Topological signatures for fast mobility analysis

Citation for published version:

Digital Object Identifier (DOI):
10.1145/3274895.3274952

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Topological Signatures For Fast Mobility Analysis

Abhirup Ghosh, Benedek Rozemberczki, † Subramanian Ramamoorthy, Rik Sarkar
School of Informatics, University of Edinburgh, U.K.
abhirup.ghosh@ed.ac.uk, benedek.rozemberczki@ed.ac.uk, s.ramamoorthy@ed.ac.uk, rsarkar@inf.ed.ac.uk
† FiveAI Ltd. Edinburgh, U.K. s.ramamoorthy@five.ai

ABSTRACT
Analytic methods can be difficult to build and costly to train for mobility data. We show that information about the topology of the space and how mobile objects navigate the obstacles can be used to extract insights about mobility at larger distance scales. The main contribution of this paper is a topological signature that maps each trajectory to a relatively low dimensional Euclidean space, so that now they are amenable to standard analytic techniques. Data mining tasks: nearest neighbor search with locality sensitive hashing, clustering, regression, etc., work more efficiently in this signature space. We define the problem of mobility prediction at different distance scales, and show that with the signatures simple & nearest neighbor based regression perform accurate prediction. Experiments on multiple real datasets show that the framework using topological signatures is accurate on all tasks, and substantially more efficient than machine learning applied to raw data. Theoretical results show that the signatures contain enough topological information to reconstruct non-self-intersecting trajectories up to homotopy type. The construction of signatures is based on a differential form that can be generated in a distributed setting using local communication, and a signature can be locally and inexpensively updated and communicated by a mobile agent.

CCS CONCEPTS
• Mathematics of computing → Algebraic topology; • Information systems → Location based services; Geographic information systems; Sensor networks; • Theory of computation → Nearest neighbor algorithms; • Computing methodologies → Machine learning approaches;

ACM Reference Format:

1 INTRODUCTION
Location traces are computationally challenging to process. Distances between trajectories like Fréchet and Hausdorff distances [3] do not form a normed space, Dynamic time warping distance [28] does not even form a metric [14]. As a result, typical computational methods are difficult to apply. Specialized techniques have been developed for location traces. Theoretical approaches such as locality sensitive hashing have to be tailored to trajectories for faster nearest neighbor search [4, 14, 20]; computation of medians [7], popular paths [21] etc have also been considered.

Analysis and prediction of motion have been developed with Markov models [24, 34, 37] that assume motion to depend only on the current “state” of an agent and not on longer history. However, recent experimental results show that the memoryless assumption of Markov models does not hold in real location data, since agents do not execute a random walk, but move with the purpose to reach a destination [31]. Specialized neural network based models [25, 33] with the power to encode longer history are shown in practice to be effective for next road segment prediction, although they do not provide rigorous models for other general tasks, e.g., search and comparison of trajectories. We instead seek a framework for interpretable encoding of large scale properties of trajectories, that is compatible with a range of analytic methods.

The motion of objects is determined by the major obstacles in the plane, which defines the geometry and topology of the space. However, considerations of topology raise complex issues, and traditional quantities like Fréchet distance are prohibitively expensive to compute in domains with obstacles [9]. Geometric algorithms thus usually assume motion in a simple plane, or of some topological type.

In past work, Hodge theory and relative homology from algebraic topology have been used for the purpose of topological classification [27, 35]. However, these approaches are only applicable to specific start and end points of trajectories, and yield only discrete classifications by homotopy type, which is not general enough for use in further analysis. They are also expensive to compute. We intend to overcome these obstacles by using more flexible topological constructs.

Our contributions. This paper shows that topological information can enhance mobility analysis by meaningfully encoding large scale patterns of movement, and demonstrates how the relevant information can be represented compactly.

The geometry and topology of the space of mobility itself can be discretized as a graph, where regions of zero or little activity are treated as obstacles or holes, and the remaining space is triangulated based on a suitable set of vertices.

Let us define a discrete differential form $\xi : E \rightarrow \mathbb{R}$ as a function that assigns values to directed edges. For any directed edge $ab$, the function $\xi$ satisfies $\xi(ab) = -\xi(ba)$, that is, the values on an edge and its reverse are negations of each other (See Section 2).
Further, the values on the edges sum to zero around any face of the graph, except possibly the obstacles. For each obstacle $i$, we define a differential form $\xi_i$, such that around obstacle $i$, $\xi_i$ on the edges sum to 1, and thus preserves topological information with respect to this obstacle.

For any mobility trace $A$, the differential form values along its directed edges are added to obtain $\xi_i(A)$. For a trajectory that goes around an obstacle $i$, $\xi_i(A) = 1$. In a space with $k$ obstacles, the general differential form $\xi = [\xi_1, \xi_2, \ldots, \xi_k]$ is a vector of length $k$.

