# Detection of anomalous trajectories for vehicle traffic data

Angelos Moavinis[1,*], Anastasios Gounaris[2,*] and Ioannis Constantinou[1]

[1] *Istognosis Ltd., Nicosia, Cyprus*
[2] *Aristotle University of Thessaloniki, Greece*

Abstract
Modern GPS recording devices and big data infrastructures enable us to study traffic patterns in a multitude of ways, including trajectory outlier detection. A trajectory is a sequence of consecutive geographical points that can also have timestamps and an outlier is defined as a record, which in our case is a trajectory, that significantly differs from the norm. In this work, we are motivated by the need (i) to encapsulate trajectory outlier detection in real-life Fleet Management Systems (FMSs) and (ii) to improve the performance of existing outlier detection methods. To this end, two trajectory outlier detection methods are proposed, the first one relying on the DBSCAN clustering algorithm and the Hausdorff distance and the second one relying on a Support Vector Machine (SVM) classifier and the Generalized Sequence Pattern algorithm. These two algorithms are evaluated against baselines on two automatically labeled traffic datasets, the former from the Beijing metropolitan area and the latter from a real FMS in Cyprus. Automated labelling process is adopted to both allow for reproducibility and lift the burden of manual annotations from domain experts. The results show that our proposals exhibit better performance than the baselines in terms of accuracy and the F1 score, while, in general, the SVM model performs better than the path clustering one. Finally, traffic clusters and specific outliers are discussed to prove the validity of the models.

**Keywords**
anomaly detection, trajectories, real-world dataset

## 1. Introduction

With the massive traffic data generation in urban and rural street grids, the rapid growth of urbanization and also the geolocation functionality embedded in virtually all portable devices such as smartphones, wearables and navigation devices, the need and opportunity for performing traffic analytics arises, mostly on urban traffic data since the traffic in cities is significantly more massive and chaotic than in rural areas. Traffic analytics comprises a broad range of techniques that include, among others, spotting traffic patterns and, in a complementary manner, detecting anomalies in the traffic data.

Targeted anomalies can either be *individual flows*, *sub-trajectories* or entire *trajectories* [1]. A flow is defined as the number of objects moving on the road section from location A to location B on a timestamp or in a certain time-slot. Trajectories are defined as sequences of locations that a vehicle passes through and can also have a timestamp for each point, thus making them spatiotemporal sequences. A sub-trajectory is a slice of a larger trajectory, which implies that a larger trajectory can be split into sub-trajectories to study traffic patterns and outliers at a finer detail.

There are several areas that utilize traffic data outliers, which highlights the value of this research field. Example use cases are the following:

- **Urban traffic detection:** The most obvious use of trajectory outlier detection is the detection of traffic patterns and anomalies. Two research works that deal with this use case are the one authored by Kong et al. [2], who study anomalous areas with long-term traffic issues by analyzing bus route data, and the one by Djenouri et al. [3], who analyze the distribution of flow outliers in the city of Odense, Denmark.
- **Fraud and crime detection:** This includes taxi drivers who intentionally take longer routes than needed in order to increase the cost for the clients [4] or do unmetered trips and overcharge without getting noticed [5]. Another representative case is the detection of illegal maritime activity [6].
- **Abnormal weather patterns:** An example of this category is detecting outlier trajectories and sub-trajectories of storms, typhoons and other meteorological data, like in the work of Lee at al. [7], who attempt to detect outlying sub-trajectories from real hurricane trajectory data.
- **Animal Movement Analysis:** Unusual animal movements, which do not conform to an expected pattern, are frequently of great interest to biologists and botanists. The approach of Li et al [8] utilizes animal trajectory data to discover animal movement patterns.

The focus of this research work is on trajectory outlier detection using real vehicle mobility data, where we aim

*Published in the Workshop Proceedings of the EDBT/ICDT 2023 Joint Conference (March 28-March 31, 2023, Ioannina, Greece)*
*Corresponding author.
aggelos@navarchos.com (A. Moavinis); gounaria@csd.auth.gr (A. Gounaris); ioannis@istognosis.com (I. Constantinou)
0000-0003-3976-7615 (A. Gounaris)

to yield solutions with higher performance than existing methods while allowing for reproducibility. We start with explaining the categories in traffic flow outlier detection systems.

A main criterion to divide trajectory outlier detection systems is based on *online* versus *offline* processing. Online systems are capable of producing their output in *real-time*, for example in real-time traffic reporting applications, while offline systems are applied on datasets to extract post-mortem insights from them. The online methods are used for processing data streams in real time, since they can provide insights about shorter parts of the route and proceed to recommendations on the fly, while the offline methods do not provide such fine-grained analysis, especially if they are not combined with an additional post-processing analysis step after the outlier detection. Our work belongs to the latter category, according to the main requirements in real Fleet Management Systems (FMSs) by which we are motivated.

The three families of traffic flow outlier detection systems are distinguished by Djenouri et al. [1] as (i) *statistical methods*, which use statistical measures and theory to find anomalous traffic flows; (ii) *similarity methods*, which apply neighbourhood computations and clustering for the same purpose; and finally (iii) *pattern mining methods*, which use and extend well-known algorithms such as FP-Growth and Apriori, and similar pattern-mining approaches.

Another way to process trajectories is viewing them as *event sequences*, where events correspond to an object passing through a crossroad, or a specific coordinates point or a grid area (in case the area studied is split into grids to group same-area GPS points). Based on this rationale, event sequence mining methods such as [9] can be useful; these methods find frequent and infrequent (outlying) sequence patterns in datasets.

## 1.1. Addressed problem

A trajectory, which, as mentioned earlier, is a sequence of points, can be labeled as an outlier because of many reasons. The difference or extremeness and rarity that the definition of the outlier implies can be attributed to various features that come with a trajectory and are not only limited to the spatial aspect of it but can also include its temporal features and new features that arise from, or even go beyond the combination of spatial and temporal features. In this work, we are interested in detecting outliers due to their spatial features, i.e., outliers are those trajectories that are different from others in terms of the location coordinates of the points they comprise. Note that we do not explicitly set a distance function and/or a distance threshold.

