Spectral Decomposition and Sparse Models for Big Data and Robust Data Analytics

Georgios T. Skaros
Department of Informatics,
University of Piraeus,
Piraeus, Greece
September 2021

Thesis co-supervisors:

Prof. Yannis Theodoridis, Dr. Harris Georgiou

Contents

Abstract

The volume of trajectory data has become tremendously large in recent years. How to efficiently store and compute such data becomes a challenging task. In this thesis, we propose a trajectory spatial compression using Principal Component Analysis. With the usage of this technique, we could compress the data set by reducing it into principal components i.e., vectors which contain condensed information of the movement of vehicles. By dividing each route into segments and applying PCA to them, we show that there is a better chance to find principal components that represent smaller and simpler movements. This allows us to compress trajectory data as well as enable us to cluster the trajectory data using a well-known density-based spatial clustering algorithm, DBSCAN, without achieving a point focus clustering. The proposed technique significantly reduces the size of the original data while retaining most of information. In addition to trajectory-based clustering, a significant reduction in computation times is achieved as well.

Περίληψη

Ο όγκος των δεδομένων τροχιών έχει γίνει εξαιρετικά μεγάλος τα τελευταία χρόνια. Η αποτελεσματική αποθήκευση και η επεξεργασία τέτοιων δεδομένων γίνεται ολοένα και πιο δύσκολο έργο. Σε αυτή τη πτυχιακή εργασία, προτείνεται μια μορφή συμπίεσης τροχιών χρησιμοποιώντας την ανάλυση κύριων συνιστωσών(PCA), μια τεχνική μείωσης διαστάσεων. Με τη χρήση αυτής της τεχνικής, θα μπορούσαμε να συμπιέσουμε το σύνολο δεδομένων μειώνοντάς το σε κύριες συνιστώσες, δηλαδή διανύσματα που περιέχουν συμπυκνωμένη πληροφορία για την κίνηση των οχημάτων. Χωρίζοντας κάθε διαδρομή σε τμήματα και εφαρμόζοντας PCA σε αυτά, δείχνουμε ότι υπάρχει καλύτερη πιθανότητα να βρεθούν κύριες συνιστώσες που αντιπροσωπεύουν μικρότερες και απλούστερες κινήσεις των δεδομένων. Αυτό μας επιτρέπει να συμπιέζουμε τα τροχιακά δεδομένα καθώς επίσης και να μπορούμε να τα συσταδοποιήσουμε χρησιμοποιώντας έναν γνωστό αλγόριθμο χωρικής συσταδοποίησης με βάση την πυκνότητα, το DBSCAN, χωρίς να επιτύχουμε μια ομαδοποίηση σημειακής εστίασης. Η προτεινόμενη τεχνική μειώνει σημαντικά το μέγεθος των αρχικών δεδομένων διατηρώντας παράλληλα το μεγαλύτερο μέρος της πληροφορίας. Εκτός από την συσταδοποίηση με βάση την τροχιά, επιτυγχάνεται επίσης σημαντική μείωση στους χρόνους υπολογισμού.

1 Introduction

Technology is advancing steadily over the years, which means that new discoveries are emerging day by day. Drastic technological advancements often tend to miss out to find themselves a mass market where they can be sold. The population of the world stands at more than 7.8 billion people [1] and more than 4.66 billion of these people are connected to the internet. More than 3.8 billion people are using (48%) smart phones [2]. The number of devices and connections will keep increasing year on year basis and thus this had led to an explosion of data [3]. This enormous amount of data, that is produced on a continuous basis is termed as big data [4]. Big data analytics has fairly gained significant amount of importance as it enables organizations to be ahead of their competitors [5] as well as enabling other disciplines to flourish, such as biology or physics, thus helping the ordinary citizen. Devices such as smart phones, GPS devices, smart watches etc. have the ability to generate, store and transmit trajectory data. A trajectory is defined as a stream of 3-tuple records consisting of the position (latitude, longitude), along with the temporal information (when the moving object was at the specified location).