The significant property here is that a signature element $\xi_i(A)$ can have a real numbered value when $A$ is not a perfect loop. The signature $\xi_i(A) \in \mathbb{R}^k$ is a point in $k$ dimensional space. Thus, beyond simple topological equality, the signatures can represent topological similarity. The mapping to a normed space enables standard analytic techniques like clustering, regression, density estimation, etc. We discuss the basic construction in Section 3 and the detailed technique in Section 4.

Summations over differential forms preserve only large scale properties, and are insensitive to noise that does not change topological type of a trajectory. As a result, the signatures are invariant to localization errors and GPS noise. They are compact compared to localization errors and GPS noise. They are compact compared to the nearest neighbor regression on the signatures performs comparably to expensive and opaque neural network based methods, which suggests that the topological features are essential to understanding and prediction of motion.

Data mining techniques of locality sensitive hashing, dimension reduction, etc run efficiently and accurately on the signatures (Section 5). Experiments (Section 6) show that the dimensionality of the signatures can be reduced by selecting only a few obstacles from the domain, and this further increases the efficiency of the system. Based on how the trajectories traverse the domains, the obstacles themselves can be characterized by how they influence mobility.

Section 7 discusses some related works and how the new approach compares with them.

Before we move on with technical details, let us summarize our approach into the following steps: 1. Find obstacle as sparse regions of the plane 2. construct a differential 1-form for each obstacle. 3. As a mobile agent moves, use these forms to track how it circumnavigates each obstacle and get a point in Euclidean space, where each dimension represents behavior with respect to one obstacle. 4. Apply efficient analytic techniques treating trajectories as points in Euclidean space.

2 THEORETICAL BACKGROUND

Discrete differential forms and exterior calculus are a varied topic of study across many disciplines. This section presents a brief description of the main ideas relevant. A more formal treatment can be found in [19]. Readers familiar with the topic can consider skipping ahead to the next section.

Let us represent the space of motion using a simplicial complex or a cell complex. A two dimensional simplicial complex is a planar graph $G$ with 0, 1, and 2 dimensional simplices as vertices, edges, and triangles as shown in Figure 1(a). A simplicial complex is built by attaching simplices together along subsimplices.

Orientation of a simplex is defined using ordering of constituent lower dimension simplices. E.g. ordering of two 0–simplices $[v_0, v_1]$ produces an oriented 1–simplex or a directed edge, $e$, while its opposite orientation is $-[v_0, v_1]$ or $-e$.

A linear combination of $d$ dimensional simplices: $\sum_{i=0}^{k} \lambda_i \sigma_i$, where each $\lambda_i$ is an integer, is called a $d$-chain. A trajectory or mobility trace can be seen as 1-chain.

The boundary ($\partial \sigma$) of an oriented $p$–chain $\sigma$ is a $(p - 1)$–chain that separates $\sigma$ from rest of the complex and the orientation of the boundary is inherited from $\sigma$. Planar graphs are orientable, meaning that all its simplices can be oriented consistently [23]. Here, all 2–simplices or faces are assumed to be oriented counter-clockwise. Consequently, the boundary of a summation of faces (2-chains) is the sum of their boundaries (1-chains); formally: $\partial(f_1 + f_2 + f_3 + \ldots) = \partial f_1 + \partial f_2 + \partial f_3 + \ldots$ (Figure 1(b)).

Figure 1: (a) 0, 1, and 2–simplices (b) Boundary operation on a chain of 2-simplices

Cell complexes are a natural extension of simplicial complexes. The only difference meaningful to this context is that a two dimensional cell, or face, can be a simple polygon instead of a triangle, otherwise a cell complex retains all relevant properties of a simplicial complex. The area external to the planar graph is called the exterior face, or the face at infinity. The edges at the boundary of this face constitute the exterior boundary.

Dual complex. The dual of a simplicial complex $G$ is a planar graph $\star G$ where a face $\sigma$ in $G$ corresponds to a node $\star \sigma$ in $\star G$ and an edge $\star e$ in $\star G$ exists between two nodes if the corresponding two faces in $G$ are adjacent with shared edge $e$. An example is shown in Figure 2(a).

2.1 Discrete differential 1-forms and co-chains

Differential forms are defined as functions over the directed cells of the cell complex. Specifically, 1-forms are values for the edge set
E of a graph as $\omega : E \rightarrow \mathbb{R}$. The values are associated with directed edges, so that $\omega(ab) = -\omega(ba)$. Given a discrete trace written as $e_1 + e_2 + e_3 \ldots$, the integral of the form over it is now computed as sum of $\omega$ over the directed edges: $\omega(e_1) + \omega(e_2) + \omega(e_3) \ldots$.

The function $\omega$ thus defines a co-chain in $\mathcal{C}^1(G; \mathbb{R})$ that maps 1-chains to real values. The entire construct translates to a dual with a dual function $\star \omega$ defined simply as $\star \omega(\star e) = \omega(e)$. Thus $\star \omega$, yields a differential form $\omega$, which will be useful in the following section.