An important matter is the comparison of the proposed methods against other trajectory outlier methods. Unfortunately, most published papers do not include links to the code used by the researchers, so, for the needs of a benchmarking comparison among different models, an implementation of a few already proposed models is needed. This benchmarking should also include methods from the event sequence mining area, as reasoned above. However, the biggest problem is not the lack of free implementations of several existing methods, but the lack of labeled data and data that has temporal features for trajectories. Regarding labeled data, very few publications include links to publicly available datasets and even fewer include labeled datasets or the ground truth. For papers that include unlabeled public datasets, most authors use an annotation by field experts as ground truth, which makes their results hard or even impossible to reproduce. As for the labeled datasets, there has to be more information (temporal features) for the datasets to be more complete. Thus, an automated annotation method is needed so that there can be produced (real-world) datasets that are accompanied by ground truth information, so that different trajectory outlier techniques can be objectively compared against each other. In our work, we adopt an established method for automated labelling so that the main experiments are objective.

## 1.2. Contribution and Structure

The main contribution of this work is twofold. More specifically, we propose two novel techniques for trajectory outlier detection. The one utilizes density-based clustering, as several other proposals do, but employs and investigates the usage of Hausdorff distance. Our argument is that the distance metric does make a big difference. The other proposed technique combines classification with sequential pattern mining. Both solutions have not been explored in the literature yet. Additionally, as already mentioned, we apply a systematic and automated technique to assign ground truth labels regarding outlierness to trajectory datasets so that the corresponding techniques can be objectively compared. Our evaluation results using two real-world datasets reveal the superiority of the Hausdorff distance over Euclidean in the generic case, but overall, the combination of classification with sequential pattern mining performs better. The F1-score improvements against state-of-the-art baselines is up to 27.31%.

A significant part of this work has appeared in a dissertation[1], which was then applied to data from a real FMS, namely Navarchos[2], which is a product of the Istognosis company in Cyprus[3] and serves several clients on a daily basis. The code used along with one dataset is freely

[1]http://ikee.lib.auth.gr/record/342389/files/GRI-2022-36973.pdf
[2]https://navarchos.com/
[3]http://www.istognosis.com/

**Table 1**
Overview of main existing methods

| Name | Outlier Type | Approach |
|---|---|---|
| TRACLUS[10] | Sub-trajectories | Clustering |
| Eldawy[11] | Trajectories | Clustering |
| Zhongjian et al.[12] | Trajectories | Clustering |
| Roman et al.[13] | Sub-trajectories | Clustering |
| iVAT+[14] | Trajectories | Clustering |
| clustiVAT+[15] | Trajectories | Clustering |
| TPRO[16] | Trajectories | Clustering |
| TPRRO [17] | Trajectories | Clustering |
| Zhang[4] | Trajectories | Statistics |
| MT-MAD[18] | Trajectories | Statistics |
| Lan et al.[19] | Sub-trajectories | Statistics |
| Liu et al.[20] | Traffic Flows | Statistics |
| Bao et al.[21] | Sub-trajectories | Statistics |
| TRAOD[7] | Sub-trajectories | Density |
| DBTOD[22] | Sub-trajectories | Density |
| LDTRAOD[23] | Sub-trajectories | Density |
| Yu et al.[24] | Sub-trajectories | Density |
| Djenouri et al[3] | Traffic Flows | Density |
| Djenouri et al.[25] | Traffic Flows | Density |
| LoTAD[2] | Sub-trajectories | Density |
| TF-Outlier[26] | Sub-trajectories | Density |
| Maiorano[27] | Sub-trajectories | Other |
| Varlamis et al.[6] | Sub-trajectories | Other |
| TOP[9] | Trajectories | Other |
| iBAT[28] | Trajectories | Other |
| DeepTEA[29] | Trajectories | Other |
| **Our proposal** | **Trajectories** | **Clustering + Other** |

available.[4]

The structure of the paper is as follows. In Section 2, the relevant literature is discussed. In Section 3, the proposed methods are presented. Section 4 deals with the evaluation results and Section 5 includes the conclusions from this work and discussion of future work.

## 2. Related Work

We organize the related work according to the algorithmic approaches followed, whereas a summary is presented in Table 1. The focus is on trajectory outliers, however the table contains a broader range of proposals.

**Clustering-based methods**: Trajectory outlier detection methods that rely on clustering usually apply a common clustering algorithm, such as DBSCAN, k-means, k-medoids, and employ a different distance function than the euclidean distance or Manhattan distance that are usually applied in most clustering libraries and application scenarios. In [30], which is extended by [31], the authors apply trajectory clustering among other methods, in order to find trajectory outliers. The clustering method that they use is the one of [10], which is based on the DBSCAN algorithm. They break-down trajectories in segments in an optimal way that maximizes preciseness and conciseness by using the minimum description length (MDL) principle. In a similar way, Eldawy and Mokhtar[11] propose a method to detect trajectory outliers that also uses the DBSCAN algorithm and the MDL principle. DBSCAN and k-means are also used in the proposed method of [12]. Other algorithms proposed include iVAT, iVAT+, clustiVAT, clustiVAT+ ([14][15]), and CaD ([13]) that uses the TSA [32] algorithm to break down the trajectories.

**Statistics-based methods**: Statistics-based methods adopt the basic rationale that if the features extracted from a data point (such as distance, speed, direction, traffic condition) exceed a threshold or deviate enough from the norm, it is an outlier. For example, the authors of [20] detect outlying traffic flows by measuring their extremeness in terms of deviation from the frequent transition volume between street network nodes and from the traffic in previous days and weeks. In [4], the authors detect outlier trajectories based on their length deviation from the optimal path's deviation for each trajectory. In [19], a sub-trajectory is marked as outlying if it deviates significantly from other sub-trajectories in terms of traffic volume. Other methods in this category include [16] [17] [21] [18].

**Density-based methods**: Density-based methods leverage the basic idea is that if a data point lies in a low-density region, then it must be some kind of an anomaly. This may look similar to the clustering based methods, however it is different since there are no frequent group-forming general patterns detected by density-based methods. The TRAOD algorithm[7] is a partition-and-detect method that uses the distance and density measures for trajectory fragments and is used as an inspiration for multiple research works, such as [22] and [23]. [24] uses a k-neighbor strategy to detect outlier sub-trajectories and leverages the duration in which a trajectory is an outlier or an inlier. [3] also follows a kNN approach. Other interesting approaches are these in [25] [2] and [26].