There are some major problems in location-based applications that use trajectory data. First, storing the sheer volume of it can quickly overwhelm available data storage space. For example, if data are collected without compression at 10 s intervals, 1 Gb of storage capacity is required to store just over 4,000 objects for a single day [6]. For this reason, efficiently storing and querying GPS trajectories is an important area of active research [6] [7] [8] [9]. Furthermore, as the trajectory data size gets larger, it becomes more difficult to detect useful patterns from the data. This happens due to the nature of the classification and clustering algorithms. They calculate each distance between the points of the data set in each repetition of the algorithm which significantly increases the computation times. Reducing the size of the data has the potential to accelerate the mining of patterns [10].

The present work focuses on compression with a dimension reduction technique aiming at the greatest possible compression while maintaining the information and shape of the original trajectory and the effects that this type of compression has on clustering algorithms. The data set that was used was from a taxi agency located in Porto, Portugal and was in the form of trips (trajectories containing tuples of longitude and latitude). For the purpose of this research 30000 trips of the original data were used.

After preprocessing and cleansing, the data was cut into smaller segments in order to be able to compress it using PCA. The segment length was determined by result using as metrics the cumulative variance of the principal components used, the root mean square error, the compressed size and compression ratio. The number of principal components used was determined using the same metrices. The clustering of the data was performed in the first four principal components that were selected during the dimensionality reduction. Various values of esp (maximum intra-cluster distance between points) have been tested in order to find results with meaningful clusters.

Through this research was found that with the usage of dimensionality reduction techniques such as PCA we can have high compression ratios while maintaining much of the information of the original data. Regarding the clustering of the compressed data, different clusters were observed in relation to the original data as well as the computational time was reduced.

The remainder of this thesis is organized as follows; a literature review will be presented below. In Chapter 2 we will review the methods used regarding segmentation, compression and clustering. Next, Chapter 3 reports the experiments and results. Finally, Chapter 4 concludes this thesis.

All the techniques that are presented in the rest of this thesis are implemented using Python 3.8 and the libraries: pandas, geopandas, matplotlib, scikit-learn, psycopg2 [11] [12] [13] [14] [15] [16]. The data was stored in a PostgreSQL database with the usage of the extension postgis in order to be able to handle the GIS data [17] [18].

1.1 Literature review

Regarding the compression of mobility data, Douglas-Peucker line simplification algorithm was proposed in Douglas and Peucker. The main functionality of this algorithm is the reduction of the number of points in a curve that is approximated by a significantly smaller series of points. Given a curve consisting of a set of line segments, the goal of the algorithm is to discover an analogous curve consisting of a smaller number of points, selected among the original ones. Here, "similarity" is defined upon the maximum distance between the original and the simplified curve. Practically, this method computes the perpendicular distance of each internal point from the line connecting the first and the last point of the original curve and discovers the point with the maximum perpendicular distance. Then, it recursively performs the same procedure with the curve now considered to be the polyline produced due to the aforementioned "breaking point", and so on. The ending condition of this algorithm is met when the distance of all the remaining points from the currently examined line is less than a given threshold or there are no points left to examine. In the first case the algorithm ends and returns this line segment as new compressed polyline. It is obvious that Douglas-Peucker does not take the time dimension into consideration since it is a (geometrical) line simplification algorithm. Its time complexity is $O(N^2)$ with N being the number of data points [19]. Different implementations have reduced the complexity of the method to NlogN [20].

