A Benchmark Framework for Trajectory Vectorization in Unsupervised Settings

Cristiano Landi[1,2,*], Petros Mandalis[3] and Nikos Pelekis[3]

[1]Univeristy of Pisa, Italy
[2]KDD-Lab at ISTI-CNR, Pisa, Italy
[3]Department of Statistics and Insurance Science, University of Piraeus, Greece

Abstract

The growth of location-tracking technologies has generated large spatiotemporal datasets that require efficient analysis methods. Recent approaches focus on vectorizing the data as a preprocessing step to enable the application of off-the-shelf classical machine learning techniques. To date, such approaches have primarily been evaluated in supervised tasks. This paper presents a benchmark framework for assessing these methods in unsupervised settings, where no target labels are available. We extend and systematically evaluate four families of vectorization techniques: feature-based, shapelet-based, dictionary-based, and matrix-based. The framework assesses both clustering quality and cross-vectorization diversity through quantitative and qualitative analyses. Overall, the findings demonstrate that these vectorization approaches offer a valuable option for unsupervised trajectory analysis. Additionally, our paper provides insight into which vectorization technique is most suitable for the specific characteristics of the task under analysis.

Keywords

Spatial-temporal systems, Knowledge representation and reasoning, Human Mobility Modeling

1. Introduction

Spatiotemporal datasets are continually expanding and include videos, social media posts, GPS traces, and other time-series data. Effective analysis of such data can have a significant societal impact, helping to address challenges in environmental and climate change, public security [1], and healthcare [2].

Several researchers in Mobility Data Science (MDS) [3] focus on understanding and analyzing this data and producing reports on various observed phenomena. This led to the development of application- and data-specific procedures [4], which reduce the reusability of algorithms. Silva et al. [5] compared various trajectory analysis methods and found that approximately 70% of them are tailored to specific applications such as transportation mode identification. To overcome this limitation, researchers have turned to the field of time series for inspiration and have explored various ways to represent spatiotemporal data to meet different analysis objectives. The concept is to transform the input sequential data into different vector-based feature spaces that can be utilized with standard, general-purpose machine learning models, thereby minimizing the need for manual feature engineering. For example, in [6], the authors explore the concept of shapelet transform, introducing Geolet to transform input trajectories into a (dis)similarity matrix that compares the input data with specific discriminant subtrajectories extracted from it.

Although these methods achieve state-of-the-art performance in classification tasks, they have not yet been evaluated in an unsupervised setting. We therefore propose a unified benchmarking framework for evaluating trajectory vectorization techniques in such a setting. Within this framework, we formulate the problem as trajectory clustering, which is the task of identifying groups of trajectories that exhibit similar movement patterns. Despite multiple definitions of trajectory similarity, most prior work assumes that trajectories are similar if spatially close [7], making clustering based on movement

Published in the Proceedings of the Workshops of the EDBT/ICDT 2026 Joint Conference (March 24-27, 2026), Tampere, Finland
*Corresponding author.
cristiano.landi@phd.unipi.it (C. Landi); pmandalis@unipi.gr (P. Mandalis); npelekis@unipi.gr (N. Pelekis)
https://github.com/cri98li (C. Landi)
0000-0003-4907-9728 (C. Landi)

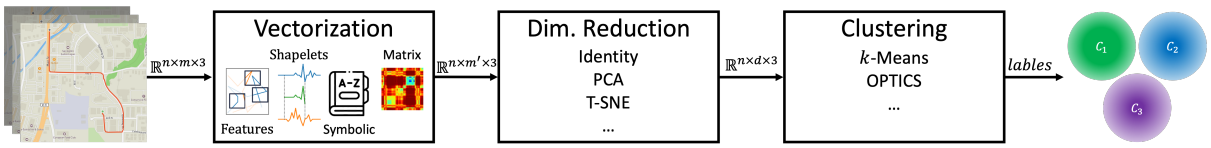

Figure 1: Overview of the proposed framework.

dynamics rather than spatial proximity far less explored in the literature. In this paper, we instead shift the focus to evaluating these novel vectorization techniques on the task of clustering trajectories based on similar events, namely their movement dynamics, while disregarding the spatial region. Under this perspective, two trajectories are assigned to the same cluster if they exhibit similar movement patterns. For instance, two trajectories that include a harsh stop should be grouped into the same cluster, even if they occur in different geographical regions.

The contribution of this work is fourfold: (i) we propose a unified benchmarking framework for evaluating trajectory vectorization techniques in unsupervised settings; (ii) we introduce a movement-oriented dictionary-based discretization scheme that captures kinematic and geometric patterns independently of geographic location; (iii) we extend Geolet [6] with an unsupervised shapelet selection mechanism; (iv) through extensive experiments we demonstrate that different vectorization techniques encode complementary mobility characteristics.

2. Background and Problem Setting

Trajectory clustering consists of partitioning a set of trajectories into groups such that *similar* trajectories are assigned to the same cluster, while dissimilar trajectories are placed in different clusters. In mobility research, similarity is typically interpreted as spatial proximity between trajectories [7]. In contrast, our objective is to examine a more abstract and dataset-dependent notion of similarity. For instance, given a dataset in which certain trajectories exhibit hazardous driving events, such as harsh braking, abrupt turns, or sustained high-speed intervals, we aim to determine whether, and to what extent, different vectorizations reveal these behavioral distinctions in the absence of labeled data.

We begin by defining the simplest type of mobility data, namely the position of an object in 2-dimensional space over time. Formally:

Definition 1 (Trajectory)**.** A *trajectory* X is a sequence of spatio-temporal points $X = \{(lat_0, long_0, t_0), \dots, (lat_m, long_m, t_m)\} \in \mathbb{R}^{m\times 3}$ where the spatial vectors $\vec{x}_j = (lat_j, long_j)$ are sorted by increasing timestamp t_j.