**Other methods**: This part includes methods that do not fit in the prior classification. The work of [6] uses graph theory to produce a network abstraction of maritime vessels' trajectories. The authors of [9] propose an event sequence outlier detection method that uses a pattern mining algorithm to detect outlying event sequences. This approach can be generalized to trajectories by representing a transition through a road section as an event. The algorithm returns contextually frequent patterns and contextual outliers by employing a pattern mining algorithm called Reduce. [27] apply a rough (uncertain) set theory-based approach to detect outlying sub-trajectories based on the number of outlying points a trajectory contains. [29] is a probabilistic model that uses deep neural and convolutional networks to detect time-dependent outliers and handle complex traffic conditions. Finally, the authors of [28] use the Isolation Forest algorithm to

[4]https://github.com/amoavinis/trajectory-outliers

detect outlying trajectories.

How does our proposal compare to the above work? The answer is that it is a novel hybrid solution for trajectory outlier detection. It utilizes the DBSCAN clustering algorithm, similarly to [30], [31], [10], [11] and [12]. But it goes beyond clustering and also employs a SVM classifier, and optionally, a sequential pattern mining component (GSP). More importantly, it investigates the impact of Hausdorff distance when employing DBSCAN. Since we extend clustering-based solutions, the evaluation presented in Section 4 focuses on (i) evaluating the effectiveness of such extensions; and (ii) evaluating the improvements upon other solutions for trajectory outlier detection, such as the ones presented in [9] and [4]. The proposals in [9] and [4] fall in the “Other" and “Statistics" approach categories, respectively. Moreover, the proposal in [9] also leverages sequential pattern mining. Due to the different focus, we do not directly compare with the other trajectory outlier detection techniques in the “Other" category. The work in [6] is more tailored to vessels and relies on attributes such as speed, bearing and bearing rate, and percentile values. The proposal in [29] focuses on time-dependent anomalies, which we leave for future work. Finally, iBAT [28] uses grid cells as we do, but focuses on trajectories including rare points, which is not the case we target.

## 3. Proposed Method

Our proposal employs (sequential) pattern mining, DBSCAN-based clustering and SVM-based classification. In the sequel, we describe each component separately. The novelty is in the usage of the Hausdorff distance. Then, we explain two approaches to combining our proposals and, at the end of the section, we summarize the novel aspects.

### 3.1. Auxiliary Techniques

**Grid partitioning** is used in order to discretize and compact trajectories. Because each trajectory consists of hundreds or thousands of GPS points, a scalable system can largely benefit from a significant reduction of the input size, since operations that require calculations between all points of two trajectories are of quadratic complexity and thus can experience speedups of several orders of magnitude. Grid partitioning simply divides the 2D plane into rectangular grid cells and each trajectory forming a sequence of GPS points gets mapped to a sequence of grid cells. Numerous consecutive GPS points fall in the same grid cell, which is recorded only once, thus greatly reducing the size of the trajectory.

Additionally, the **GSP** algorithm (Generalized Sequential Pattern) [33], which is based on the Apriori algorithm, is used for sequence mining. It includes two main steps that can run iteratively, namely candidate generation and support counting. It returns frequent sub-sequences of the dataset’s sequences.

### 3.2. Path Clustering

The path clustering method that we adopt uses the DBSCAN algorithm to cluster the trajectories based on the similarity of their paths. In our context, two trajectories are considered close to each other if their paths are ‘close’ to each other. This closeness is measured with two different distance functions, the Hausdorff distance and the DTW distance.

Roughly speaking, two trajectories, each represented as a sequence of points, are close with regard to the Hausdorff distance $D_H$ if every point of either trajectory is close to some points of the other trajectory. The formula is:

$$D_H(L_i, L_j) = max(h(L_i, L_j), h(L_j, L_i))$$

$$h(L_i, L_j) = \max_{a \in L_i}(\min_{b \in L_j}(dist(a, b)))$$

with *dist(a, b)* being the Euclidean distance between points $a$ and $b$.

The Hausdorff distance between trajectories $L_i$ and $L_j$ returns the maximum unidirectional Hausdorff distance from $L_i$ to $L_j$ and from $L_j$ to $L_i$. It measures the maximum degree of mismatching between two trajectories.

Hausdorff distance is extremely sensitive to noise. For example, a single point of $L_j$ far away from $L_i$ will result in a large distance value. In this case, the calculated distance is unable to represent actual difference between trajectories. Employing the Hausdorff distance is a novel element of our solution, since, to the best of our knowledge, it has not been used for traffic outlier detection in any published work known to the author. This distance function has quadratic complexity because it requires the calculation of all distances between the points of the two trajectories. So, given two trajectories with lengths $n$ and $m$, the time complexity is $O(nm)$. For this reason, the length of the trajectories should be decreased in such a way as to retain the properties of the paths and at the same time to not make the distance computation too time-expensive. This is achieved by grid-partitioning the 2D plane and then converting the trajectories into sequences of grid cells, as already explained.

Dynamic Time Warping (DTW) is a technique proposed by Sankoff and Kruskal to find the optimal alignment between two time-dependent sequences [34]. It is suitable for matching trajectories of different length.

The sequences are "warped" non-linearly with regards to time to calculate a measure of their similarity independently of certain non-linear variations in the time dimension. The formula to calculate the DTW score between a trajectory A of length $m$ and a trajectory B of length $n$ is:

$$\text{DTW(A, B)} = \begin{cases} 0 & m = n = 0 \\ dist(A[-1], B[-1]) & \\ +min\begin{cases} DTW(Rest(A), Rest(B)) \\ DTW(Rest(A), B) \\ DTW(A, Rest(B)) \end{cases} & m * n > 0 \\ \infty & else \end{cases}$$

where *Rest(L)* is the trajectory $L$ without its final point and *dist* is the distance function used to compute the deviation between two points of the trajectory (Euclidean distance). The complexity of this algorithm is $O(n * m)$, i.e., the same complexity as the Hausdorff distance.