**Sources and sinks.** Suppose $X(v)$ is the set of edges of the type $(v, e)$ – that is, all edges incident on $v$, with orientation selected as pointing outward from $v$. Then we can compute the net outflow from $v$ as $h(v) = \sum_{e \in X(v)} \omega(e)$. Depending on this value, $v$ can be a:

- **Source:** net outflow $h(v) > 0$.
- **Internal or regular node:** net outflow $h(v) = 0$.
- **Sink:** net outflow $h(v) < 0$.

Observe the effect on the dual system. For a primal face $f$ with dual vertex $\star f$, we have: $\sum_{e \in X(\star f)} \star \omega(\star e) = \omega(\partial f)$.

### 3 TOPOLOGICAL SIGNATURES

The goal is to utilize differential forms to characterize the behavior of the traces with respect to obstacles in the domain. We assume that the obstacles such as buildings and lakes reside in the larger faces of the graph and the mobile objects move along the nodes and edges of the graph. The question of what constitutes important obstacles in a domain will be taken up later, for now assume that some faces are given to us to be treated as obstacles.

**Dual sources.** Each obstacle is a source of a dual vector field $\overline{\xi}$.

More specifically, given a face $f$ with an obstacle, the corresponding dual node $\star f$ is the only source for $\overline{\xi}$ in the dual graph. The external face, dual node $\star f_0$ is the only sink. A specific weight $h(\star f)$, usually set to 1, is associated with each source $\star f$.

The dual form with a value for each directed dual edge defines a differential form for the primal graph using $\xi(e) = \overline{\xi}(\star e)$. For each face $f_i$ that is an obstacle, we create one such differential form $\xi_i$, which equivalently sums to $h(\star f_i)$ around the boundary $\partial f_i$ – representing a cyclic vector field along $\partial f_i$. We refer to these as the signature forms for the obstacles. The idea is shown in Figure 2.

The important property of this structure is that over any cycle $\gamma$ surrounding $f$, the form sums to $\xi(\gamma) = \xi(\partial f) = h(\star f)$.

**Theorem 3.1.** For a 2-chain $U$ with coefficients in $0, 1$ in $G$ (excluding $f_0$) with a single source face $f_1$, the differential form $\xi(\partial U) = h(\star f_1)$ if $f_1 \in U$, otherwise $\xi(\partial U) = 0$.

The proof follows simply from the fact that for any face $f$, $\xi(\partial f) = h(\star f)$, and that $\partial f_1 + f_2 + f_3 + \ldots = \partial f_1 + \partial f_2 + \partial f_3 + \ldots$.

**Topological signatures of trajectories.** Using these differential forms, we can now define the topological signature of a trace. If there are $k$ obstacles and corresponding signature forms, then for a trace $T$, we get a $k$ dimensional vector $\theta(T)$. The component $\theta_i(T)$ represents to what extent $T$ goes around obstacle $i$. If it forms exactly one complete cycle, then $\theta_i(T) = h(f_i)$.

The useful feature here is that even if $T$ does not quite form a cycle, $\theta_i(T)$ still reveals valuable information. If the value of $\theta_i(T)$ is higher, it implies that $t$ goes farther around the obstacle. This construction is more general than simple winding numbers, since the differential forms construction can assign arbitrary weights to individual edges, and can be made sensitive to specific mobile agents. From a computational point of view, it works directly on a discrete structure, with or without knowledge of locations. The symmetric nature of directed edges nullifies the effect of noisy GPS localization.

The signature $\theta : T \rightarrow \mathbb{R}^k$ maps trajectories in set $T$ to a $k$ dimensional space.

### 4 ALGORITHMS

This section describes how the mathematical construct of discrete topological signatures can be realized, including in a distributed environment.

#### 4.1 Construction of planar graphs

The algorithms work on any planar graph on the mobility domain. Following are a few possible strategies varied by availability of mobility data and infrastructure.

**Graphs from data.** In many scenarios, road map data is available and directly yields a graph, where road intersections are vertices and road segments are edges. Alternatively, a simple model – a grid or a triangulation of randomly deployed point sets (vertices) may serve as the planar graph. In [27] a triangulation of the locations from the dataset itself is used as the discrete domain. A mobile trace is then converted to a sequence of edges in this graph. This can be done by simply mapping locations to nearby vertices. With sufficient data, it is also possible to construct a roadmap itself as is done in openstreetmap and other projects [8].

**Discrete sensor domains.** In sensor networks and robotics, trajectories may be recorded by sensors. Such a trajectory may not have any localization at all, and is represented simply as a sequence of sensors that have detected the mobile object. For localized and unlocalized wireless sensor networks, there is a large body of research in topology control, localization and planar graph extraction, relying
on the locality of wireless transmission. Depending on the network, such a technique can compute planar graph of the nodes [16, 30].

Our differential forms setup, as described in the previous section, works on the planar graph, without the need for an explicit embedding.