This type of data is typically not sampled at a constant rate, as observations are often recorded irregularly due to sensor limitations, energy-saving policies, communication constraints, or event-driven acquisition processes. Additionally, trajectories into a dataset often have a different number of observations, so we denote by $\mathcal{X} \in \mathbb{R}^{n\times m\times 3}$ a set of n trajectories of length *at most* m. Many mobility analysis tasks can be formulated as a two-phase process: first, the data are transformed into a tabular vector representation, then analyzed using classical machine learning methods. For example, consider the following definition of the trajectory clustering task:

Definition 2 (Trajectory Clustering)**.** Given a trajectory dataset $\mathcal{X}$, *Trajectory Clustering* is the task of defining a function f from the space of possible input trajectories $\mathcal{X}$ to an output $c \in \mathcal{C}$ denoting its cluster identity.

We can define the clustering function f as the composition of a vectorization function g, which maps each trajectory to a fixed-size feature vector T, and a function h, which maps $g(\mathcal{X})$ to one cluster. By changing the vectorization function g, the definition can accommodate different clustering objectives.

3. Related Work

We organize the related work around our definition of a trajectory clustering function f as the composition of a vectorization function g and a clustering function h that assigns $g(X)$ to a cluster. Since h can be any off-the-shelf algorithm, such as k-MEANS or OPTICS, we first review the Mobility Data Science (MDS) [3] literature from the perspective of trajectory vectorization. We then summarize existing clustering approaches in MDS to emphasize the diversity of similarity objectives. We conclude by situating our framework within the current literature.

Vectorization techniques for MDS To better structure this section, we adopt a time series taxonomy that categorizes approaches by the core idea underlying data transformation [8]. *Feature-based* vectorizations compute explicit motion descriptors like speed, acceleration, and heading change and aggregate them into trajectory-level vectors; they often require the practitioner to select features appropriate to the specific task and data characteristics, and can be sensitive to preprocessing steps like resampling or segmentation and aggregation choices [9]. In *shapelet-based* vectorization, trajectories are represented by their similarity to discriminative sub-trajectories, following the shapelet idea from time series [10]. Such approaches are fully data-driven, avoiding the heavy feature engineering often required in feature-based approaches. In MDS, the most prominent approaches are MOVELETS [11] and GEOLET [6]. In *dictionary-based* approaches, the core idea is to discretize the data into tokens, enabling the use of text-inspired representations such as bag-of-words or tf–idf. In MDS literature, most works focus on data compression, for example, through tessellation techniques or pathlet learning algorithms. In our setting, however, discretization must preserve episodic events rather than merely serving compression or coarse partitioning [12]. *Matrix-based* vectorizations encode trajectories into structured objects (e.g., images/accumulators) and then embed them into vectors, namely transform-based encodings that can provide invariances and reduce overlap artifacts, as in rotation/scale-invariant representations [13]. *Deep-learning* vectorization techniques use trained encoders to produce task-specific embeddings. Their results are highly sensitive to architectural and optimization choices, making it challenging to disentangle the impact of the vectorization strategy itself. A comprehensive assessment of such methods is deferred to future work.

Trajectory clustering and similarity objectives. Trajectory clustering groups trajectories according to a chosen notion of similarity, largely determined by the representation and distance functions. Existing surveys [14, 7] note that classical methods primarily focus on spatial or temporal proximity, whereas many applications require *behavioral* similarity, capturing comparable motion dynamics across different regions. This has motivated sub-trajectory-based approaches such as TRACLUS [15], as well as semantic and mobility-oriented methods that move beyond pure geometry [16, 17]. In our setting, clustering is used as a diagnostic tool: a vectorization is effective if it yields a cluster structure that aligns with the intended similarity notion under a controlled evaluation.

Positioning of our framework. To the best of our knowledge, there is no existing framework that benchmarks *cross-family* trajectory vectorizations for an unsupervised clustering objective under a single, fixed evaluation pipeline; most related efforts either standardize mobility primitives (libraries) or benchmark different tasks and stages rather than representation families.

4. Evaluation framework

In this section, we propose a unified benchmarking framework, summarized in Figure 1, to evaluate off-the-shelf trajectory vectorization techniques. Rather than asking whether two trajectories are geographically close, our framework allows us to explore how different vectorization techniques naturally group trajectories that share behavioral or structural similarities. To make this evaluation transparent and reproducible, we define a common pipeline and measure both (i) clustering quality

within a representation space and (ii) diversity/consistency across representation spaces. The framework is composed of 3 modules: (i) a vectorization function g; (ii) an optional dimensionality reduction step; (iii) a clustering function h. Finally, the resulting trajectory clustering is evaluated both in terms of clustering quality within each vectorization and in terms of diversity/consistency across vectorization spaces.

In the following paragraphs, we motivate the inclusion of the three modules and outline the algorithms considered for each in the experimental section. The framework is general and can accommodate a wide range of algorithms per module; the choices made here are preliminary and will be extended in future work.

Vectorization functions. The first step in our framework is selecting a set of vectorization functions to compare. The scope of this module is to transform complex trajectory data into a tabular-like format such that a classical off-the-shelf machine learning technique can be applied. Motivated by recent work in mobility, we select 4 different functions. Additionally, inspired by the time series literature, we divided them into feature-based, shapelet-based, dictionary-based, and matrix-based functions. For *feature-based* vectorization, i.e., where explicit descriptors are extracted from the data, we selected the Trajectory Interval Forest (TIF) [18]. TIF randomly generates a set of intervals and, for each interval and trajectory in the dataset, it extracts a wide range of descriptors (speed, acceleration, sinuosity, intensity use etc.). For *shapelet-based* vectorization, which extracts a set of discriminatory subsequences and uses them as references for data transformation, we select Geolet. Geolet relies on statistical methods to identify discriminatory subsequences, and represents each trajectory by its similarity to the selected patterns. Since Geolet was designed for supervised settings, its subsequences selection requires prior knowledge. In Section 4.1, we extend Geolet to unsupervised tasks. Finding a *dictionary-based* vectorization was more challenging. In time series analysis, this family of methods discretizes the input into sequences of symbols, which are then matched against a learned dictionary of patterns. This allows temporal dynamics to be represented through the frequency or arrangement of symbolic motifs. In the mobility literature, discretization is typically performed using tessellation techniques or pathlet-learning approaches [12], but these methods mainly capture global spatial characteristics of movement. As a result, two harsh right turns occurring in different geographic regions would be encoded as different symbols. In Section 4.2, we propose a simple yet effective approach to address this limitation. Finally, for matrix-based vectorization, we selected the algorithm proposed in [19], namely RoSITa. This vectorization technique represents each trajectory in a scale- and rotation-invariant representation by performing the Hough transform.