In addition, we may want to experiment with unidimensional trajectories. A mathematical method to transform 2D sequences into single dimensional ones is the Hilbert curve, which falls into the larger family of space-filling curves. A space-filling curve is a curve whose range contains the entire 2-dimensional unit square. The Hilbert curve is a space-filling curve that adequately preserves point locality. The curve is firstly constructed and each point on it corresponds to a number, which starts from 0 at the start of the curve and increases linearly as the curve progresses. For any 2D point that is to be converted to a Hilbert curve 1D representation, the nearest point of the curve is found and the point is mapped to its number value. This notion is used to calculate a distance between 2D points as a substitute for the Euclidean distance. In this paper, a third distance function is used for the Path Clustering method, which is the same as the already mentioned DTW distance but uses this Hilbert distance instead of the Euclidean distance between points. In detail the distance of 2 points is $abs(Hilbert(a) - Hilbert(b)$, where $a$ and $b$ are the two points, $abs$ is the absolute value function and $Hilbert$ is the function that converts the 2D point into a 1D Hilbert curve representation. For both DTW variations, the grid cell conversion of trajectories that happens before the Hausdorff distance is calculated is also applied.

## 3.3. SVM approach

For each trajectory, multiple features can be extracted. In this proposal, we build a classification model using the following features:

1. *Starting coordinates*: converted from latitude-longitude to min-max-normalized values from 0 to 1.
2. *Destination coordinates*: similar to starting coordinates.
3. *Distance*: the total distance of the trajectory. The length of each path section between two points is calculated and the lengths are aggregated into a total length of the trajectory.
4. *Deviation from closest frequent subsequence*: the Hausdorff distance of the trajectory and its closest frequent sub-trajectory, as calculated by the GSP algorithm.

In order to train the SVM on these features, first they have to be min-max scaled in the [0-1] range. In order to obtain the last feature, namely the deviation from the closest frequent subsequence, the Generalized Sequential Path Mining (GSP) is used. For this final step, the GSP model first learns the frequent sub-trajectories by using the *grid-converted* trajectories, and the Hausdorff distance calculation between a trajectory under evaluation and a grid representation frequent sub-trajectory requires the trajectory to be also converted to its grid cell representation.

## 3.4. Combining the two solutions

We explore if a combination of the outputs of the two solutions, the one based on path clustering and the SVM one, will perform better, two approaches are examined. The first one involves a simple logistic regression model, which is applied on the results of the two solutions. It takes 2 inputs (the 0 or 1 value from each model) and produces 2 outputs, one for each class. In detail, the outputs of the path clustering and the SVM model, which (for each instance) are two single integers that can be 0 or 1, are stored and then used as input to train the logistic regression model, which is trained with the ground truth; 0 corresponds to the 'inlier' class and 1 to the 'outlier' class. The second combination approach is a logical OR operator applied on the outputs. For a single trajectory, if either the clustering or the SVM model labels it as an outlier, the system decides that it is an outlier; else it is an inlier. The above process implies that we are interested in binary labelling of the trajectories. Quantifying the degree of outlierness is left for future work, although it is straightforward to extend our techniques to produce outlier scores rather than just binary labels.

## 3.5. Summary of our proposal and its novelty

In a nutshell, we investigate a solution consisting of (i) a path clustering module that uses the DBSCAN algorithm and the Hausdorff distance, and (ii) a SVM classifier module that uses the starting and ending coordinates, trip distance and Hausdorff distance from nearest GSP-generated sub-trajectory as features. The output of these two modules can be combined. The novelty of our work is the usage of the Hausdorff distance as a distance metric for the DBSCAN algorithm and in the features fed to the

SVM-based solution, which yields a novel proposal for employing a SVM model and leveraging the GSP algorithm in order to find the nearest frequent sub-trajectory for each trajectory. Furthermore, these novelties can be employed in combination. As the following experiments suggest, our solution improves on the state-of-the-art in real traffic datasets.

## 4. Experimental Evaluation

To evaluate our techniques, we need to employ ground truth. To this end, and since there are no publicly available traffic datasets that are suitable for assessing the performance of trajectory outlier detection solutions, we first explain how such ground truth can be yielded in a systematic manner. The provision of labels is a significant by-product of our work and a useful contribution for other researchers and practitioners in the same field. Then, we explain our competitors and finally, we present our experiments that focus on assessing the effectiveness of the outlier detection process using the accuracy and F1 measures.

### 4.1. Datasets and systematic labeling

Two datasets were used to evaluate this research work. The first one is the Geolife dataset[35][36][37], created by the Geolife project of Microsoft Research Asia. The data was collected from 182 users in the time period from April 2007 to August 2012 in the area of Beijing. A trajectory of this dataset is a sequence of time-stamped points. For each point, the features include latitude, longitude and altitude. This dataset contains 17,621 trajectories, running a total distance of approximately 1.2 million kilometers. The trajectories were recorded by using various GPS tracking devices and are thus of different sampling rates. A broad range of outdoor movements are included, such as commuting, sports activities, shopping, sightseeing, hiking, and cycling.

We first proceed with dataset cleaning. The dataset includes trajectories with data points that are not valid GPS coordinates or are produced by wrong measurements and appear to be very far from the rest of the dataset points, e.g. a latitude value of 440 (which is not a valid value) or 30, which is unreasonably far from Beijing. Also, there are few trajectories that reach very far from the area of the city; including such datasets in the test dataset would shift the outlier detection away from detecting inner-city outlying trajectories and towards detecting peripheral outliers, while this work chooses to study the trajectories that are in the close vicinity of the city. Thus, the dataset is filtered so that it includes only trajectories with points with a latitude from 39.65 to 40.3 degrees and a longitude of 116.1 to 116.6 degrees.

The second dataset is provided by Istognosis Ltd. and is not publically available. It is referred to as the “Cyprus" dataset. It contains approximately 17K trajectories from the company’s customer base. The main trajectory type it includes is movement of cargo vehicles that serve retail stores by providing shopping goods from distribution centers. The total length of the trajectories is 151000 kilometers, making each trajectory significantly shorter (on average) than the Geolife trajectories.

