c)$ be an undirected, weighted and connected graph. A trajectory $T = (v_1, t_1), ..., (v_l, t_l)$ is a sequence of pairs such that for all $1 \leq i \leq l$ the pair (v_i, t_i) consists of a vertex $v_i \in V$ and a discrete time interval $t_i = [a_i, b_i]$ where $a_i < b_i$ and $a_i, b_i \in \mathbb{Z}$.

A standard distance measure between trajectories is the Hausdorff metric which is defined as the maximum distance from a point in one trajectory to the closest point in the other trajectory. Let $T_1 = \{v_1, v_2, ..., v_{l_1}\}$ and $T_2\{v_1, v_2, ..., v_{l_2}\}$ of lengths l_1 and l_2, respectively.

Definition 1.2 (Hausdorff Distance)**.** The Hausdorff distance between T_1 and T_2 is defined as

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

Driemel et al. [13] suggest a new spatio-temporal psuedometric that can used to build an index on which one can answer the top-k query problem. Their metric is defined on trajectories over networks. The goal is to capture both temporal and spatial aspects of the trajectory such that those trajectories which are the within close proximity during the same period of time are considered close.

Definition 1.3. [13][Spatio-Temporal Metric] Let $T = ((v_1, t_1), ..., (v_l, t_l))$ and $Q = ((u_1, s_1), ..., (u_l, s_k))$ be two trajectories and s a time interval, and $d(p, q)$ be the shortest path distance between p, q on the underlying network

from which Q and T arise. The similarity of Q and T in the time interval s is defined as

$$\mathsf{Sim}(Q,T,s) = \frac{1}{|s|} \sum_{\substack{(v_i,t_i)\in T \\ (u_j,s_j)\in Q}} |s \cap t_i \cap s_j| \cdot e^{-d(v_i,u_j)}.$$

Therefore, the distance between trajectories Q and T during a time interval s is $\mathsf{Dist}(Q,T,s) = 1 - \mathsf{Sim}(Q,T,s)$. It can be computed in linear time with respect to the lengths of the trajectories.

Distances between curves are challenging to compute. For inputs of length m, many standard metrics for curves, namely Fréchet, DTW, can be computed in $O(m^2)$ time. Astefanoaei et al [8] present a suite of low-dimensional sketches for trajectory data that summarize the dataset and drastically reduce the computation costs associated with near neighbor search and distance estimation. We present the one most relevant to our work.

Definition 1.4. [8]($(d,\ r)$-Binary Sketch) The Binary Sketch of a trajectory is a 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. The distance between two such sketches is simply their Hamming distance.

These sketches can obtain provable locality sensitive hash families for both Hausdorff and Fréchet which are useful in developing NNS data structures with queries sublinear in the size of the dataset, and show how the sketches can be used in a novel data structure for answering similarity queries. Additionally, Phillips et al. [22] and [23] develop a new class of distances for objects including lines, hyperplanes and trajectories based off sketches derived from a collection of landmarks. A sketch of a geometric object is defined as

Definition 1.5. [22] Let $Q \subseteq \mathbb{R}^d$ be a collection of landmarks, where $|Q| = n$. For a geometric object $J \subset R^d$, its sketch representation $v_Q(J) \in \mathbb{R}^n$ is a vector whose i-th coordinate is

$$v_i(J) = \inf_{p\in J} \|p - q_i\|,$$

where $q_i \in Q$.

The distance between two sketches is referred to as the SketchMin distance and is the Euclidean distance between them.

Definition 1.6. [22](SketchMin Distance) Let $J_1, J_2 \subset \mathbb{R}^d$ be two geometric objects. The SketchMin distance between them their sketch representations is

$$d_Q(J_1, J_2) = \|v_Q(J_1) - v_Q(J_2)\|.$$

When the geometric objects are trajectories, this metric matches or out-performs all other metrics in terms of computational speed, and was shown to speed up approximate nearest neighbor data structures.

2 Algorithm

The construction of our data structure follows four steps:

1. Lift each trajectory to 3D by taking into account its temporal component.
2. Scale the time component, possibly incorporating a user-defined parameter indicating the importance of the temporal aspect.
3. Construct a binary sketch representation of each trajectory.
4. Store the binary sketches in a distance-sensitive Bloom filter data structure of Goswami et al. [19].