Optional feature reduction. As some vectorization functions may produce very high-dimensional or sparse feature spaces, this step aims to mitigate the curse of dimensionality. Due to their widespread use among practitioners, we selected Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE). We also introduced a pass-through option that allows the clustering algorithm to be executed directly on the full vectors.

Clustering algorithms. The third module applies an off-the-shelf clustering algorithm to the resulting (reduced) vectorized dataset. We consider two widely used methods from different families: k-means for centroid-based clustering and optics for density-based clustering. This selection will be expanded in future work.

4.1. Geolet for Unsupervised Tasks

This section extends Geolet to unsupervised tasks. Geolet is composed of four steps: (i) trajectory segmentation into a set of candidate discriminative sub-trajectories, (ii) sub-trajectory normalization, (iii) sub-trajectory filtering to select the most discriminative candidates, and (iv) transformation of the dataset into a dissimilarity matrix between each trajectory and the selected discriminative candidates. Since Geolet was originally proposed for supervised tasks, the filtering step relies on dataset labels to

identify, through statistical tests, the most relevant discriminative sub-trajectories. We therefore focus on replacing this step with a more general approach suitable for unsupervised settings. Inspired by time series literature, we build upon the concept of *unsupervised shapelets* (u-shapelets) [20], which extends the original notion of time series shapelets to scenarios where class labels are unavailable. The underlying intuition of the proposed work is that, even in unlabeled data, certain short subsequences tend to recur within groups of instances while remaining dissimilar to the rest of the dataset. Accordingly, a u-shapelet can be identified by assessing how effectively a candidate subsequence separates the dataset into two partitions: trajectories that contain a close match to the subsequence and those that do not. We incorporate the following iterative procedure into Geolet. Candidate subsequences are first extracted from a reference trajectory using spatial, temporal, or spatio-temporal sliding windows. For each candidate, a distance measure is used to compute a dissimilarity vector for all trajectories in the dataset. The distance measure is defined as the minimal distance over all possible alignments of the subsequence within each trajectory. Sorting this vector produces an *orderline*, over which all possible split points are evaluated. For each split, a gap score measures the separation between the two induced groups by balancing the difference in their mean distances against their respective variances. Given a subsequence S that partitions the dataset into two trajectory groups A and B, the gap score is formally defined as:

$$gap = \mu_B - \sigma_B - (\mu_A + \sigma_A) \tag{1}$$

Where μ_A and μ_B are the means of the best fitting of S in all trajectories in A and in B respectively, while σ_A and σ_B represent the standard deviations. The subsequence that maximizes this separation is selected as a discriminative subsequence. Trajectories that are well explained by the selected u-shapelet are then removed from the dataset. The process is repeated on the remaining instances to extract additional u-shapelets, until the desired number of u-shapelets has been extracted or no further splits of the remaining instances are possible.

4.2. Dictionary-based vectorization

This section lays the groundwork for shifting from continuous to discrete representations of trajectories. We introduce a set of algorithms designed to transform trajectory data into sequences of symbols. This discretization provides several important benefits: it enables the use of highly efficient data structures; it simplifies storage and indexing by reducing continuous data into compact symbolic forms; and it facilitates faster similarity searches and pattern discovery. We formally define the problem of trajectory discretization as the composition of a normalization function and a discretization function, i.e., $discretize \circ normalize$. The $discretize : \mathbb{R}^{m' \times 3} \rightarrow \Sigma^{m'}$ function maps each observation in the normalized trajectory to a symbol from an alphabet Σ, producing a symbolic sequence. The $normalize : \mathbb{R}^{m \times 3} \rightarrow \mathbb{R}^{m' \times 3}$ function transforms the original trajectory into a normalized version. This step serves two purposes: first, it acts as an approximation function similar to PAA in SAX [21], reducing noise and redundancy; second, it regularizes the trajectory by interpolating or resampling to ensure a consistent sampling rate or spatial distance. This normalization ensures that trajectories of comparable "length" yield the same number of symbols, enabling more meaningful comparisons: trajectories exhibiting similar motion patterns are more likely to share a similar symbolic representation. Finally, the discretized trajectory data can be vectorized by computing a TF-IDF matrix over words extracted via a sliding window.

Normalization We present three possible normalization strategies suited for trajectory data. The first involves resampling trajectories at constant time intervals, ensuring that observations are uniformly spaced in time. The second normalizes by constant travel distance, where a new observation is sampled each time the moving object covers a fixed spatial distance. However, due to positional noise, this method may produce redundant samples when the object is stationary. To address this, we introduce a third variant that generates a new observation only after the object has moved a minimum distance from the last sampled point, rather than after a fixed cumulative distance. In all three strategies, missing observations are interpolated, and excess ones are subsampled to preserve temporal or spatial regularity.