Both datasets, in their original format, just include coordinates and timestamps for the trajectories and do not contain labels for the training of a supervised trajectory outlier detection system. So, a systematic, impartial and reproducible labeling process has to take place for the usage of the dataset in the evaluation of the proposed method. In addition, the labeling process has to be automated because the options of manual labeling by the dedicated user groups and/or experts is too resource-consuming. In our approach, the process followed by Wu et al.[38] for the automatic labeling of their dataset is applied. They first partition the dataset into start-destination pair-based partitions, and the ones that contain fewer trajectories than a threshold *minThr* are filtered out. In this implementation, a start-destination pair is not defined by the coordinates but by the grid square that the points fall in. After this filtering, the removed trajectories are marked as outliers and for each remaining partition, the following process is followed. A complete linkage agglomerative clustering is performed with the Jaccard distance as the distance function. The exact formula is:

$$J(R_a, R_b) = 1 - \frac{|(R_a \cap R_b)|}{|(R_a \cup R_b)|}$$

for routes $R_a, R_b$; the routes refer to grid cell sequences as already explained.

After the clustering, for each start-destination partition the clusters that are of relative size to their respective partition’s size greater than a ratio threshold *thr* are marked as inliers; otherwise, they are marked as outliers. Note that this labeling technique relies on spatial information without taking into account the temporal information; so the outlier ground truth produced refer only to the spatial aspect of the trajectories.

In our implementation, the grid partitioning parameter is set to 5 for the Geolife dataset and 10 for the Cyprus dataset, the complete linkage clustering parameter to stop cluster merging is set to 0.4 for both, the *minThr* parameter is 15 and the *thr* parameter is 0.03. These parameters are set so that the outlier ratio is near 5% of the dataset.

## 4.2. Competitors

The chosen baseline models are the ones of [9] and [4]. The former is named TOP both in the original paper and also in our evaluation. The latter is originally based on Djikstra shortest path calculation with a significant speedup from the usage of the Contraction Hierarchies algorithm. In our implementation though, the Contraction Hierarchies is not used. The reason is that we are mostly interested in the accuracy and F1 measure of the solutions. Since it is a variation of the originally proposed algorithm, it is named DODB in the results presented (**D**ijkstra **O**ptimal **D**istance **B**ased Trajectory Outlier Detection System).[5]

## 4.3. Evaluation setting

All techniques discussed were implemented in Python 3.10.6. The experiments were conducted on a 12th Gen i7-12700K processor with 64GB of RAM and a NVMe SSD drive, with the system running on the Ubuntu 22.04 OS.

For the experiments, the datasets were split into a 75-25% train-test ratio and converted into a grid representation. The accuracy and F1 Score are measured for each experiment and presented.

The configuration is given in the repository and is performed in such a manner that the highest accuracy is achieved. For the clustering module, the grid parameter is 40 for the Cyprus dataset and 20 for the Geolife dataset. The *epsilon* parameter is set to 1.5 and *minPts* to 5. *minPts* is set to 20 for PC1-Geolife, 2 for PC2/PC3-Geolife, 30 for PC1-Cyprus, 5 for PC2-Cyprus and 2 for PC3-Cyprus. For the SVM module, *C* is set to 8000, *gamma* is set to 'scale' as per the *sklearn* documentation, the SVM kernel is the RBF one and the GSP support parameter is 0.05, with the grid parameter being also used if the GSP algorithm is ran and is set at 40 for the Cyprus dataset and 20 for the Geolife.

Since the DBSCAN algorithm does not have a prediction function for instances that are not included in the train set, a custom one is applied here. The training labels are obtained from the fitting process and for each test instance, its nearest neighbor in the training set is found and its class is assigned to the instance.

## 4.4. Results

The main effectiveness results for both datasets are summarized in Table 2. PC1 is the Path Clustering solution with the Hausdorff distance, PC2 is the Path Clustering solution with the DTW distance, PC3 is the Path Clustering solution with the DTW-Hilbert distance and SVM is the plain SVM-based model. Wherever GSP is mentioned, it is used as a feature for the SVM solution's input data. Finally, the SVM+GSP+PC1(LR) model is the LogReg-combined model, as described above, and the SVM+GSP+PC1(OR) is the logical OR ensemble model.

The results show that in general F1 and accuracy scores are high and our proposals can significantly improve upon existing state-of-the-art solutions. The SVM+GSP combinations yield the best results in terms of both accuracy and F1. When combined with the PC1 model and the LogReg method, the total performance does not increase for either dataset. If the logical OR is applied instead, the accuracy and F1 drop for both datasets. However, PC1 is significantly more effective than PC2 that employs the Euclidean distance, and also DTW, since trajectories are of different length in general. Finally, both proposed models outperform the baseline. More importantly, the improvements are larger with regards to F1, which is more suitable for an outlier detection problem. Our solutions improve F1 by up to 21.26% and 27.31% for the Geolife and the Cyprus datasets, respectively. The corresponding accuracy improvements are 5.7% and 9.6%.

For all models, the execution times (including both fitting and predicting) are mentioned in Table 3. By examining this table, it can be observed that the TOP model is the fastest solution overall, while DODB is a computationally expensive model due to the calculation of optimal distances (i.e., it does make sense for performance reasons to resort to the implementation in [4]). From the proposed models, the DTW-based Path Clustering ones are also very slow, which implies that the DTW distance is significantly more expensive to calculate than the Hausdorff one. Regarding the SVM-based models, they are reasonably fast and the addition of GSP increases the execution time. Finally, in the proposed models, the Geolife dataset takes more time than the Cyprus one, because it has longer trajectories in terms of points and thus calculations that use the point sequences (i.e., grid conversion and distance calculation) are more expensive. On the other hand, TOP is slower for the Cyprus dataset, mainly due to the fact that the same number of trajectories refer to a smaller overall surface in square kilometers.

Since the DTW distance proved to be costly to calculate and in our last experiment (to be discussed below), it has its merits, we applied the following simplification to speed-up its processing. More specifically, the trajectories whose grid square representation exceeds a limit $L$ are sampled in a way that we keep $L$ points as equidistant as possible, including the first and last ones. For example, in a trajectory originally spanning 20 cells with $L$=6 sampling points, the trajectory points with indices [0, 4, 8, 11, 15, 19] are the ones chosen in the sample. In

[5]We have also tried to adapt and apply the sub-trajectory-oriented technique in [10] using the codebase provided at https://github.com/MillerWu2014/trajectory-cluster but the results were much inferior compared to any of the techniques presented in the evaluation results section. Given also that we have not investigated sub-trajectory-oriented techniques in depth, no results from [10] are presented.