Step 1: Lifting Trajectories. Lift each trajectory $T = (v_1, t_1), ..., (v_l, t_l)$ to 3D by placing the first component in the xy-plane and the time in which the a trajectory arrives at this location on the z-axis.

Step 2: Scaling the Time Component Let s be the average speed of the trajectories in the dataset.

Also, let $w \in [0,1]$ denote a user defined parameter indicating the importance of the temporal aspect in the application. $w = 0$ corresponds to the temporal aspect being irrelevant, and the trajectory being just a 2D curve, as is treated in most of the literature. On the other hand, $w = 1$ corresponds to the temporal aspect being as important as the spatial aspect.

For each $T_i \in \mathcal{T}$ we scale its time component by sw; that is swt_i for all $i \in [n]$. For the remainder of this article we will assume $w = 1$.

Our time scaling means that one unit of time corresponds to the average speed s of the trajectories in $\mathcal{T}$. Consider points $p = (x, y, t), q = (x', y', t)$ and $r = (x, y, t + 1)$, where $d(p, q) = s$. Here p and q are equal temporally but are a distance s away. Similarly, p and r occupy the same space but at one unit time difference; see figure 1.

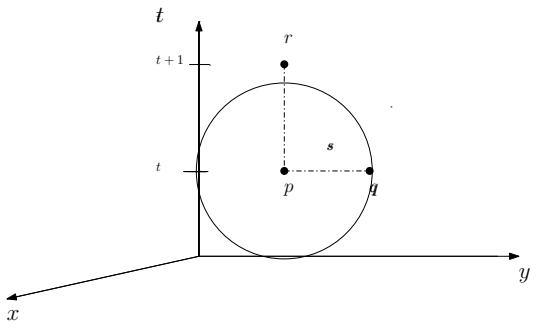

Figure 1: Scaling the Temporal Component

We would like the scaling to be such that these distances are the same; that is, $d(p, q) = d(p, r) = s$, where $d(.,.)$ is the standard Euclidean norm between two points.

Step 3: Computing Binary Sketches. Deploy and store a set of d random spheres in 3D $S = \{s_1, ..., s_d\}$, and for each trajectory, test for its intersection with these spheres; see Figure 1. The sketch $\sigma(T_j)$ for trajectory $T_j \in \mathcal{T}$ is now a d-dimensional bit vector $b_j[1, ..., d]$ where the i-th entry $b_j[i] = 1$ if and only if T_j contains point $p_k = (v_k, t_k)$ such that $p_k \in s_h$, for some $1 \leq k \leq l$ and some $1 \leq h \leq d$.

Step 4: Constructing Filter.

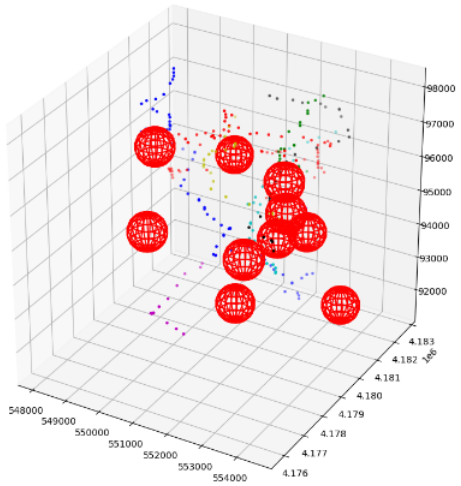

Figure 2: Trajectory sketches with respect to its intersection with 10 randomly deployed spheres.

Given two sketches $\sigma(T_i)$ and $\sigma(T_j)$ for trajectories $T_i, T_j \in \mathcal{T}$, a simple measure of distance between them is the Hamming distance $d_H(\sigma(T_i), \sigma(T_j))$, which is defined as the number of positions in which b_i and b_j differ. As was noted in [8], when spheres are deployed randomly, similar trajectories will likely intersect a similar set of spheres which will make their respective sketches similar.

We now store this collection of binary signatures in the (r, c, ε) distance-sensitive Bloom filter data structure by Goswami et al. [19].