4.1.1 Obstacles and dual sources. Having constructed the planar graph and map of trajectories to sequence of edges, we are left with the question of what are the ‘obstacles’ in the domain. In principle, every face can be an obstacle, but this is neither intuitive, nor informative to any algorithm operating on the data.

Instead, a face can be an obstacle if it is larger than a certain size, measured as area, diameter, or some other measure. The strength $h(\ast f)$ of the source $\ast f$, which is normally set to 1, can also be set to be proportional to the measure to reflect the weight of an obstacle. In unlocalized scenarios, there are techniques for detecting boundaries of large enough sizes [32], while a measure such as the number of edges on the face can be used to determine its weight.

Subsection 5.3 discusses more involved questions of determining the importance of an obstacle and dimension reduction techniques to merge nearby obstacles to simplify data.

4.2 Constructing signature forms

For each obstacle face $f$, our basic strategy is to take its dual face $\ast f$, and start multiple walks in the dual graph. The walks spread in all directions, and end in the exterior face at infinity $\ast f_0$. The total weight of the walks is $h(\ast f_0)$. Having constructed walks of total weight of $h(\ast f_0)$, the weight of each dual edge can be transferred to the corresponding primal edge as $\xi(\ast f_0):=\ast \xi(\ast f_0)$.

In a distributed sensor network scenario, the operations of a dual node $\ast f$, which is a face $f$ in the primal, can be handled by an elected sensor node on its boundary. There are different ways to structure these walks and the corresponding differential form. Unless mentioned otherwise, the following mechanisms work in localized as well as unlocalized graphs.

Random walk in the dual. Starting from the dual node $\ast f_i$, create several random walks that stop only when they reach $\ast f_0$. Each walk has a weight $w$, and to each directed dual edge $\ast ab$ it traverses, the weight is added as $\ast \xi(\ast ab) := \ast \xi(\ast ab) + w$. Note that this process preserves the strictly one source ($\ast f_i$) and one sink ($\ast f_0$) since at every other vertex it exits every time it enters. The walk is also allowed to pass through $\ast f_i$ as well as any other dual vertex.

Shortest paths in the dual. Another possible way of constructing the signature forms is using shortest paths to the boundary faces, which are adjacent to the external face $f_0$.

From these boundary faces, select a subset either randomly or at some uniform separation. From the dual source node $\ast f_i$, a shortest path in the dual is computed to each such boundary face $\ast f_0$ and appended a final edge ($\ast f_j$, $\ast f_0$) to complete the walk.

Following a straight line to the boundary. In scenarios with available locations, instead of selecting faces at the boundary, it is possible to select specific locations at the boundary and follow a straight line to them. By picking a location to place the dual nodes inside faces, e.g., at centroids, a line starts from a source location $\ast f_i$.

Following’ a straight line in the dual graph is essentially choosing the next node in the path that minimizes the distance to the line segment [26].

4.3 Computing topological signatures

A trace consists of a sequence of edges, and for each edge $e$, the value $\xi(e)$ is computed corresponding to each obstacle $i$. Thus, for each $e$ traversed by the trajectory $T$ results in updates of the form $\theta_i(T) = \theta_i(T) + \xi(e)$.

In a distributed scenario, this works naturally, where for any edge $ab$, the $\xi$ values for this edge are held by sensors $a$ and $b$, and when a mobile object traverses this edge, it can obtain the values locally and maintain its own $\theta(T)$ vector. Or in a tracking scenario, the sensors as they track the object’s motion can maintain and handoff this value.

While in principle $\theta_i(T)$ can take any real value, the larger values of $\theta$ imply an agent circumscribing an obstacle many times. This is unlikely in many realistic scenarios such as trips in a city. In such cases, it is natural to assume that on a single trip an agent does not take sub-optimal self-intersecting paths that that go all the way around the obstacle. Thus, for such trajectories, for any obstacle $i$, $\theta_i(T) < 1$, and let us refer them as simple trajectories. Theorem 4.1 shows that for simple trajectories, the signature uniquely determines their homotopy type.

**Theorem 4.1.** For two simple trajectories $\alpha$ and $\beta$ between same source $s$ and destination $d$:

$$\theta_i(\alpha) = \begin{cases} \theta_i(\beta) & \text{if } \alpha, \beta \text{ have same homotopy w.r.t. source } f_i \\ \theta_i(\beta) \pm 1, \text{ otherwise [assuming } h(\ast f) = 1] \end{cases}$$

**Proof.** An example setup is shown in Figure 3(a). Where paths $\alpha$ and $\beta$ have the same homotopy type with respect to $f_j$ and different types with respect to $f_i$. The concatenation of $\alpha$ with $-\beta$ gives loop $\gamma = \alpha - \beta$.

The first claim simply says $\theta_j(\alpha) = \theta_j(\beta)$, since otherwise $\theta_j(\alpha - \beta) \neq 0$, contradicts Theorem 3.1.