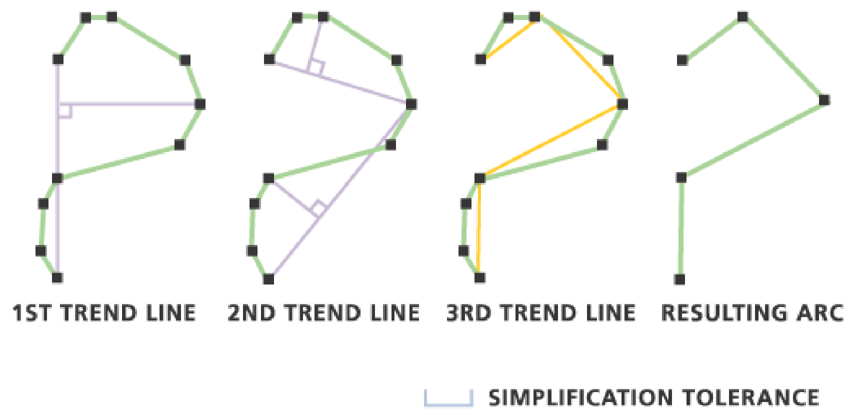

Figure 1 **Example of Douglas-Peucker line simplification algorithm**

Opening window compression algorithms are popular even though they are computationally expensive having a complexity of $O(N^2)$. The popularity come from their ability to deal with noise and because they are online algorithms. Top-Down Time Ratio (TD-TR) and Open Window Time Ratio (OPW-TR) algorithms for compressing spatiotemporal data were proposed in Meratnia and de By [6]. This approach differentiates form the original top-down algorithm by integrating the concepts of speed difference threshold and the time-ratio distance. This is an important improvement not only because we are using a more accurate distance measure but also because the temporal factor is now included. The continuous nature of moving objects necessitates the inclusion of temporal as well as spatial properties of moving objects. The algorithm sets the anchor point, and then gradually 'opens the window'. In each step, two halting conditions are verified, one on the synchronous distance measure (using the time interval ratio), the other being a difference in speed values between previous and next trajectory segment. These speeds are not measured speeds, as we do not assume these to be available; rather, they are speed values derived from timestamps and positions.

Potamias et al. [21] proposed two algorithms, called Thresholds and STTrace, respectively, for online trajectory data compression. The key intuition is that a point should be taken into the sample as long as it reveals a relative change in the course of a trajectory. Therefore, a decision to accept or reject a point is taken according to user-defined rules specifying the allowed tolerance ("threshold") in changes in speed and orientation. A new point must be appended to the sample when that threshold is exceeded for an incoming location. Under this scheme and in contrast to uniform sampling, the total amount of items in the sample keeps increasing without eliminating any point already stored. The time complexity of threshold-guided sampling is O(1) per incoming tuple. The intuition behind STTrace is to use an insertion scheme based on the recent movement features (as in threshold-guided algorithms), but at the same time also allowing deletions from the sample to make room for the newly inserted points without exceeding allocated memory (as occurs in uniform sampling). However, there is an important differentiation: the sampled point chosen for deletion is not selected randomly over the current sample contents, but according to its significance in trajectory preservation. For every new insertion, the most appropriate candidate point must be searched for deletion over the entire sample with O(M) worst-case cost, where M is the current sample size. M should be expected very small, and the sampled points may be maintained in an appropriate data structure (e.g., a binary balanced tree) with logarithmic cost for operations (search, insert, delete), normally this is an affordable trade-off.

A trajectory simplification method aiming at Location-based social networking (LBSN) applications was proposed in Chen et al. [22]. The trajectory simplification algorithm is comprised of four steps: segmentation, point distribution, point weighting and point selection. At first, we partition a GPS trajectory into walking and non-walking segments. Then, the headcounts of points will be assigned to each segment in terms of the product of the average heading change and the distance of each segment. In each segment, the points with a relatively large production of heading change and neighbor distance will be assigned a big weight. Finally, the top points with large weights are remained to represent the simplified trajectory.

A combined compression and map-matching approach was proposed in Kellaris et al. [23]. The main idea was to build the compressed trajectory by replacing some paths of a given map-matched trajectory with shorter ones. This could be done by executing a shortest path algorithm on appropriately selected points of the trajectory without considering the weights of the edges. The weights are ignored in order to get the result with the minimum number of nodes and, hence, achieve a high compression.

The literature on mobility data mining, viewed in Giannotti and Pedreschi [24] and Pelekis and Theodoridis [25], can be classified in various fields depending on the mining models used to discover collection of patterns. As such, there have been proposed approaches that identify various types of clusters of moving objects. Many of these techniques are based on existing density clustering techniques. These types of clustering consider the density of the dataset to form clusters. The most prevalent is DBSCAN (more detail in 2.4 Clustering algorithm with its successor and OPTICS which can detect clusters with variant densities. According to OPTICS, objects are linearly ordered based on their spatial closeness. Additionally, for each point, a distance measure (called reachability distance) that represents the required density of this point to be accepted to belong to the same cluster with its neighbor is stored. The reachability distance of point P_1 with respect to point P_2 is actually the smallest distance, such that P_1 is directly density-reachable from P_2, provided that P_2 is a core point, otherwise it is undefined. Gaffney and Smyth [26] and Cadez et al. [27] proposed probabilistic algorithms that cluster small trajectories. The unsupervised learning technique, Expectation-Maximization (EM), is carried out using maximum likelihood principles. Nanni and Pedreschi [28] extended density-based clustering techniques and introduced the T-OPTICS algorithm an extension to OPTICS algorithm by defining a similarity function and using it to group trajectories. In the so-called Trajectory-OPTICS (T-OPTICS), trajectories are clustered by means of a variant of the DISSIM distance function. This variant simply normalizes the integral of the DISSIM distance function with the duration of the common lifespan of the trajectories. As the algorithm is driven by the OPTICS method, the outcome of T-OPTICS is also a reachability plot. Performing global trajectory clustering (i.e., the grouping considers the whole trajectories) may sometimes lead to a misleading outcome, as in the case of partial membership of objects to several clusters.