Such a data structure stores a set $S \subset \{0,1\}^d$ of points from d-dimensional Hamming space, such that given a query $q \in \{0,1\}^d$, the data structure reports:

- YES, if $q \in \{x \in \{0,1\}^d : D(x, S) \leq r\}$.
- NO, if $q \in \{x \in \{0,1\}^d : D(x, S) > cr\}$, with probability at least $1 - \varepsilon$.

where $D(q, S) = \min_{p \in S} d(q, p)$ is the minimum Hamming distance between q and any point in S. The following guarantee is provided:

Theorem 2.1. *[19] There exists a (r, c, ε)-distance sensitive approximate membership filter with pointwise error which requires*

$$O\Big(n\Big(\frac{r}{(c-1)} + \Big(\frac{c}{c-1}\Big)^2 \log(\frac{n}{\varepsilon})\Big)\Big)$$

bits for any $c > 1$ on a set S of n points. When $c \geq 2$, the filter uses $O(n(\frac{r}{c} + \log(\frac{n}{\varepsilon})))$ bits, and it is optimal if $r/c \leq \log(n\varepsilon)$ or $\varepsilon \leq 1/n^{1+o(1)}$.

Query: Given a query trajectory Q and user defined similarity range of interest $0 \leq r \leq d$, compute its sketch $\sigma(Q)$ by testing its intersection with the set of spheres S. Now given its d-dimensional bit representation b_Q, feed it to the (r, c, ε)-DSBF D. If D answers YES, then return YES, otherwise return NO.

3 Experiments

In this section, we evaluate the efficacy of our data structure. We are interested in answering the following questions:

- What is the correlation between our sketches with other metric spaces now that our trajectories have an additional temporal component?
- What is the relationship between the average Hamming weight for the sketches of the trajectories to the correlation between different metric spaces?
- How accurate are the sketches in labeling similar trajectories according to some other underlying space of interest?
- How does the filter affect this labeling and can it still be used to answer similarity queries without too many mismatches?
- What is the space savings that the filter provides?

3.1 Dataset

For the evaluation of the data structure, we used the following datasets:

- The CRAWDAD (Rome) dataset [10] contains the GPS locations of 320 taxis working in the city of Rome, Italy for one month in 2014. Each trace represents one driver over the course of a single day and is collected every 7 seconds on tablet which sends this data to a central server, giving roughly 22 million data points.
- The T-Drive dataset [25] contains the GPS locations of 10,357 taxis in Beijing. The average sampling interval is 177 seconds with a distance of 623 meters, giving 15 million data points in total.

3.2 Dataset Preprocessing

We constructed $10,000$ trajectories from each dataset which would be stored and 1000 trajectories that would be used as queries. These trajectories were constructed from a given region of the dataset. The dimensions of the region from which the trajectories were taken was to ensure a diverse set of trajectories by selecting from a relatively concentrated area. For the queries, we selected trajectories from an overlapping, larger region to diversify the query set. These dimensions are summarized in Table 1.

Name	$\lvert D\rvert$	Dimensions of Bounding Box
Rome	10,000	$50km \times 40km \times 4hr$
T-Drive	10,000	$7.5km \times 4km \times 3hr$
Rome Queries	1,000	$20km \times 20km \times 5hr$
T-Drive Queries	1,000	$12km \times 8km \times 5hr$

Table 1: Region from which the trajectories are taken

Each dataset is preprocessed to make the movement of the trajectories more standard and to remove much of the inherent noise. Points lying outside the bounding box for its dataset are ignored or not included in the description of the trajectory. Similarly, we limit or completely remove those points in a trajectory that stay stationary. This limit is set to include five consecutive stationary points. Lastly, we break a trajectory into two trajectories if it contains consecutive points that are over five minutes apart or greater than $1km$ in distance. This gives trajectories with lengths of roughly $24km$ on average for the Beijing T-Drive

dataset and query set, where the average number of GPS points per trajectory is 48. For the Rome taxi cab dataset the average length is smaller at $13km$, but the average number of GPS points per trajectory is considerably larger at 224.

3.3 Experimental Setup

Experiments were implemented using Python 3.8.8 and ran on a Windows 10 operating system equipped with 32 GB @32000 MHz of RAM and an Intel core i9-9900k CPU @ 3.60 GHz eight-core processor. The source code are online and available at `https://github.com/pcesaretti/trajectory_approximate_nn_filter`.

3.4 Overview

Here we will give an overview of the remainder of the section. The experiments were broken into two phases. In phase 1 we explore the correlation between Hamming distance and three other distance measures, namely that of Hausdorff, SketchMin of [22], and the Spatio-Tempral distance of [13]. If the correlation is sufficiently high to establish a link between the two spaces, we investigate affect of the number of spheres and size of radius on the correlation between Hamming space and the metric in question. We additionally look into the affect the Hamming weight of each vector has on this space correlation. Phase 1 determines the parameter values, namely the number of spheres and the size of a radius, which will remain constant through the second set of experiments. In phase II we look into the rate of mismatch between the two spaces in correctly labeling trajectories as being within a user prescribed range before and after they are stored in the filter. Lastly, we look at the space savings of using the filter.