For the second claim, any closed oriented one dimensional loop object $\gamma$ in the two dimensional cell complex is the boundary $\partial U$ of a chain $U$ with coefficients in $[0, 1]$. Thus $f_i \in U$ implies $\xi(\gamma) = 1$. 

---

**Figure 3:** (a) Two simple trajectories $\alpha, \beta$ go from $s$ to $d$ have same homotopy type w.r.t. source $f_j$, but different homotopy type w.r.t. $f_i$. (b) Two self-intersecting paths between same source and destination may have same signature value but different homotopy type.
Depending on the orientation of $y = \alpha - \beta$, $\theta(\alpha - \beta) = \pm \theta(\hat{y}) = \pm 1$, implying the second claim.

Figure 3(b) shows an example where the above condition is violated, and trajectories of the same signature have different homotopy types.

### 4.4 Reconstructing traces from signatures

The following lemmas say that given a signature it is possible to check in polynomial time if such a trajectory exists, and if so it can be reconstructed up to homotopy equivalence in linear time.

**Lemma 4.2.** Given a signature vector $X$, source $s$, and destination $d$, it is possible to find in linear time if there exists a homotopy class of simple paths going from $s$ to $d$ with the signature $X$.

**Proof.** Consider the shortest path $P$ from $s$ to $d$ as shown in Figure 4(a) and its signature $\theta(P)$. Using the result from Theorem 4.1, the only thing to test is if $(\theta_i(P) - X_i) \in \{0, 1, -1\}$ for all $i$. This can be done in linear time in the number of holes in the domain. $\square$

**Lemma 4.3.** Given signature vector $X$, source $s$, and destination $d$, there exists an algorithm to find a unique simple trajectory up to homotopy equivalence with signature $X$. The algorithm runs in linear time in the number of holes in the domain.

**Proof.** A natural extension of Lemma 4.2 uniquely identifies the source faces $f_i$ where we need to change the side by which $P$ passes $f_i$ – precisely where $(\theta_i(P) - X_i) \neq 0$. Next, these sources are fixed iteratively.

Initialize this process by setting the path $P$ as the shortest path from $s$ to $d$. Suppose at iteration $l$, the path is $P^l$ and the next obstacle to fix is $f_i$. Consider an arbitrary node $b \in P^l$, and another arbitrary node $a \in \partial f_i$ as shown in Figure 4(b). Now $P^{l+1}$ is the concatenation of the following path segments: $s$ to $b$, $b$ to $a$, $a$ to $a$ (along $\partial f_i$), $a$ to $b$, and finally $b$ to $d$. So, the path $P^{l+1}$ has now right homotopy type w.r.t. $f_i$.

This construction essentially determines the homotopy type from the signature. Given that, we can now compute a short path in that class using well known algorithms in computational geometry [18].

5. **APPLICATIONS AND OPTIMIZATIONS**

Here we discuss how they can be designed and implemented for better applied analytics.

5.1 **Nearest neighbor search and Locality Sensitive Hashing**

A fundamental question in data analysis is the search for nearest neighbor, or $k$ nearest neighbors, which forms the basis of several classification and clustering techniques and queries on noisy data. In case of trajectories, searching for nearest neighbors is expensive due to the fundamental cost of comparing two long trajectories [3].

Locality sensitive hashing (LSH) is used in data mining to accelerate the search for nearest neighbors. However, defining LSH for trajectories is a non trivial problem, as is constructing computationally efficient hash functions [14]. We want to instead use representations of trajectories as points in the Euclidean space, where good LSH techniques are known.

A standard LSH scheme [11] uses a hash function $h$ constructed as follows. Select a random straight line in $\mathbb{R}^k$. Then project each data point linearly onto the line. Partition the line into segments of length $w$ for some given $w$, and treat each segment as a "bucket". Nearby data points are likely to hash into the same bucket, while far apart points are likely to hash to different ones. Given a query point $q$, we can perform the same projection and hashing, and then search for its nearest neighbor in the bucket containing $q$. Expensive methods [3] can be used for this search since the pruned candidate set is smaller. Repeating the process with multiple hash functions increases the probability to find the true nearest neighbor of $q$. Note that Euclidean hash is substantially more efficient than direct locality sensitive hashing of trajectories described in [14, 20].

5.2 **Predicting motion at large scales**

Let us define motion prediction at scale $r$, where $r$ is a given distance – e.g. 500 meters. We wish to ask where will a mobile object be, when it is at a distance $r$ from its current location. Formally suppose $o$ is the current location, $C_r$ is the circle of radius $r$ centered at $o$. Based on our mobility data, we can predict $p$ on $C_r$ as a point where the object’s trajectory will exit (intersect) $C_r$.

To measure the error in prediction, suppose the object actually exits the circle at point $z$. The prediction error is defined as the angle $\phi = \angle poz$. More accurately, it is $\min(|\phi|, |2\pi - \phi|)$. See Figure 5(d).