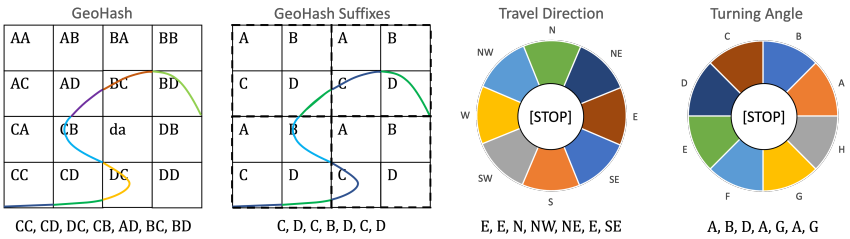

Figure 2: Strategies to transform trajectories into symbolic sequences (left to right): position encoding with Geohash, location grouping via Geohash suffixes, direction-based discretization, and turning-angle.

Discretization We present several strategies for assigning a sequence of symbols to a trajectory. A first, straightforward approach relies on tessellation techniques already established in the literature. Figure 2 (left) illustrates a conceptual example using Geohash [22], where each geohash region crossed by the trajectory generates a corresponding symbol[1]. The same idea can be extended to other spatial partitioning methods. However, since our goal is to analyze multiple aspects of trajectories beyond mere origin–destination movement, we aim to design a discretization strategy that yields comparable symbolic vectorizations across different geographic contexts. One possible approach is to leverage the hierarchical structure of some tessellation encodings, such as Geohash. Leveraging this property, we can design discretization techniques based on suffix relationships among GeoHash sequences, enabling vectorizations that remain structurally consistent even when trajectories span different geographic areas. Figure 2 (center-left) illustrates such an example. The main limitation of this approach is its reliance on a fixed spatial grid: two similar movements are represented by similar subsequences only if they occur within the same subregion. For instance, consider two right-turn events–one from cell "CB" to "AD" and another from "BC" to "BD". Despite being similar, the discretized subsequences differ ("CB" vs. "CD") solely due to their geographic position. This spatial dependence limits the generalization of symbolic vectorizations across regions. To overcome this, we introduce two alternative discretization strategies that are independent of geographic location and instead rely on the intrinsic dynamics of motion. Specifically, the *traveling direction* and the *turning angle*. The first computes the direction between consecutive observations and assigns symbols based on predefined angular intervals, as shown in Figure 2 (center-right). The second focuses on the turning angle, i.e., the change in direction between successive segments. Revisiting the right-turn example, this vectorization consistently captures both events as the same symbolic pattern, generating the symbol "G" twice. The resulting longest common subsequence, "AG", effectively captures a straight movement followed by a right turn, offering a location- and orientation-independent yet behaviorally meaningful vectorization. Additionally, to account for stationary behavior, we introduce a dedicated symbol that indicates when the object is not moving.

5. Experimental Setting

This section describes the hyperparameters for vectorization, feature reduction, and clustering evaluated in our experiments. Due to limited prior knowledge of the effectiveness of these vectorization techniques for unsupervised tasks, we designed the experiments to evaluate all possible combinations of the three components. Consequently, each vectorization hyperparameter setting was evaluated with every feature reduction hyperparameter configuration, and each of those combinations was in turn evaluated with every clustering hyperparameter configuration. As the total number of experiments increases exponentially with the number of hyperparameters across modules, we prioritize extensive exploration of the vectorization hyperparameters while restricting the feature reduction and clustering stages to a smaller set of representative methods and settings.

[1]For clarity, the figure shows the full trajectory, though after normalization each cell (symbol) correspond to a single point.

5.1. Hyperparameters

Vectorization method hyperparameters. For TIF, we explored intervals defined by the number of observations, time, or distance-based intervals. The number of intervals ranged from 2 to 20, with different constraints on minimum and maximum lengths. We disabled the use of latitude and longitude to avoid making position-based tasks trivial. For GEOLET, we tested different numbers of subsequences, from 5 up to 100, using a Geohash-based partitioning with precision levels between 3 and 7. To select the most discriminative shapelets, we experimented with the algorithm described in Section 4.1. For the distance computation between trajectories and shapelets, we tested standard Euclidean, time-interpolated Euclidean, and space-interpolated Euclidean distances. For ROSITA, we used a time-based sliding window ranging from 10 seconds to 1 hour. For the Hough Transform, the angle resolution varied between 1° and 10°, and the image size ranged from 50×50 to 1080×1080 pixels. Consequently, the Hough accumulator had dimensions ranging from 70×18 to 1527×180. We tested clustering using the full accumulator, its vertically summed-squared version, and its FFT transform, as in [23]. For the dictionary-based approach described in 4.2, we evaluated all discretization and resampling strategies. Time-based resampling was tested every 1, 50, or 300 seconds, and distance-based resampling was tested every 3, 50, or 300 meters. For tessellation-based discretization, we used Geohash with precision levels 3 to 7 and suffix variants, ignoring the first 1 to 3 characters. Lower precision values help to capture movement patterns of objects traversing wide areas. Direction-based discretization employed alphabets of 4 to 16 symbols, plus one for stationarity. We then built a TF-IDF matrix over words extracted with sliding windows of 3 to 20 symbols.

Feature reduction hyperparameters. Regarding the feature reduction step, we evaluated several approaches commonly adopted in machine learning. In particular, we experimented with Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (T-SNE). For both methods, we tested output dimensionalities of 5, 10, 50, and 100 components. For T-SNE, we additionally varied the perplexity parameter over 5-100. All remaining hyperparameters for both PCA and T-SNE were left unchanged and set to their default values as defined in the scikit-learn implementation[2]. We denote the output dimension using the symbol d.

Clustering methods hyperparameters. Finally, regarding the clustering step, we limited our experiments to k-MEANS and OPTICS. For k-MEANS, we evaluated values of k equal to 2, 3, 4, 6, and 10. For OPTICS, we tested two values for the minimum number of samples required in the neighborhood of a point for it to be considered a core point: 5 and 10% of the trajectories in the dataset. All remaining hyperparameters were left at their default values[2]. Please note that clustering quality metrics were computed on the non-reduced vectorized representations. This choice was made to enable a fair comparison of clustering scores across different vectorization methods sharing the same hyperparameter configurations. Otherwise, such metrics would also be conditioned by the output of the feature reduction step, thereby confounding the evaluation of the vectorization methods themselves.