**Table 2**
Results of the experiments.

| Dataset | Method | Testing Accuracy | Testing F1 |
|---|---|---|---|
| Geolife | TOP | 0.9226 | 0.7042 |
| Geolife | DODB | 0.8754 | 0.6152 |
| Geolife | PC1 | 0.9589 | 0.7817 |
| Geolife | PC2 | 0.8459 | 0.6109 |
| Geolife | PC3 | 0.7811 | 0.5777 |
| Geolife | SVM | 0.9748 | 0.8379 |
| Geolife | SVM+GSP | 0.9753 | 0.8539 |
| Geolife | SVM+GSP+PC1 (LR) | 0.9753 | 0.8539 |
| Geolife | SVM+GSP+PC1 (OR) | 0.9659 | 0.8369 |
| Cyprus | TOP | 0.8908 | 0.6866 |
| Cyprus | DODB | 0.9041 | 0.6468 |
| Cyprus | PC1 | 0.9331 | 0.7428 |
| Cyprus | PC2 | 0.8923 | 0.694 |
| Cyprus | PC3 | 0.9039 | 0.6677 |
| Cyprus | SVM | 0.9742 | 0.8626 |
| Cyprus | SVM+GSP | 0.9763 | 0.8741 |
| Cyprus | SVM+GSP+PC1 (LR) | 0.9763 | 0.8741 |
| Cyprus | SVM+GSP+PC1 (OR) | 0.9387 | 0.7753 |

**Table 3**
The execution times of the various experimental settings of the paper. These are the execution times of the implementation that can be found in the GitHub link in Section 1. The abbreviations are the ones of Table 2.

| Method | Exec. time: Geolife | Exec. time: Cyprus |
|---|---|---|
| TOP | 0.91sec | 37sec |
| DODB | 3hrs | 1.5hrs |
| PC1 | 175sec | 100sec |
| PC2 | 4hrs | 2hrs |
| PC3 | 4hrs | 3hrs |
| SVM | 110sec | 14sec |
| SVM+GSP | 439sec | 20sec |
| PC1+SVM+GSP(LR) | 590sec | 121sec |
| PC1+SVM+GSP(OR) | 152sec | 21.7sec |

our implementation, to configure $L$, the average length of the grid-converted trajectories is calculated, the ceiling function is applied to it and the result is incremented by 1. Any trajectory larger than that is sampled in the way mentioned above. All the accuracy results for DTW-based techniques presented follow this simplification.

In our final experiment, with a view to determining the proposed models' performance in known outliers, 22 manually created outliers by domain experts were added in the Cyprus dataset and the percentage of them accurately getting labeled as outliers is measured. The data matrix used for the SVM model building (without the GSP column) is used to fit a $k$-Means clustering model. This is performed in order to find meaningful clusters in the data and be capable of explaining them. With $k$=6, the silhouette score is maximized and interesting patterns are revealed. The six clusters that appear are the ones in Figure 1 and they correspond to the 5 administrative regions of Cyprus plus some trajectories among the regions (and especially the cities) of Nicosia, Limassol and

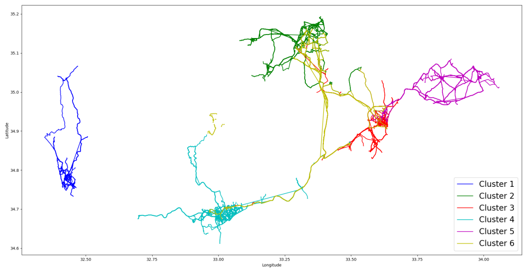

**Figure 1:** The trajectory clusters from the $K$-Means clustering process; the trajectories form clusters near the major urban centers of Cyprus and roughly correspond to the administrative regions.

**Table 4**
The performance of the proposed models regarding the detection of the manual outliers.

| # | Cluster | Trip | Distance | PC1 | PC2 | PC3 | SVM | SG | PC1+SG(LR) | PC1+SG(OR) |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Paphos | Kouklia-Paphos-Pegeia-Tsada | 55.6km | X | X |  | X | X | X | X |
| 2 | Paphos | Kathikas | 170m | X |  |  |  |  |  | X |
| 3 | Limassol | Ag.Ambrosios-Limassol-Palodia-Foinikaria | 55.4km | X | X | X | X | X | X | X |
| 4 | Limassol | Palodia | 170m | X | X | X |  | X | X | X |
| 5 | Larnaca | Mazotos-Larnaca-Kosi-Oroklini-Mazotos | 87.6km | X | X | X | X | X | X | X |
| 6 | Larnaca | Aradippou | 270m |  |  |  |  |  |  |  |
| 7 | Deryneia | Xylotymvou-Xylofagou-Avgorou-Deryneia-Ag.Napa | 116km | X | X | X | X | X | X | X |
| 8 | Deryneia | Paralimni | 600m |  |  |  |  |  |  |  |
| 9 | Nicosia | Peristerona-Malounta-Kaimakli-Dali-Tseri-Akaki | 44.3km | X | X | X | X |  |  | X |
| 10 | Nicosia | Encomi | 450m |  |  |  |  |  |  |  |
| 11 | - | Paphos-Nicosia | 152km |  | X | X | X | X | X | X |
| 12 | - | Paphos-Limassol | 68km | X | X | X | X | X | X | X |
| 13 | - | Paphos-Larnaca | 132km | X | X | X | X | X | X | X |
| 14 | - | Paphos-Deryneia | 178km |  | X | X | X | X | X | X |
| 15 | - | Limassol-Deryneia | 114km | X | X | X | X | X | X | X |
| 16 | - | Limassol-Paphos | 68km | X | X | X | X | X | X | X |
| 17 | - | Larnaca-Paphos | 132km | X | X |  | X | X | X | X |
| 18 | - | Deryneia-Limassol | 114km | X | X | X | X | X | X | X |
| 19 | - | Deryneia-Paphos | 179km |  | X | X | X | X | X | X |
| 20 | - | Deryneia-Nicosia | 89km | X | X | X | X | X | X | X |
| 21 | - | Nicosia-Deryneia | 86km | X | X | X | X | X | X | X |
| 22 | - | Nicosia-Paphos | 151km |  | X | X | X | X | X | X |
|  |  | Percentage of outliers detected |  | 68.2% | 81.8% | 72.7% | 77.2% | 77.2% | 77.2% | 86.36% |

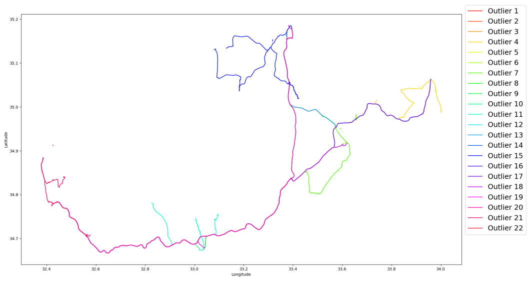

**Figure 2:** The manually added outliers: trajectories that connect regions that are usually not connected, very long trajectories in a certain cluster and very short trajectories in a certain cluster (that they may not be easily visible in the chart).