Since the scale itself is given, the essential task here is to predict the right direction from $o$ at scale $r$, which is represented as predicting the angle.

The motion prediction itself can proceed according to any machine learning method. Experiments in 6 finds that standard prediction techniques based on $k$-nearest neighbors, linear regression, random forest, gradient boosting trees work well.

5.3 **Dimension reduction, correlation and domain analysis**

Given that the trajectories are now points in $k$ dimensional Euclidean space given by $\theta(T)$, standard analytic techniques can be applied to better understand the trajectories, and to understand the
the underlying domain from the perspective of the given trajectories. We can ask, which obstacles are important, or which obstacles are similar with respect to actual mobility patterns?

Figure 5(a) shows a set of trajectories moving in the plane around obstacles 1, 2, . . . , 8. Clearly, there is a natural grouping of obstacles where obstacles 3 and 4 are similar in the sense that they act practically as a single obstacle, since every trace passes either to the right or left of both. The same holds for pairs (5, 6) and (7, 8). Obstacles 1, 2 are also similar in the sense that most traces behave similarly with respect to both, with only a few passing between them.

These intuitive results can be obtained from standard analytic and visualization methods. Figure 5(b) shows the correlation matrix of \( \theta \), where darker shades indicate higher correlation. The correlation lets us have a more abstract view of the domain, where related nearby holes can be viewed as one.

Techniques like Principal Component Analysis can be applied to the Euclidean data to determine which dimensions record larger variance – implying larger variability in how traces traverse the obstacles. We apply this idea to get the intuitive result in Figure 5(c) that obstacles 1 and 2 are clearly more significant than others as they split a large body of motions.

**Greedy filtration of obstacles.** A different strategy which is found effective in experiments, is to greedily select obstacles (dual sources) that maximize the accuracy for the task at hand.

For example, for nearest neighbor search, this accuracy can be defined as the fraction of times that the true Fréchet nearest neighbor is contained among the \( m \) nearest neighbors. The greedy strategy will iteratively select the obstacle that maximizes the total accuracy over all pairs, until it has selected some \( k \) obstacles. For scenarios with large number of obstacle, similar strategies can be implemented for distributed cluster computation [7]. The benefit of this heavy computation is that the signatures become more compact and assuming the dataset is representative of typical motion, future computations of signatures, nearest neighbors, predictions etc will be more efficient.

**5.4 Adaptive resolution signatures**

It is natural to ask for representations that give greater weight to recent information and less weight to history, for example, for short term prediction. The signatures can be seamlessly adapted to achieve this using weighted averaging. Where for each edge traversed by the trajectory, we modify the update rule to be \( \theta_i(T) = \beta \theta_i(T) + (1 - \beta)\xi_i(e) \) for a constant \( 0 < \beta < 1 \) that determines the weight of history in the signature.

In the example in Figure 6(a) at point \( o \), the adaptively weighted signature finds the set \( y \) to be currently more relevant to the query \( \alpha \).

**5.5 Composing signatures**

The composability of differential forms means that a set of traces can be represented in the signature space as a linear combination of the constituent traces – such as the mean of all the signatures. This simplifies center computation by creating an abstract ‘center’ that is not a trajectory itself, but can be used to compare other trajectories to the given cluster or set.

**5.6 Implementation in distributed and mobile setups**

Mobile computers are a natural platform for application of trajectory signatures, with applications in social mobile networks, autonomous driving, etc. For large computers in automobiles, the differential forms can be pre-loaded into the memory, downloaded incrementally as required. In sensor or other dense network scenarios, the mobile device can obtain local values of \( \xi \) from nearby
sensors and devices. The signature is computed and stored incrementally.

The knowledge of signatures allows mobile entities to exchange data with other nearby ones and predict which ones are likely to have similar behavior. This is useful, for example, in autonomous vehicles and delay tolerant networks.

The above constructions of differential forms are all naturally implementable in distributed scenarios like sensor networks. For example, consider a Voronoi diagram of sensors where they can detect the direction and identity of the objects crossing the edges of the Voronoi graph, as used in [29]. The sensors maintain the weights on the edges giving the differential forms and as an object moves between cells, its signature is handed off (similar to hand-off in cellular network) to the successive sensor. An alternative approach could be for the sensors to record and store the edge crossings of mobile objects, and compute the signature on demand by summing along the handoff path.

6 EXPERIMENTS

In this section, we show experimental evidences that the topological signatures simplify the representation of trajectories and also perform well in clustering, prediction, and other tasks. The main observations are:

- Nearest neighbor of a query trajectory according to Fréchet distance can be found efficiently using fast Locality Sensitive Hashing and pruning on topological signatures.
- Motion prediction of trajectories is accurate and efficient. It achieves similar accuracy as Long Short Term Memory neural networks (henceforth LSTM), but provides much faster training and query time. Prediction results are verified using two real mobility datasets.
- Greedy source selection reduces dimension to get compact signatures, with minimal loss in quality of analytic results.
- Popular paths or paths with the most number of trajectories in a given region can be estimated by kernel density estimation on signatures.
- Topological signature based method is robust to sparse, noisy trajectories: it can predict future direction of mobile objects accurately in trajectories with many missing data points.