3.5 Defining Testing Measures

The metric of interest in analyzing our data structure is the level of mismatch in labeling between the two spaces; that is, though the correlation could be high, there is still the chance that important information about the trajectory is not maintained by its binary sketch representation and hence measurements made within the Hamming space may erroneously imply a positive (negative) response to a given sketch of a query trajectory.

Such errors are normally referred to as false positives and false negatives, but this is not a fair description as this is not a error of the data structure per say but is a problem with approximating between the two metric spaces. We define such events to be a positive (negative) metric mismatch. Here is the formal definition of both:

Definition 3.1. A **Positive Metric Mismatch (PMM)** for a metric space $M = (d, X)$ on query trajectory Q occurs if for $r, R, c > 0$ there is a signature for a trajectory T such that $d_H(\sigma(T), \sigma(Q)) \leq r$ but for all trajectories T' in our dataset $d(T', Q) > cR$.

Definition 3.2. A **Negative Metric Mismatch (NMM)** for a metric space $M = (d, X)$ on query trajectory Q occurs if for $r, R, c > 0$, on all signatures $\sigma(T)$, $d_H(\sigma(T), \sigma(Q)) > r$ but there exists a trajectory T' in our dataset such that $d(T', Q) \leq cR$.

The efficacy of our data structure is dependant on how often these events arise and whether or not they can be reduced. Hence, we are concerned with the rate at which this occurs over a collection of queries.

Definition 3.3. Let $r, R \geq 0$, $c > 0$ and $\mathcal{Q}$ be a collection of query trajectories. Define the following sets:

- $C^p_{\mathcal{Q}} = \{T_i \in \mathcal{T} : d_H(\sigma(Q), \sigma(T_i)) \leq r \wedge d(T_i, Q) \leq cR \wedge Q \in \mathcal{Q}\}$
- $C^n_{\mathcal{Q}} = \{T_i \in \mathcal{T} : d_H(\sigma(Q), \sigma(T_i)) > r \wedge d(T_i, Q) > cR \wedge Q \in \mathcal{Q}\}$
- $I^p_{\mathcal{Q}} = \{T_i \in \mathcal{T} : d_H(\sigma(Q), \sigma(T_i)) \leq r \wedge d(T_i, Q) > cR \wedge Q \in \mathcal{Q}\}$
- $I^n_{\mathcal{Q}} = \{T_i \in \mathcal{T} : d_H(\sigma(Q), \sigma(T_i)) > r \wedge d(T_i, Q) \leq cR \wedge Q \in \mathcal{Q}\}$

The **Positive Metric Mismatch Rate (PMMR)** for a collection of queries $\mathcal{Q}$ is defined as $|I^p_{\mathcal{Q}}|/(|I^p_{\mathcal{Q}}| + |C^n_{\mathcal{Q}}|)$. Similarly, the **Negative Metric Mismatch Rate (NMMR)** for a collection of queries is defined as $|I^n_{\mathcal{Q}}|/(|I^n_{\mathcal{Q}}| + |C^p_{\mathcal{Q}}|)$.

3.6 Phase 1: Space Correlation

Our first set of experiments investigate the the correlation between Hamming space and three metrics of interest: Haussdorff, SketchMin and Spatio-temporal distance of [13]. For each pair of trajectories we construct a two-dimensional vector whose first coordinate is the Hamming distance between their respective signatures and whose second coordinate is their distance under the metric in question and compute the Pearson coefficient. As in [8], we grouped the possible values in 30 bins and show in the plots the median (blue line) value as well as the 5-95 (pink), 10-90 (purple), and 25-75 (grey) percentiles. For both Hausdroff and SketchMin the coorrelation is very high, 94.3% and 98.4%, respectively, though the standard deviation differs widely, with Hausdorff being more than 4 times that of Sketchmin. However, correlation between Spatio-temporal distance was nearly non-existent at 17.6% and had no exploitable structure from which we could learn a function mapping between the two spaces. Figure 3. shows these results on the Rome dataset; results hold similarly for the T-drive dataset.