5.2. Datasets

This section describes the datasets used in our experiments and the preprocessing steps applied. We consider both real-world and synthetic datasets. This distinction allows us to capture different levels of similarity to real data. Synthetic datasets are fully controlled and used to test specific, isolated aspects of the vectorization functions.

Real-world datasets. We selected five real datasets[3] for GPS trajectory classification with different sizes, semantics, and classification objectives, i.e., `animals`, `vehicles`, `seabirds`, `geolife`. The `vehicles` and `geolife` datasets mainly represent road-constrained vehicle movements, whereas

[2]Specific default hyperparameters are available in sk-learn (t.ly/rlINN) and in the project GitHub repository (t.ly/oSmLh).
[3]`animals` and `vehicles`: t.ly/klP5M, `seabirds`: t.ly/LpMx3, `geolife`: t.ly/FbqVa.

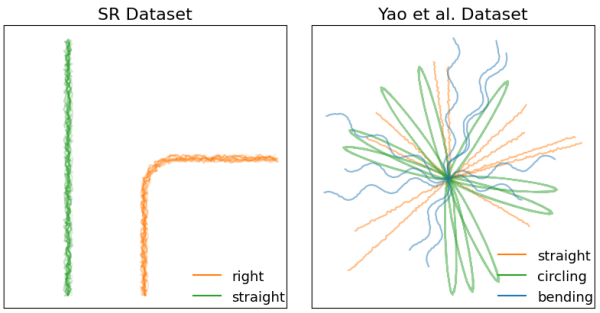

Figure 3: A sample of the two Shape-based synthetic datasets.

`animals` and `seabirds` capture unconstrained animal movements on the ground and in the air. Table 1 (top five rows) reports descriptive statistics for the datasets, including the number of trajectories, the total number of observations, the average and standard deviation of observations per trajectory, the average sampling rate, and the number and proportion of target classes. For the `animals` dataset, the classification task is to recognize three species. For `vehicles`, the objective is to distinguish between buses and trucks. For `seabirds`, the objective is to recognize the flying trajectories of three species of seabirds. For the `geolife` dataset, we set the classification problem to recognize trajectories of public versus private means of transport as in [6].

Synthetic Datasets For synthetic datasets, we use an extended version of those introduced in [24], consisting of a larger number of trajectories per dataset. The synthetic datasets are deliberately simple to isolate the hypothesis under analysis, minimizing confounding variables such as traffic, traffic lights, and road priorities.

To measure the extent to which different vectorization techniques capture the movement's geometry, we generated two synthetic datasets shown in Figure 3. The first dataset, `Straight-Right (SR)`, consists of objects that either move in a straight line or make a 90-degree right turn. Each object in this dataset travels at a constant speed for precisely 100 seconds. The second dataset, `Yao`, is inspired by the dataset introduced by Yao et al. in [25]. It includes three distinct movement patterns: straight, circling, and bending. Unlike the `SR` dataset, this dataset introduces additional complexity to the analysis task, as each object travels in a different direction with a variable sampling rate for a varying number of seconds between 70 and 130. We generated 100 trajectories per class, resulting in 200 for the `SR` dataset and 300 for the `Yao` dataset.

Kinematic-related synthetic datasets. We generated a simple synthetic dataset –`Fast Slow (FS)`– of moving objects turning 90 degrees right and traveling at two different speeds. By isolating a specific structural movement pattern while varying the travel speed, we aim to assess the effectiveness of vectorization techniques in capturing movement dynamics without confounding effects from other variables. The target labels associated with the trajectories could be either moving "slow" or "fast".

6. Experiments Results

In this section, we analyze how distinct vectorization captures structural patterns in the data under an unsupervised setting. Standard clustering practice often relies on the knee method or selects hyperparameters that maximize the silhouette (SIL) score. However, both heuristics depend on pairwise distances; hence, they are affected by the shape of the clusters. Additionally, clustering is applied in the vectorized space; therefore, these metrics can be used only when comparing clustering outcomes derived from the same vectorization type and configuration. In our framework, we compute the SIL score in the non-reduced vectorization spaces, enabling the use of this metric to compare clustering results across various feature reduction techniques. Additionally, since our datasets include class labels, we use them

		#trj	#obs	#obs per trj	Avg Δtime (s)	Classes
Real-world	Animals	102	15k	145 ± 62	4.85 ± 4.85	Cattle 33% Deer 29% Elk 37%
	Seabirds	108	264k	2.4k±1k	100 ± 0.0	tCOGU 29% tEUSH 14% tRAZO 57%
	Vehicles	381	178k	467 ± 251	122 ± 160	Bus 28% Truck 72%
	GeoLife	5977	5.3M	892 ± 2.5k	265 ± 2.9k	Public T. 43% Private T. 57%
Synthetic	SR	200	20*k*	100 ± 0.0	1.0 ± 0.0	Straight 50% Turn 50%
	FS	200	20*k*	100 ± 0.0	3.0 ± 2.0	Slow 50% Fast 50%
	Yao	300	21*k*	69 ± 12	1.5 ± 0.7	Straight 33% Circling 33% Bending 33%

Table 1
Datasets metadata.