Pelekis et al. [29] [30] proposed CenTR-I-FCM. Transforming trajectories to vectors for clustering has been followed by the CenTR-I-FCM algorithm in connection with a soft clustering algorithm based on fuzzy logic, namely Fuzzy-C-Means (FCM). FCM is similar to K-means, but it further considers uncertainty by allowing each data element to belong to different clusters by a certain degree of membership. The approach initially adopts a symbolic representation and model trajectories as sequences of regions (i.e., wherefrom a moving object passes) accompanied with intuitionistic fuzzy values, i.e., elements of a so-called intuitionistic fuzzy set. Intuitionistic Fuzzy Sets (IFS) are generalized fuzzy sets that can be useful in coping with the hesitancy originating from imprecise information. IFS elements are characterized by two values representing, respectively, their belongingness and non-belongingness to this set. In the case of MOD where this set is the region that a trajectory possibly crosses, the above values represent the probabilities of presence and non-presence in the area. In order to exploit this information, the approach defines a novel variant of the ERP distance metric, especially designed to operate on such intuitionistic fuzzy vectors, having as goal to incorporate it in some variant of the FCM algorithm that will effectively cluster trajectories under uncertainty.

Contrary to the above algorithms, which consider entire trajectories, the TRACLUS framework was the first sub-trajectory clustering algorithm (Lee et al. [31]). The main idea of the algorithm is that trajectories are partitioned to directed segments (i.e., sub-trajectories) and then grouped irrespective of whether the trajectories as a whole are split into different clusters or not. After the grouping and for each cluster, a virtual representative trajectory is synthesized in order to describe the overall movement of the grouped set of directed segments. TRACLUS also proposed an algorithm for computing a virtual representative trajectory for each cluster. More specifically, the TRACLUS workflow consists of three steps. In the first step (trajectory partitioning), the algorithm aims at discovering characteristic points of each trajectory where its behavior changes significantly. In the second step (sub-trajectory clustering), the algorithm groups sub-trajectories by using a variant of the DBSCAN algorithm, applicable to the directed segments. at the third step (representative trajectory generation), for each cluster formed the algorithm calculates its representative trajectory.

In case where moving objects move under the restrictions of a transportation network, FlowScan and NEAT are road network-aware approaches for clustering spatial trajectories of objects moving in a road network and were proposed by Li et al. [32] and Han et al. [33], respectively. FlowScan is a technique that discover hot routes patterns, operating upon trajectories moving on a road network. A hot route can be considered as a general traffic flow pattern of moving objects, which is a sequence of nearby, though not necessarily adjacent, edges that share a high amount of traffic. The hot route discovery algorithm proceeds by first defining all hot route starts. An edge e is considered as a hot route start, if there are more than (a user-defined threshold) MinTraffic objects that start from r or when MinTraffic or more objects converge at e from other edges, however by not accounting traffic from edges that belong to other hot routes. For each hot route start, the related hot routes are extracted following a traffic-density expansion process. The expansion of a hot route is performed only at the last edge (initially, the last is equal to the start). The last edge is grown by attaching the route traffic density-reachable edges. An edge s is route traffic density-reachable from an edge e, if there is a chain of directly density-reachable edges in between, which also share common traffic for a sliding window of consecutive edges in this chain. This technique uses the number of hops to measure closeness in the road network.