6.1 Experimental setup

Dataset. We use two publicly available datasets: Rome Taxi dataset [6] and Porto Dataset. While the former has trajectories of 320 taxis for a month in Rome, Italy, the latter has trip partitioned trajectories of 442 taxis for one and half years in Porto, Portugal.

Constructing planar graph. The Delaunay triangulation [15] of 20000 random points in the plane make the planar graph $G$. Each gps point of a trajectory is mapped to its nearest node in $G$ and connecting them by the shortest path in the graph produces its representation in $G$.

Identifying obstacles. All faces in $G$ where less than a predefined threshold number of trajectories pass a boundary edge are considered as obstacles. E.g, the portion of Rome dataset in Figure 7(a) have 67 such obstacles with threshold 2. In Porto dataset (Figure 10(a)), we choose 30 largest regions of low trajectory density as obstacles.

Constructing differential forms. Dual paths along linear rays with random slopes from each obstacle yields the differential forms (Section 4.2). E.g, in both Figure 7(a) and Figure 10(a) 20 dual paths each with weight 0.05 are constructed from each obstacle.

6.2 Nearest neighbor search

Fréchet nearest neighbor of a query trajectory is approximated by the pruning based on LSH with trajectory signatures using independently chosen 3 sets of 5 linear projection hash functions. This experiment uses each trajectory in Figure 7(a) as a query trajectory in turn.

Figure 7(b) confirms that with high probability, Fréchet nearest neighbor is included in the approximate $k$ nearest neighbors found by LSH. Larger bucket size naturally yields better accuracy and offers standard accuracy and efficiency trade-off of LSH.

In efficiency, even including preprocessing time to construct planar graph (15.9 sec), differential forms (20.5 sec), and computing signatures (varies with dataset size), this method greatly outperforms basic Fréchet based method of computing distance to all trajectories (Figure 7(c)). The Euclidean distance measure is faster than Fréchet and LSH based pruning makes a query even faster. Fréchet distance is computed using [1]. The experiments are run on a typical desktop machine with 8 GB primary memory and Intel i5 processor.

6.3 Clustering and estimating density

The signatures are useful to cluster trajectories. Figure 8 shows a small example with 66 trajectories. The varied types of trajectories are well separated even if they have large portion of overlapping pattern.

High density regions in the signature space correspond to popular topological types of trajectories. The popular bidirectional traffic flow in Figure 9(a) is neatly identifiable in the estimated density using Gaussian Kernel in Figure 9(b). Clustering can also be applied to anonymize personal data [36].

6.4 Motion prediction

Figure 7(a) and Figure 10(a) show the datasets from Rome and Porto respectively used in prediction experiments. With randomly chosen 90% trajectories as training, we predict direction $\phi$ for different scales $r$ on each test trajectory where current location $o$ is randomly chosen from middle 1/3rd of its length. The prediction error is measured as described in Section 5.2.

Using topological signatures, next direction is predicted as the mean direction $\phi$ of $k$ nearest training trajectories that pass within 20 meters of $o$. The nearest neighbors are computed using $kd$-tree on the space of signatures with the training trajectory segments ending near $o$. In both Porto and Rome dataset, our method attains high accuracy even with small number of nearest neighbors (Figure 10(b)) and large prediction radius (Figure 11).

As a benchmark we use neural networks with LSTM as regresors [17] to predict $\phi$, given equi-spaced 20 locations on the trajectory between $o$ and start of the trajectory. Each model has two
The busiest road in this map is easily visible. Two signature density threshold as $k$ (d) nearest neighbors are used, Sources are placed based on the trajectory signatures. With suitable segment length, Locality Sensitive Hashing can reliably approximate Fréchet based nearest neighbor. (c) Time to run naive Fréchet increases super linearly when increasing the size of the dataset whereas topological signature method consumes very little time in comparison. (d) $k$ nearest neighbor query accuracy with 5 obstacles selected using different source selection strategies.

![Figure 7](image1.png)

**Figure 7:** (a) A region (latitude=[41.8708°N, 41.8826°N] and longitude=[12.4919°E, 12.5089°E]) of size roughly 1km x 1km from Rome taxi dataset with 1189 sub-trajectories. We detect 67 internal holes (shaded) using trajectory density threshold as 2. (b) CDF of percentage of times the nearest neighbor is included in the approximate $k$ nearest neighbors ($x$-axis) found using Locality Sensitive Hashing based on topological signatures. With suitable segment length, Locality Sensitive Hashing can reliably approximate $k$ nearest neighbors to find Fréchet based nearest neighbor. (c) Time to run naive Fréchet increases super linearly when increasing the size of the dataset whereas topological signature method consumes very little time in comparison. (d) $k$ nearest neighbor query accuracy with 5 obstacles selected using different source selection strategies.