as a proxy for ground truth and select, for each component of the pipeline, the hyperparameters that maximize the Fowlkes–Mallows (FM) score. Alternative heuristics to select the best clustering results without any prior knowledge are left as future work. SIL and FM scores alone do not characterize differences between feature spaces; they only quantify cluster separation and agreement with the labeled structure. To evaluate how the relative positions of instances vary across vectorizations of different dimensionality, we exploit the Triplet Loss introduced in [26], a widely used one-shot learning loss function that encourages embeddings in which similar instances are close together, and dissimilar instances are far apart. In our framework, we adopt a variant called Random Triplet Accuracy (RTA), proposed in [27], to measure the impact of feature reduction techniques. This measure operates by randomly sampling triplets of instances from two feature spaces and evaluating the proportion of triplets in which the relative distance ordering between instances is preserved across both spaces. A score near 1 indicates that both spaces encode instance relationships consistently, while a score near 0 suggests that they reverse these relationships, placing samples that are close in one space farther apart in the other. Figure 4 illustrates three scenarios. In the first, 90 instances are randomly projected into two unrelated feature spaces. Since RTA samples triplets at random, we repeat the evaluation 100 times and obtain an average score of ≈ 0.50, indicating that an ordering such as $\text{ED}(a, b) \leq \text{ED}(a, c)$ in one space fails to hold in the other in roughly half of the cases. In the second scenario, both vectorizations yield the same three clusters; however, the points are randomly arranged within each cluster, and the orange cluster is closer to different neighbors across the two spaces. Under the same experimental conditions, the resulting score is approximately 0.70. Please note that such a scenario would have the same FM score and approximately the same SIL score. Finally, in the third scenario, the vectorizations differ only by scale, leading to a score close to 1.

Tables 2 and 3 present the experimental results on vectorization diversity, reported through the RTA score, and clustering quality, reported through the SIL and FM scores. The optimal pipeline for each vectorization and dataset is selected using the FM score. Table 2 additionally reports the SIL score associated with the pipeline that maximizes the FM score, as well as the FM and SIL scores of the pipeline that performs best with respect to the SIL score, thereby simulating a selection process that relies solely on internal clustering criteria.

We begin by examining the upper part of Table 2, which reports the FM and SIL scores of the pipeline achieving the highest FM. To facilitate interpretation, FM and SIL scores are averaged across datasets with the same characteristics: shape-based, kinematic-based, and real-world trajectories. By

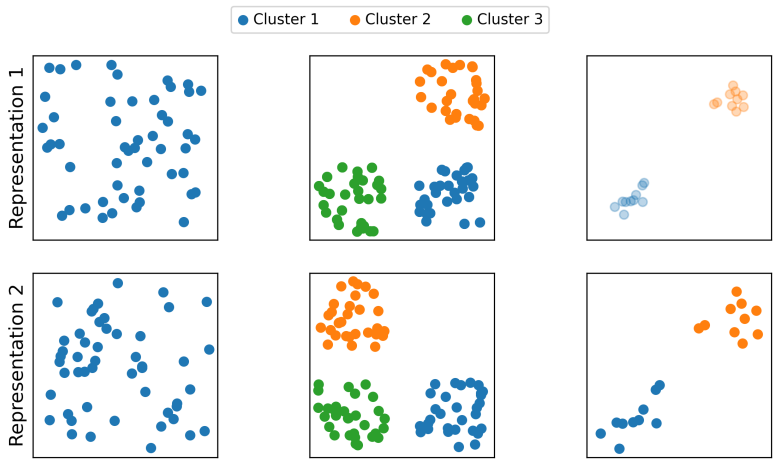

Figure 4: Comparison of RTA across three settings.

	FM				SIL			
	TIF	GEOLET	ROSITA	TF-IDF	TIF	GEOLET	ROSITA	TF-IDF
	Best on FM							
Geometry	*0.881*	0.880	**0.993**	-	0.350	*0.639*	**0.777**	-
Kinematic	**0.997**	0.709	*0.890*	-	**0.805**	*0.659*	−0.005	-
Real-world	*0.754*	**0.781**	0.675	0.744	−0.004	*0.437*	**0.480**	0.429
All	**0.877**	0.790	*0.852*	0.744	0.384	**0.578**	0.417	*0.429*
	Best on SIL							
Geometry	**0.881**	**0.881**	**0.881**	-	0.824	*0.867*	**0.888**	-
Kinematic	**0.993**	*0.702*	0.502	-	**0.945**	*0.778*	0.072	-
Real-world	**0.661**	0.603	*0.644*	0.605	0.832	*0.963*	**0.975**	0.927
All	**0.847**	*0.729*	0.675	0.605	0.867	*0.869*	0.645	**0.927**

Table 2
Experimental results on clustering quality. The top rows present FM and SIL scores for the clusters with the highest SIL score, while the lower rows report the same metrics for the clusters with the highest FM score.

selecting the pipeline that maximizes FM, TIF achieves the overall best performance across all datasets. Notably, ROSITA performs particularly well on geometry-based datasets, which aligns with its design. Moreover, GEOLET consistently attains the overall best SIL scores, indicating an average higher degree of cluster separation w.r.t. other vectorization techniques, which can be attributed to its sub-trajectory filtering strategy based on the *gap* score. For the other vectorizations, we do not observe any interesting pattern[4]. Instead, selecting the best pipeline that maximizes SIL yields slightly lower FM scores across all vectorization methods and dataset categories. On the other side, SIL scores are extremely high for most datasets and vectorization techniques. Our interpretation is as follows: we found pipelines with high FM or SIL scores, which means that, for some vectorization hyperparameters, each vectorization is able to discover clusters similar to the ground truth or produce vectorization spaces with well-separated clusters, but with lower similarity to the ground truth. While this highlights the flexibility of such vectorization in capturing higher-level mobility patterns beyond mere spatial proximity, it also makes it difficult for practitioners to determine the optimal hyperparameters without additional information about the data.