NEAT is another road network-aware approach. It considers the physical constraints of a road network, the network proximity, and the flows between road segments during the clustering procedure, which is accomplished in three phases. In the first phase, called base cluster formation, the original trajectories are partitioned into a set of trajectory fragments, called t-fragments. A t-fragment is considered as a sub-trajectory whose sequential points lie on the same road segment. Then, base clusters are formed each one containing the equivalent t-fragments that belong to the same road segment. In the flow cluster formation phase that follows, base clusters are combined into flow clusters (i.e., sequences of base clusters that compose a route in the network) with respect to the merging selectivity. Merging selectivity is a weighted function that it takes into account flow, density and road speed factors. Finally, at the flow cluster refinement phase, a variant of DBSCAN clustering algorithm is employed to compress flow clusters in order to optimize the results. The density-connectivity process is driven by the well-known Hausdorff distance function modified to consider the network distance. In detail, the distance between two flow clusters is calculated based on the corresponding routes. If the ending locations of two flow clusters do not exceed a predefined distance in terms of network proximity, the clusters are merged into one larger cluster. The final clustering fulfills high density and continuity criteria, thus depicting frequent traffic flows extracted from mobility data. In the same field, Sacharidis et al. [34] proposed an online approach to discover and maintain hot motions paths while Chen et al. [35] tackled the problem of discovering the most popular route between two locations based on the historical behavior of travelers.

2 Methods

This section presents all the methods during our research. It will proceed as follows; firstly, in Section 2.1 is the problem statement of this thesis. Section 2.2 focuses on the segmentation prosses that was used. Section 2.3 presents the dimensionality reduction algorithm PCA with examples. Section 2.4 discuss the basic concepts of grouping as well as the algorithm DBSCAN that was used in our research.

2.1 Problem statement

The purpose of this thesis is to address the problems of storage and consequently processing times of trajectory data. It will focus on the reduction of the data size with a dimensionality reduction technique (PCA) while preserving the shape of the trajectory. In addition, it will address the effect of the dimensionality reduction in the clustering of the trajectories.

2.2 Segmentation of trips

Trajectory data have traditionally been preprocessed and separated to be represented as a single line consisted of Points (latitude, longitude, timestamp) known as trip. Each trip consists of a starting point and a destination point as well as intermediate points with a constant sampling rate. This aims to limit a possibly larger trajectory with noise to a realistic trip that we can make better use of in our analysis. This applies that all the estimations, distances etc. are calculated with respect to the entire trajectory. In this thesis, we will focus on smaller segments of each trip.

Assume that a random trajectory has a length of N points. Dividing it by a value λ will produce κ segments of length λ and a segment with the remainder of the trajectory. This segment will have a length from 0 (in case the length of the trajectory is a multiple of λ) to $\lambda - 1$. Therefore, it arises that the length of the last segment will be in average $\lambda/2$. Two matrices are created one for latitude and one for longitude. Each one of them is filled with the corresponding data, adding each segment as a new column to the matrix thus creating two $\lambda \times (\kappa + 1)$ matrices. PCA cannot deal with missing values. In our implementation, we decided to act as follows:

- If the instances that have to be filled are less than $\lambda/2$ the rest of the last segment is filled with the last instance of latitude and longitude respectively.
- If the values to be filled are more than $\lambda/2$ we delete the remainder of the trip in order to make a multiple of λ.

To be able to calculate the results and reconstruct the original data set, the first instance of each trip will be stored as well as its length. In addition, it is necessary that all the trajectories must have the same sampling rate. The value of λ depends on the sampling rate. Bigger intervals between samples will correspond to a smaller λ than smaller intervals. There are a lot of algorithms and processes that will achieve a universal sampling rate [36], but this topic will not be covered in this thesis.

The algorithm that was described above, is applied to each trip in the data. Each of the segments that are created are then appended as columns to a new matrix. This will result in two horizontal tables, one for latitude and one for longitude, with many smaller and simpler trajectories. This gives us the ability to focus our research at the processing and encoding of these segments as well as mining patterns that are hidden within them.

2.3 PCA

The central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables. We will better explain the operation of the algorithm with an example below [37].

Suppose that x is a vector of p random variables, and that the variances of the p random variables and the structure of the covariances or correlations between the p variables are of interest. Unless p is small, or the structure is very simple, it will often not be very helpful to simply look at the p variances and all of the

$$\frac{1}{2}p(p-1)$$

correlations or co-variances. An alternative approach is to look for a few ($\ll$ p) derived variables that preserve most of the information given by these variances and correlations or co-variances.