Larnaca. This conforms to the domain knowledge that most trajectories happen from warehouses in the urban centers to dropping locations inside or near the city, with some also connecting different warehouses in different cities.

Based on the aforementioned understanding of the dataset, outlying trajectories are injected manually. These trajectories are significantly different than the norm. For example, some of them start from an urban center and go to another urban center that does not usually interconnect with the former. Others have a generally very long distance and, finally, others may run between frequently connected cities but are too lengthy since they take too many detours through rural areas. The 22 trajectories that are used for this experiment are plotted in Figure 2.

The proposed solutions along with their variants are then evaluated on detecting the manual outliers using the same setting as in the experiment that yielded Table 2. The results are shown in Table 4. PC1 is the lowest-performing model, with a 68% detection rate, while SVM, SVM+GSP and PC1+SG(LR) have a 77% detection rate, PC3 has 77%, PC2 has 82% and PC1+SG(OR) has the highest detection rate, 86.4%. Interestingly, there is a single outlier case in the second row that only PC1 and PC1+SG(OR) managed to detect. All models face difficulties in detecting very short trajectories as outliers, while most of them correctly detect the very long ones as outliers. Finally, most of them perform almost perfectly on detecting infrequent trips between administrative region centers as outliers.

Overall, our remarks are summarized as follows. In the generic case, where there outliers are a small proportion of the whole dataset at the level of 5%, SVM along with GSP performs better and significantly improves upon the competitors. When tested using very few artificially generated and manually injected outliers, using an ensemble in which SVM with GSP is paired with Hausdorff distance-enabled path clustering is the dominant solution. This supports our main observation that, in general, the Hausdorff distance is preferable in path clustering. Finally, in absolute values, the achieved F1-scores are high and reach 0.8741, whereas [9] achieved up to 0.7042.

## 5. Conclusion and Further Research

In this paper, two methods are proposed, a path clustering method and a SVM+GSP method. The former performs

DBSCAN clustering on the trajectories by using their Hausdorff distance, DTW distance and DTW-Hilbert distance, while the latter fits a SVM classifier on a 2-class dataset by using as features the starting and destination coordinates, the distance of the trajectory and the deviation from the closest frequent subsequence. The last feature is calculated by applying the GSP algorithm to find the top frequent sub-trajectories and using the Hausdorff distance as well. The results show that both methods outperform the baseline and detect outliers efficiently, with SVM+GSP performing better than path clustering both in terms of accuracy/F1 and execution time. However, when the outputs of the 2 models are combined with a logistic regression layer, the performance of the classification does not improve. When an ensemble combining the two proposals using a logical OR is applied, the performance drops in the case of the datasets with 5% outliers but is the the highest one in terms of detecting the rare manually injected outliers. All our code has become publicly available.

The proposed methods have space and time limitations, being quadratic for both aspects. Thus, it would be of interest to study more scalable solutions with a view to rendering the solutions suitable for real-time processing. Finally, the application of the temporal dimension in the proposed method, along with derived features such as velocities, accelerations and so on, would be beneficial and we plan to explore it in the future.

## Acknowledgments

The research is funded by Research Promotion Foundation under Restart Research 2016-2020 Programme under the ENTERPRISES/0521/0138 agreement.

## References

[1] Y. Djenouri, A. Belhadi, J. C.-W. Lin, D. Djenouri, A. Cano, A survey on urban traffic anomalies detection algorithms, IEEE Access 7 (2019) 12192–12205.

[2] X. Kong, X. Song, F. Xia, H.-Y. Guo, J. Wang, A. M. Tolba, Lotad: long-term traffic anomaly detection based on crowdsourced bus trajectory data, World Wide Web 21 (2017) 825–847.

[3] Y. Djenouri, A. Belhadi, J. C.-W. Lin, A. Cano, Adapted k-nearest neighbors for detecting anomalies on spatio–temporal traffic flow, IEEE Access 7 (2019) 10015–10027. doi:10.1109/ACCESS.2019.2891933.

[4] J. Zhang, Smarter outlier detection and deeper understanding of large-scale taxi trip records: A case study of nyc, in: Proceedings of the ACM SIGKDD International Workshop on Urban Computing, UrbComp '12, Association for Computing Machinery, New York, NY, USA, 2012, p. 157–162. URL: https://doi.org/10.1145/2346496.2346521. doi:10.1145/2346496.2346521.

[5] X. Zhou, Y. Ding, F. Peng, Q. Luo, L. Ni, Detecting unmetered taxi rides from trajectory data, 2017, pp. 530–535. doi:10.1109/BigData.2017.8257968.

[6] I. Varlamis, K. Tserpes, M. Etemad, A. S. Júnior, S. Matwin, A network abstraction of multi-vessel trajectory data for detecting anomalies., in: EDBT/ICDT Workshops, volume 2019, 2019.

[7] J.-G. Lee, J. Han, X. Li, Trajectory outlier detection: A partition-and-detect framework, 2008, pp. 140–149. doi:10.1109/ICDE.2008.4497422.

[8] Z. Li, J. Han, M. Ji, L. Tang, Y. Yu, B. Ding, J.-G. Lee, R. Kays, Movemine: Mining moving object data for discovery of animal movement patterns, ACM TIST 2 (2011) 37. doi:10.1145/1989734.1989741.

[9] L. Cao, Y. Yan, S. Madden, E. Rundensteiner, M. Gopalsamy, Efficient discovery of sequence outlier patterns, Proc. of VLDB 12 (2019) 920–932. doi:10.14778/3324301.3324308.

[10] J.-G. Lee, J. Han, K.-Y. Whang, Trajectory clustering: a partition-and-group framework, in: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, 2007, pp. 593–604.