Despite that, several consistent trends emerge from the results in Table 2. The TIF vectorization

[4]We would like to recall the dependencies between SIL scores and vectorization spaces. As a result, we can only consider the single SIL score, comparison between vectorization methods or different datasets may be misleading

		TIF	GEOLET	ROSITA	TF-IDF
Geometry	TIF	1.000 ± 0.000	0.623 ± 0.111	0.614 ± 0.095	-
	GEOLET	0.623 ± 0.111	1.000 ± 0.000	0.741 ± 0.009	-
	ROSITA	0.614 ± 0.095	0.741 ± 0.009	1.000 ± 0.000	-
	TF-IDF	-	-	-	-
Kinematic	TIF	1.000 ± 0.000	0.512 ± 0.027	0.503 ± 0.014	-
	GEOLET	0.512 ± 0.027	1.000 ± 0.000	0.511 ± 0.011	-
	ROSITA	0.503 ± 0.014	0.511 ± 0.011	1.000 ± 0.000	-
	TF-IDF	-	-	-	-
Real-world	TIF	1.000 ± 0.000	0.608 ± 0.044	0.548 ± 0.012	0.583 ± 0.049
	GEOLET	0.608 ± 0.044	1.000 ± 0.000	0.509 ± 0.017	0.499 ± 0.034
	ROSITA	0.548 ± 0.012	0.509 ± 0.017	1.000 ± 0.000	0.592 ± 0.035
	TF-IDF	0.583 ± 0.049	0.499 ± 0.034	0.592 ± 0.035	1.000 ± 0.000

Table 3
Experimental results on clustering diversity via RTA.

demonstrates the highest overall FM values across almost all dataset categories. This indicates that the handcrafted feature extraction pipeline effectively captures both spatial and temporal properties of movement, producing clusters that closely correspond to the actual labeled classes. Conversely, GEOLET achieves superior performance on real-world datasets, where data are more complex, irregular, and noisy. This suggests that shapelet-based vectorizations are especially well-suited to capture localized movement variations or subpatterns (e.g., turns, stops, or loops) that often define behavior in natural or human mobility. The relatively lower SIL values, combined with high FM scores, indicate that, while GEOLET clusters may be less compact geometrically, they still align closely with the semantic ground truth. The ROSITA vectorization performs best on geometry-related datasets, as expected due to its design for rotation- and scale-invariant shape encoding. This confirms that geometric alignment is particularly important when trajectories exhibit clear structural regularities. However, ROSITA's performance drops slightly on datasets dominated by dynamic differences, suggesting its invariance properties may suppress meaningful kinematic variation. Finally, TF–IDF (dictionary-based) achieves competitive results on real-world datasets, suggesting significant potential for this technique.

Table 3 reports the RTA scores between pairs of vectorizations across all dataset categories. Vectorization hyperparameters follow the configuration that yielded the highest FM score. Overall, the scores obtained by comparing different transformation spaces average around 0.543, with values reaching up to 0.714 for specific dataset–vectorization combinations. As illustrated in Figure 4, entirely unrelated vectorizations typically yield a score of about 0.500, while points that fall into the same clusters but differ in their cluster-wise positioning tend to produce scores near 0.700. Interpreting these results benefits from revisiting Table 2, particularly the FM values. Since the FM score, defined as the geometric mean of precision and recall, ranges from 0 to 1 and reflects cluster similarity, its generally high values here suggest that most RTA scores correspond to cases where clusters are similar but differ in their internal positioning.

The highest RTA values (≈ 0.714) occur between GEOLET and ROSITA for geometry-based datasets, suggesting that although the methods rely on different principles, they capture similar global structure. In contrast, the lowest RTA values (≈ 0.500) appear between GEOLET and TF–IDF in real-world datasets, reflecting a fundamental difference between local shapelet patterns and symbolic discretization. These relationships highlight the complementarity among vectorization families. From a practical perspective, the reported RTA scores suggest promising research avenues for ensemble or hybrid vectorizations.

To qualitatively observe such complementarity, we now examine representative embedding spaces for the selected datasets. Figure 5 presents 2D PCA projections for TIF, GEOLET, and ROSITA (left to right). For the SR dataset (top row), all three embedding spaces clearly separate instances by their true clusters. For FS (bottom row), where classes differ only in movement speed, TIF yields a clean

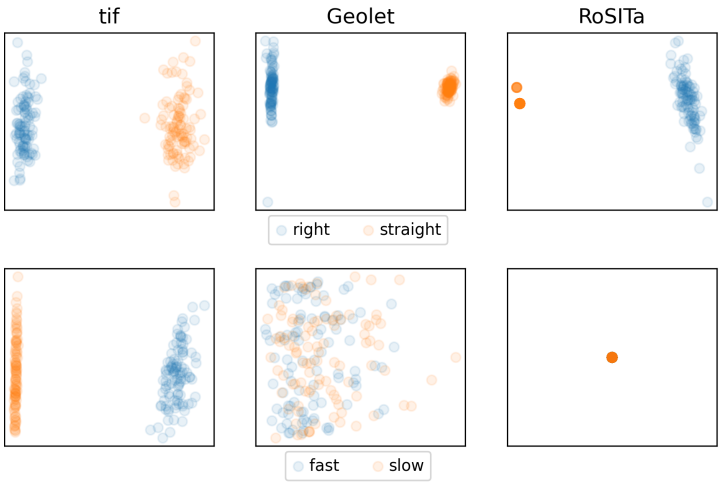

Figure 5: 2D PCA projections of the vectorization spaces produced by TIF, GEOLET, and ROSITA across the SR and FS datasets. Colors indicate ground-truth clusters.

separation, whereas the 2D projection of the ROSITA space collapses the instances into a single point. Nonetheless, its FM score is nearly optimal, indicating that the collapse is a visual artefact of PCA. The GEOLET projection also appears scattered randomly. However, its FM score of 0.705 suggests that, despite the imperfect visualization, meaningful separation persists in this balanced binary setting.