Effect of the Number of Sphere and Radius on the correlation: The above result raises the question on whether the correlation can be tuned by the number and radii of the spheres. For a fixed number of spheres, we varied the radii between $500m$ to $30km$ and computed the Pearson correlation coefficient; this is reflected in each subplot. The results show that there is a point at which this correlation is maximized and a range in which it remains within the 90-th percentile. Both Hausdorff and SketchMin maximize their correlation with Hamming space with 500 spheres, but the appropriate radii seems mildly dependent on the dataset itself. For the Rome dataset we have a radius of $22km$ for both SketchMin and Hausdorff but for the T-drive dataset you need a larger radius of $30km$. As noted in [8], increasing the number of spheres beyond a certain number does not improve the correlation and that for a good correlation we need to choose enough spheres that cover the space from which the trajectories are sampled. The variance involved is more stable and consistent for SketchMin than for Hausdorff, where for $50-500$ spheres we can achieve 90% correlation for any radius between $18km-26km$.

Hamming Weight and Correlation: Additionally, we checked how the correlation could be affected by the average Hamming weights of our binary sketches. Intuitively, if the average Hamming weight of our binary sketches is too high then we loose the ability to properly label trajectories that are within a certain threshold in some underlying metric space which would in turn skew the correlation between the two spaces. Figure 4e shows that for both spaces maintaining sketches whose average Hamming weight is between $30-60\%$ suffices to maintain a high correlation, and any higher our correlation degrades.

3.7 Phase 2: Mismatch Error

For our sketches to be an effective, representative stand-in for trajectory data such that we can perform distance queries, we require them to be able to capture the most important features of a trajectory while being diverse enough appropriately distinguish between trajectories that are within a certain threshold according to some underlying distance metric. The first collection of experiments show correlation between Hamming space and two other spaces (SketchMin and Hausdorff) is high enough to learn a function mapping distances in one space to the distances the other. Given this, our second set of experiments investigate the efficacy of the data structure as a means of being used as a light-weight filter to perform similarity queries in these spaces. To this end we look at the rate at which the filter incorrectly returns a response of YES when the answer is NO and visa versa.

Constant Parameters: For experiments concerning the Rome dataset, we set the number of spheres $n = 500$ and the radius per sphere $R_s = 22000$, which were values at which the correlation between Hamming and the two spaces (Hausdorff and SketchMin) is highest. Similarly, for the T-Drive dataset we keep $n = 500$ but set $R_s = 30000$. The parameters of the distance sensitive filter are set to $\varepsilon = 0.1$, $c_{mod} = 2$,

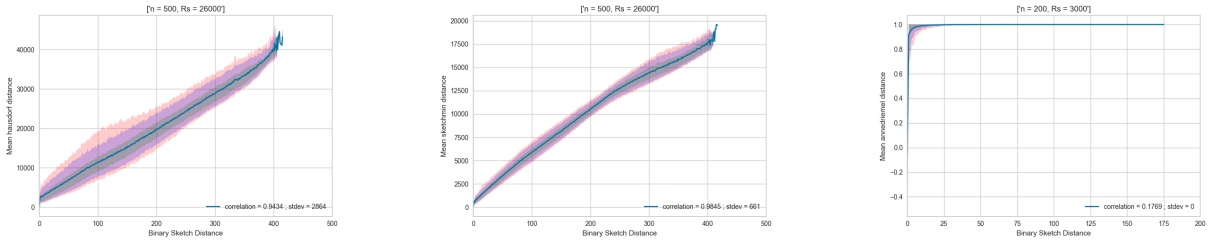

Figure 3: Correlation of binary distances with Hausdorff, Sketchming, and the Spatio-Temporal Distance of [13] on the Rome dataset