[11] E. O. Eldawy, H. M. Mokhtar, Clustering-based trajectory outlier detection, International Journal of Advanced Computer Science and Applications 11 (2020). URL: http://dx.doi.org/10.14569/IJACSA.2020.0110520. doi:10.14569/IJACSA.2020.0110520.

[12] Z. Lv, J. Xu, P. Zhao, G. Liu, L. Zhao, X. Zhou, Outlier trajectory detection: A trajectory analytics based approach, in: International Conference on Database Systems for Advanced Applications, Springer, 2017, pp. 231–246.

[13] I. San Román, I. Martín de Diego, C. Conde, E. Cabello, Outlier trajectory detection through a context-aware distance, Pattern Analysis and Applications 22 (2019) 831–839.

[14] D. Kumar, M. Palaniswami, S. Rajasegarar, C. Leckie, J. Bezdek, T. Havens, Clusivat: A mixed visual/numerical clustering algorithm for big data, 2013, pp. 112–117. doi:10.1109/BigData.2013.6691561.

[15] D. Kumar, J. Bezdek, S. Rajasegarar, C. Leckie, M. Palaniswami, A visual-numeric approach to clustering and anomaly detection for trajectory data, The Visual Computer 33 (2017). doi:10.1007/s00371-015-1192-x.

[16] J. Zhu, W. Jiang, A. Liu, G. Liu, L. Zhao, Time-dependent popular routes based trajectory outlier detection, in: International Conference on Web Information Systems Engineering, Springer, 2015,

pp. 16–30.

[17] J. Zhu, W. Jiang, A. Liu, G. Liu, L. Zhao, Effective and efficient trajectory outlier detection based on time-dependent popular route, World Wide Web 20 (2017) 111–134.

[18] P.-R. Lei, A framework for anomaly detection in maritime trajectory behavior, Knowledge and Information Systems 47 (2016) 189–214.

[19] J. Lan, C. Long, R. C.-W. Wong, Y. Chen, Y. Fu, D. Guo, S. Liu, Y. Ge, Y. Zhou, J. Li, A new framework for traffic anomaly detection, in: Proceedings of the 2014 SIAM International Conference on DATA MINING, SIAM, 2014, pp. 875–883.

[20] W. Liu, Y. Zheng, S. Chawla, J. Yuan, X. Xing, Discovering spatio-temporal causal interactions in traffic data streams, 2011, pp. 1010–1018. doi:10.1145/2020408.2020571.

[21] B. Lei, D. Mingchao, A distance-based trajectory outlier detection method on maritime traffic data, in: 2018 4th International Conference on Control, Automation and Robotics (ICCAR), 2018, pp. 340–343. doi:10.1109/ICCAR.2018.8384697.

[22] Z. Liu, D. Pi, J. Jiang, Density-based trajectory outlier detection algorithm, Journal of Systems Engineering and Electronics 24 (2013) 335–340. doi:10.1109/JSEE.2013.00042.

[23] F. Luan, Y. Zhang, K. Cao, Q. Li, Based local density trajectory outlier detection with partition-and-detect framework, in: 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2017, pp. 1708–1714. doi:10.1109/FSKD.2017.8393023.

[24] Y. Yu, L. Cao, E. Rundensteiner, Q. Wang, Outlier detection over massive-scale trajectory streams, ACM Transactions on Database Systems 42 (2017) 1–33. doi:10.1145/3013527.

[25] Y. Djenouri, A. Zimek, M. Chiarandini, Outlier detection in urban traffic flow distributions, 2018, pp. 935–940. doi:10.1109/ICDM.2018.00114.

[26] J. Mao, T. Wang, C. Jin, A. Zhou, Feature grouping-based outlier detection upon streaming trajectories, IEEE Transactions on Knowledge and Data Engineering 29 (2017) 2696–2709. doi:10.1109/TKDE.2017.2744619.

[27] F. Maiorano, A. Petrosino, Granular trajectory based anomaly detection for surveillance, in: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, 2016, pp. 2066–2072.

[28] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, S. Li, ibat: Detecting anomalous taxi trajectories from gps traces, 2011, pp. 99–108. doi:10.1145/2030112.2030127.

[29] H. Xiaolin, C. Reynold, M. Chenhao, T. Grubenmann, Deeptea: Effective and efficient online time-dependent trajectory outlier detection (2022). doi:10.14778/3523210.3523225.

[30] Z. Li, M. Ji, J.-G. Lee, L.-A. Tang, Y. Yu, J. Han, R. Kays, Movemine: Mining moving object databases, in: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, 2010, pp. 1203–1206.

[31] F. Wu, T. K. H. Lei, Z. Li, J. Han, Movemine 2.0: Mining object relationships from movement data, Proceedings of the VLDB Endowment 7 (2014) 1613–1616.

[32] O. S. Siordia, I. M. de Diego, C. Conde, E. Cabello, Section-wise similarities for clustering and outlier detection of subjective sequential data, in: M. Pelillo, E. R. Hancock (Eds.), Similarity-Based Pattern Recognition, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 61–76.

[33] R. Srikant, R. Agrawal, Mining sequential patterns: Generalizations and performance improvements, in: International conference on extending database technology, Springer, 1996, pp. 1–17.

[34] D. Sankoff, J. B. Kruskal (Eds.), Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, Addison-Wesley, 1983.

[35] Y. Zheng, Q. Li, Y. Chen, X. Xie, W.-Y. Ma, Understanding mobility based on gps data, in: Proceedings of the 10th international conference on Ubiquitous computing, 2008, pp. 312–321.

[36] Y. Zheng, X. Xie, W.-Y. Ma, et al., Geolife: A collaborative social networking service among user, location and trajectory., IEEE Data Eng. Bull. 33 (2010) 32–39.

[37] Y. Zheng, L. Zhang, X. Xie, W.-Y. Ma, Mining interesting locations and travel sequences from gps trajectories, in: Proceedings of the 18th International Conference on World Wide Web, WWW '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 791–800. URL: https://doi.org/10.1145/1526709.1526816. doi:10.1145/1526709.1526816.

[38] H. Wu, W. Sun, B. Zheng, A fast trajectory outlier detection approach via driving behavior modeling, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 837–846. doi:10.1145/3132847.3132933.

## A. Online Resources

https://github.com/amoavinis/trajectory-outliers