7. Conclusions

We introduced a benchmark framework for evaluating alternative trajectory vectorizations for unsupervised clustering. By comparing representative vectorization families on real-world and synthetic datasets, we analyzed how sub-trajectory dynamics and geometry are reflected in different vector spaces. The results show that vectorization choice substantially affects clustering, that methods performing well under controlled conditions may not generalize to noisy real-world data, and that different vectorizations capture complementary (not redundant) information, as indicated by moderate RTA values. We also observe that relying on a single metric (e.g., SIL or FM) can be misleading, whereas combining internal and external criteria supports more robust model selection. At a finer level, feature-based methods are effective for motion dynamics, shapelet-based methods capture local geometric variation, and ROSITA performs strongly on geometry-dominated datasets. These findings motivate future work on multi-vectorization ensembles, fusion strategies, and broader evaluation across more datasets, vectorizations, and clustering algorithms.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT-4 and Grammarly to perform grammar and spelling checks. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

Acknowledgments

This work has been partly supported by the University of Piraeus Research Center.

References

[1] S. A. Ahmed, D. P. Dogra, S. Kar, P. P. Roy, Trajectory-based surveillance analysis: A survey, IEEE Trans. Circuits Syst. Video Technol. 29 (2019) 1985–1997.

[2] A. Hamdi, K. B. Shaban, A. Erradi, A. Mohamed, S. K. Rumi, F. D. Salim, Spatiotemporal data mining: a survey on challenges and open problems, Artif. Intell. Rev. 55 (2022) 1441–1488.

[3] M. Musleh, M. F. Mokbel, Let's speak trajectories: A vision to use NLP models for trajectory analysis tasks, ACM Trans. Spatial Algorithms Syst. (2024).

[4] M. F. M. et. al., Towards mobility data science (vision paper), CoRR abs/2307.05717 (2023).

[5] C. L. da Silva, L. M. Petry, V. Bogorny, A survey and comparison of trajectory classification methods, in: BRACIS, 2019, pp. 788–793.

[6] C. Landi, F. Spinnato, R. Guidotti, A. Monreale, M. Nanni, Geolet: An interpretable model for trajectory classification, in: IDA, volume 13876 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 236–248.

[7] Y. Zheng, Trajectory data mining: An overview, ACM Trans. Intell. Syst. Technol. 6 (2015) 29:1–29:41.

[8] M. Middlehurst, A. Ismail-Fawaz, A. Guillaume, C. Holder, D. Guijo-Rubio, G. Bulatova, L. Tsaprounis, L. Mentel, M. Walter, P. Schäfer, A. J. Bagnall, aeon: a python toolkit for learning from time series, CoRR abs/2406.14231 (2024).

[9] M. Etemad, A. S. Júnior, S. Matwin, Predicting transportation modes of GPS trajectories using feature engineering and noise removal, in: Canadian Conference on AI, volume 10832, 2018, pp. 259–264.

[10] L. Ye, E. J. Keogh, Time series shapelets: a novel technique that allows accurate, interpretable and fast classification, Data Min. Knowl. Discov. 22 (2011) 149–182.

[11] C. A. Ferrero, L. O. Alvares, W. Zalewski, V. Bogorny, MOVELETS: exploring relevant subtrajectories for robust trajectory classification, in: SAC, 2018, pp. 849–856.

[12] C. Chen, H. Su, Q. Huang, L. Zhang, L. J. Guibas, Pathlet learning for compressing and planning trajectories, in: SIGSPATIAL/GIS, ACM, 2013, pp. 382–385.

[13] C. Landi, R. Guidotti, Shape-based methods in mobility data analysis: effectiveness and limitations, GeoInformatica (2025).

[14] J. Bian, D. Tian, Y. Tang, D. Tao, A survey on trajectory clustering analysis, arXiv (2018). arXiv:1802.06971.

[15] J. Lee, J. Han, X. Li, H. Gonzalez, *TraClass*: trajectory classification using hierarchical region-based and trajectory-based clustering, Proc. VLDB Endow. 1 (2008) 1081–1094.

[16] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. A. F. de Macêdo, N. Pelekis, Y. Theodoridis, Z. Yan, Semantic trajectories modeling and analysis, ACM Computing Surveys 45 (2013) 42:1–42:32.

[17] I. Ben-Gal, S. Weinstock, G. Singer, N. Bambos, Clustering users by their mobility behavioral patterns, ACM Transactions on Knowledge Discovery from Data 13 (2019) 45:1–45:28.

[18] C. Landi, R. Guidotti, M. Nanni, A. Monreale, The trajectory interval forest classifier for trajectory classification, in: SIGSPATIAL/GIS, ACM, 2023, pp. 67:1–67:4.

[19] C. Landi, N. Andrienko, G. Andrienko, Rotation- and scale-invariant shape extraction from vessel trajectories for human-in-the-loop monitoring, in: SIGSPATIAL/GIS, ACM, 2025.

[20] J. Zakaria, A. Mueen, E. J. Keogh, Clustering time series using unsupervised-shapelets, in: IEEE Computer Society ICDM, 2012, pp. 785–794.

[21] J. Lin, E. J. Keogh, L. Wei, S. Lonardi, Experiencing SAX: a novel symbolic representation of time series, Data Min. Knowl. Discov. 15 (2007) 107–144.

[22] I. S. Suwardi, D. Dharma, D. P. Satya, D. P. Lestari, Geohash index based spatial data model for corporate, in: ICEEI 2015, IEEE, 2015, pp. 478–483.

[23] M. Vlachos, Z. Vagena, P. S. Yu, V. Athitsos, Rotation invariant indexing of shapes and line drawings, in: CIKM, ACM, 2005, pp. 131–138.

[24] C. Landi, R. Guidotti, Shape-based methods in mobility data analysis: effectiveness and limitations,

GeoInformatica 29 (2025) 999–1032.
[25] D. Yao, C. Zhang, Z. Zhu, J. Huang, J. Bi, Trajectory clustering via deep representation learning, in: IEEE IJCNN, 2017, pp. 3880–3887.
[26] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 815–823.
[27] Y. Wang, H. Huang, C. Rudin, Y. Shaposhnik, Understanding how dimension reduction tools work: An empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization, J. Mach. Learn. Res. 22 (2021) 201:1–201:73.