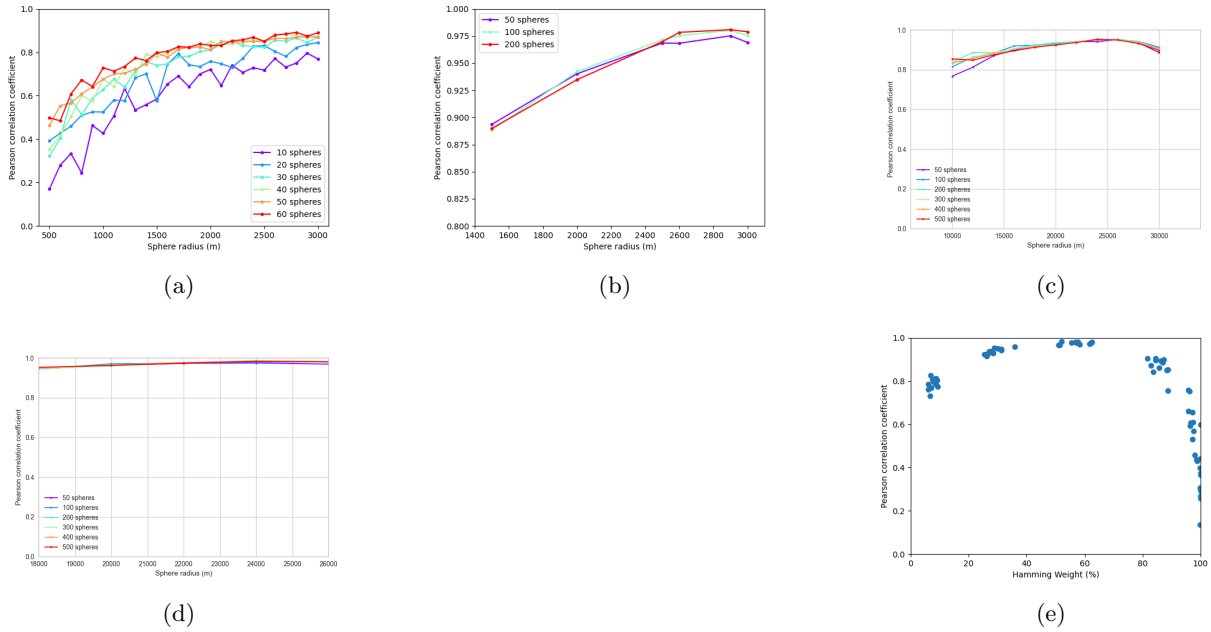

Figure 4: (a) Correlation coefficient vs number of spheres and radii for Hausdorff on 1000 trajectories of the Rome dataset from a smaller region. (b) Correlation coefficient vs number of spheres and radii for SketchMin on 1000 trajectories of the Rome dataset from a smaller region. (c) Region of high correlation for Hausdorff on all 10000 trajectories of the T-drive dataset. (d) Consistently high correlation and limited variance for Sketchmin for $50 - 500$ spheres. (d) Hamming weight affect on the correlation coefficient.

$c_{div} = 1$, $\delta = 1$, $c = 2$.

Metric Mismatch Rate of the Pipeline: In this experiment we simulate the pipeline: On each trial, we process the $10,000$ trajectories and store them in a distance-sensitive Bloom filter using the parameters above. Next we use a sample of $1,000$ trajectories to compute a linear regression on the correlation between the spaces. We return the regression line f produced by averaging slopes of the line found in the previous step over 3 independent runs (see). After, we compute the signatures for $1,000$ queries and per-

form a similarity search for $r = f(R)$, where R is the metric radius of the space being tested. Similarity search is first performed on the sketches themselves by referencing a precomputed array of the minimum Hamming distance to all query trajectories, seeing if this value is within r, and determing if there exist any trajectories within the dataset that are within cR to the query trajectory. We then query the filter and check the validity of its response in the same way. Using this information we compute positive (negative) mismatch rate incurred and average over 10 trials each with a new set of random spheres. We test over $R \in [200, 6000]$, where $R = 6000$ is the maximum distance between trajectories in both Hausdorff and SketchMin.

Figure 5 shows the PMMR and NMMR of the sketches. We see (Figures 5b and 5d) that the PMMR is between 0 and 12% for Hausdorff and between 0 and 3% for SketchMin. This increases by roughly 1% when we involve the filter, where increase is due to compression function used in its construction. The NMMR is negligible maxing out at 0.08%. Though not zero, such a low negative metric mismatch rate makes the data structure report similarity queries with high degree of accuracy which closely imitates the main feature of a Bloom Filter. Also, figures 5a and 5c show that DSBF of [19] actually decreases the NMMR. This is due to the Las Vegas guarantee of the filter which ensures that if signatures of binary vectors when mapped by the compression function are within a certain threshold then the Hamming distance between these binary vectors is with r. Lastly, the above results indicate that the pairing of Hamming distance and Sketchmin provides the lowest PMMR and would provide the best overall performance if used in a two-tier system to reduce the processing time in very large trajectory databases where the underlying metric of interest is SketchMin.