![Figure 8](image2.png)

**Figure 8:** Clustering 66 trajectories from Rome taxi dataset using DBSCAN clustering algorithm with neighborhood distance parameter set as 0.25 and minimum samples to form a cluster is set as 2. To construct the triangulation, 20000 random points are used, Sources are placed based on the trajectory density threshold as 2.

![Figure 9](image3.png)

**Figure 9:** Detecting popular topological types using kernel density estimate of the trajectory signatures. (a) Portion of trajectories from [6] with 1809 trajectories (area 1km x 1km). The busiest road in this map is easily visible. Two signature sources are manually placed in the map. (b) Two hotspots in the kernel density estimate denotes the two way traffic on the busy road.

In domains with large number of obstacles, the signatures will have correspondingly large size, losing the benefits of compact representation. We found that in practice, differential forms with respect to a small number of sources suffice to accurately capture the motion.

**6.5 Signature source selection**

In domains with large number of obstacles, the signatures will have correspondingly large size, losing the benefits of compact representation. We found that in practice, differential forms with respect to a small number of sources suffice to accurately capture the motion.

**Figure 7(d)** compares different signature source selection strategies in Rome dataset from Figure 7(a); different strategies can select small number of signature sources to attain high nearest neighbor query accuracy. Specifically, this experiment compares the following strategies: i) all faces in $G$ with low trajectory density (holes) are signature sources, ii) $k$ largest holes, iii) greedy selection as described in Section 5, and iv) randomly choose $k$ sources from all
Topological Signatures For Fast Mobility Analysis

Figure 10: (a) A region latitude–[41.1468°N, 41.1699°N] and longitude–[8.62931°W, 8.60411°W] of size 2.5Km × 2.1Km from Porto dataset with 3952 trajectories. Among 101 regions with low trajectory density (threshold 2), we consider largest 30 to be obstacles. (b) In Porto dataset, topological signature based method accurately predicts next direction even with small number of nearest neighbors. Here, r = 100 meters.

Figure 11: Topological signature based method can accurately predict next direction even for large r in (a) Porto dataset and in (b) Rome dataset. Here, k = 4 neighbors.

Figure 12: (a) Topological signature based method attains similar accuracy as LSTM. (b) Popular regression methods using topological signatures achieve similar high accuracy.

Figure 13: (a) Preprocessing and (b) query time. Both increase slowly for topological signature based method compared to LSTM, with dataset size.

Figure 14: Predicting locations with sparse trajectories: Each location point is retained with certain probability in each trajectory. Our method achieves high prediction accuracy even with sparse trajectory.

7 RELATED WORK

Hodge decomposition of a randomly generated vector field has been used in [35] to classify paths into homotopy types. This approach relies on a numerical process, and only applies to trajectories with same start and end points. This problem is avoided to an extent in [27], which uses relative homology, and marks certain regions of the space where start and end points lie, but other trajectories do not pass. Thus these approaches do not apply to general trajectory datasets of the type we have considered here. Both these methods are also computationally expensive.

In other works, topological characterization has been used to measure distances between curves [10] but restricted to curves with same start and end points, since homotopy cannot be defined between open curves. A seminal work [5] Baryshnikov and Ghrist used integrals of Euler characteristics to compute number of mobile targets. Topological persistence can be used to simplify trajectories [22]. Discrete exterior calculus has been a subject of study in graphics, modeling and discrete geometry and developed in several different flavors [12, 19]. In sensor network, it has been used for performing range queries of mobile objects [29].

Mobility modeling using (hidden) Markov models have been developed in [24, 34, 37] and other works, usually with the objective to predict the agent’s next road segment. As mentioned earlier, the Markov assumption can be shown to be unrepresentative of typical trajectory datasets [31]. In line with recent developments in machine learning, Neural networks have been designed for the prediction task. Recurrent Neural Networks using Long Short Term Memory are considered useful in sequential data, and has been used
We presented a topological framework to represent a trajectory as a point in a Euclidean space, enabling natural applications of well established machine learning and mining techniques. The topological construct preserves relevant qualitative features in trajectories while ignoring the extraneous details and noise. The framework can be set up using fast distributed algorithms and used in an online manner by mobile entities. The framework is quite general and flexible, and it allows application specific choice of obstacles and domains. Dynamic obstacles, temporal characteristics of trajectories are interesting future directions to consider.

8 CONCLUSION

We presented a topological framework to represent a trajectory as a point in a Euclidean space, enabling natural applications of well-established machine learning and mining techniques. The topological construct preserves relevant qualitative features in trajectories while ignoring the extraneous details and noise. The framework can be set up using fast distributed algorithms and used in an online manner by mobile entities. The framework is quite general and flexible, and it allows application specific choice of obstacles and domains. Dynamic obstacles, temporal characteristics of trajectories are interesting future directions to consider.

REFERENCES