3.8 Space Savings

To provide the possibility of constructing a two-tiered system with a succinct representation of the trajectories on RAM and a nearest-neighbor data structure on SSD, we need to ensure that the filter is indeed light-weight and reduces the space complexity of storing the trajectories outright. For both datasets we set $m = 500$ which corresponds to the number of rows in random matrix used in the construction of the individual signatures stored in the filter. This produce m-dimensional bit vectors. The total size of the filter storing $10,000$ is only 625 kilobytes. For the T-Drive dataset, at 216 megabytes, the space savings is 99.7%. Similarly, for the Rome dataset, at 46.6 megabytes, the space savings is 98.6%. We can reduce the space of the filter further by lowing our value of m but we do so at the cost of increasing the PMMR. These results are summarized in Table 2.

Dataset	Dataset Size	Filter Size	Space Savings
T-Dirve	216 MB	625 KB	99.7%
Rome	46.6 MB	625 KB	98.6%

Table 2: Space Savings of the Filter Storing the Trajectories

4 Conclusion

The results show that our main techniques of computing sketches from a suitable-scaled 3D representation of trajectories provides a meaningful substitute trajectory data which can used to build a distance-sensitive filter with a low to negligible false negative rate and manageable false positive rate. Furthermore, we show a link between the Hamming space and two other metrics, namely Hausdorff and SkethcMin, which allows the possibility of constructing a two-tier system to perform similarity queries on a large dataset stored on SSD.

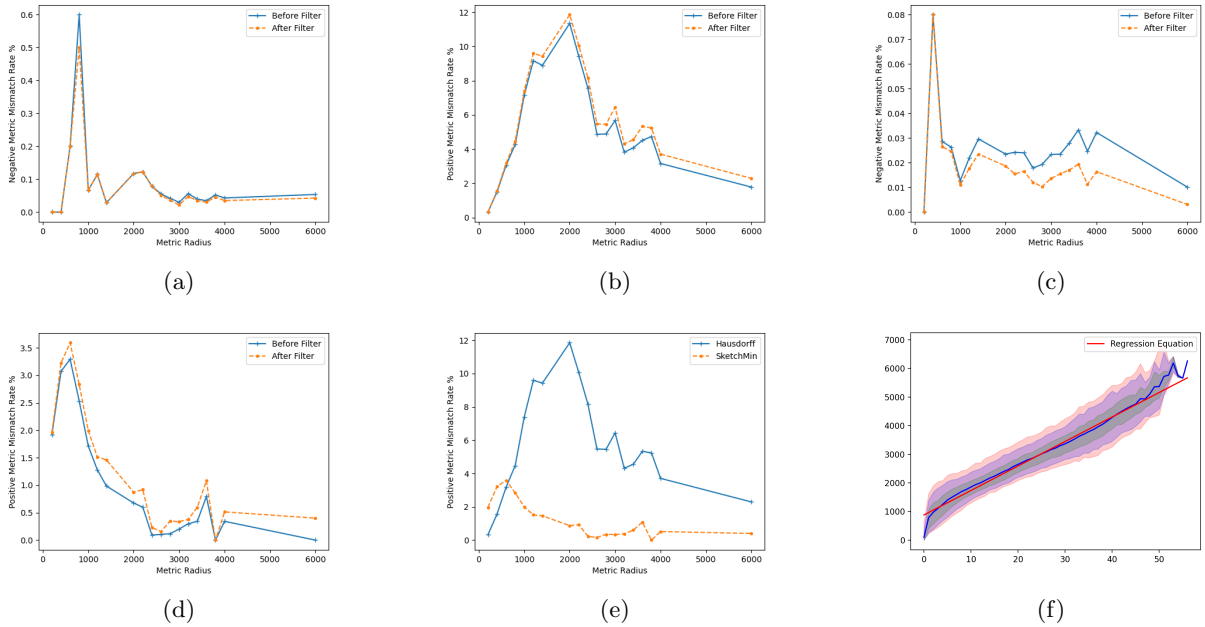

Figure 5: (a) T-Drive - NMMR for Hausdorff of the before and after the filter. (b) T-Drive - PMMR for Haosdorff before and after the filter. (c) T-Drive - NMMR for SketchMin of the before and after the filter. (d) T-Drive - PMMR for SketchMin of the before and after the filter. (e) Comparison of the PMMR rate between Hausdorff and SketchMin. (f) The computed regression line which we need to find the appropriate value for r for the specified space.

References

[1] Thomas Dybdahl Ahle. Optimal las vegas locality sensitive data structures. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 938–949. IEEE, 2017.

[2] Thomas Dybdahl Ahle. Subsets and supermajorities: Unifying hashing-based set similarity search. *CoRR*, abs/1904.04045, 2019.

[3] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *Advances in neural information processing systems*, pages 1225–1233, 2015.

[4] Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya P. Razenshteyn. Beyond locality-sensitive hashing. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1018–1028. SIAM, 2014.

[5] Alexandr Andoni, Huy L. Nguyen, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Approximate near neighbors for general symmetric norms. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, page 902–913, New York, NY, USA, 2017. Association for Computing Machinery.

[6] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-*

Seventh Annual ACM Symposium on Theory of Computing, STOC '15, page 793–801, New York, NY, USA, 2015. Association for Computing Machinery.

[7] Boris Aronov, Omrit Filtser, Michael Horton, Matthew J. Katz, and Khadijeh Sheikhan. Efficient nearest-neighbor query and clustering of planar curves. In Zachary Friggstad, Jörg-Rüdiger Sack, and Mohammad R. Salavatipour, editors, *Algorithms and Data Structures - 16th International Symposium, WADS 2019, Edmonton, AB, Canada, August 5-7, 2019, Proceedings*, volume 11646 of *Lecture Notes in Computer Science*, pages 28–42. Springer, 2019.

[8] Maria Astefanoaei, Paul Cesaretti, Panagiota Katsikouli, Mayank Goswami, and Rik Sarkar. Multi-resolution sketches and locality sensitive hashing for fast trajectory processing. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '18, page 279–288, New York, NY, USA, 2018. Association for Computing Machinery.

[9] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[10] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from `https://crawdad.org/roma/taxi/20140717`, July 2014.

[11] Matteo Ceccarello, Anne Driemel, and Francesco Silvestri. FRESH: fréchet similarity with hashing. *CoRR*, abs/1809.02350, 2018.

[12] Tobias Christiani and Rasmus Pagh. Set similarity search beyond minhash. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1094–1107, 2017.

[13] Anne Driemel, Petra Mutzel, and Lutz Oettershagen. Spatio-temporal top-k similarity search for trajectories in graphs. *CoRR*, abs/2009.06778, 2020.

[14] Anne Driemel and Ioannis Psarros. Ann for time series under the fréchet distance. In Anna Lubiw and Mohammad Salavatipour, editors, *Algorithms and Data Structures*, pages 315–328, Cham, 2021. Springer International Publishing.

[15] Anne Driemel, Ioannis Psarros, and Melanie Schmidt. Sublinear data structures for short fréchet queries, 2019.

[16] Otmar Ertl. Probminhash–a class of locality-sensitive hash algorithms for the (probability) jaccard similarity. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[17] Martin Farach-Colton and Piotr Indyk. Approximate nearest neighbor algorithms for hausdorff metrics via embeddings. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 171–179. IEEE, 1999.

[18] Arnold Filtser, Omrit Filtser, and Matthew J. Katz. Approximate Nearest Neighbor for Curves - Simple, Efficient, and Deterministic. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 48:1–48:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[19] Mayank Goswami, Rasmus Pagh, Francesco Silvestri, and Johan Sivertsen. Distance sensitive bloom filters without false negatives. *CoRR*, abs/1607.05451, 2016.

[20] Piotr Indyk. Approximate nearest neighbor under edit distance via product metrics. In *SODA*, volume 4, pages 646–650, 2004.

[21] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 604–613, New York, NY, USA, 1998. Association for Computing Machinery.

[22] Jeff M. Phillips and Pingfan Tang. Simple distances for trajectories via landmarks. In Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam, editors, *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, pages 468–471. ACM, 2019.

[23] Jeff M. Phillips and Pingfan Tang. Sketched mindist. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry, SoCG 2020, June 23-26, 2020, Zürich, Switzerland*, volume 164 of *LIPIcs*, pages 63:1–63:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[24] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. A survey of trajectory distance measures and performance evaluation. *The VLDB Journal*, 29(1):3–32, jan 2020.

[25] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 316–324, New York, NY, USA, 2011. Association for Computing Machinery.