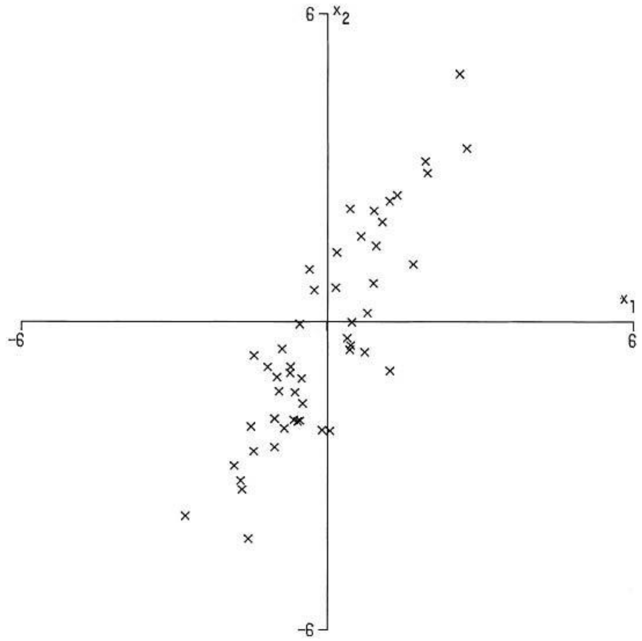

Figure 2 Plot of 50 observations on two variables x_1, x_2 .

Although PCA does not ignore co-variances and correlations, it concentrates on variances. The first step is to look for a linear function a'_1x of the elements of x having maximum variance, where α_1 is a vector of p constants $a_{11}, a_{12}, \ldots, a_{1p}x$ and ´ denotes transpose, so that

$$a'_1x = a_{11} + a_{12} + \cdots + a_{1p}x_p = \sum_{j=1}^{p} a_{1j}x_j.$$

Next, we look for a linear function a'_2x, uncorrelated with a'_1x having maximum variance, and so on, so that at the kth stage a linear function a'_kx is found that has maximum variance subject to being uncorrelated with $a'_1x, a'_2x, \ldots, a'_{k-1}x$. The kth derived variable, $a'_kx$is the kth PC. Up to p PCs could be found, but it is hoped, in general, that most of the variation in x will be accounted for by m PCs, where $m \ll p$. The reduction of complexity is achieved by transforming the original variables to PCs, but it might be useful here to consider first the unrealistic, but simple, case where p = 2. The advantage of p = 2 is, of course, that the data can be plotted exactly in two dimensions.

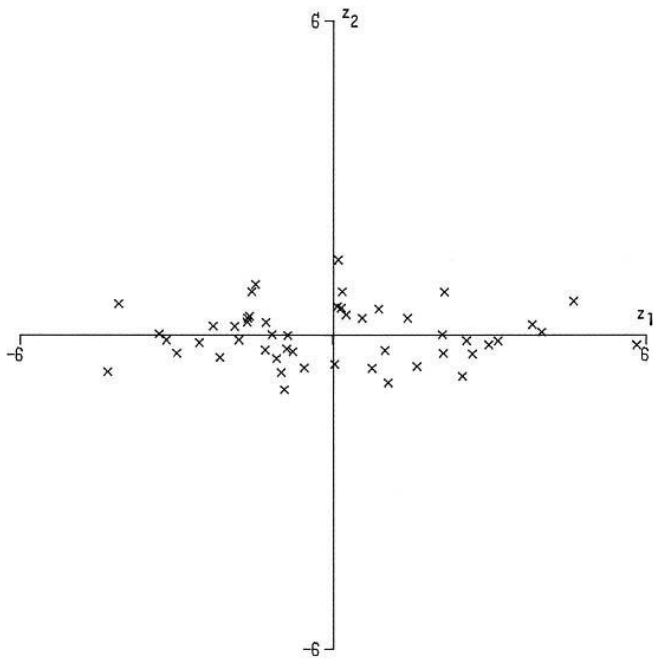

Figure 3 Plot of the 50 observations from Figure 1. with respect to their PCs z_1, z_2.

Figure 2 gives a plot of 50 observations on two highly correlated variables x_1, x_2. There is considerable variation in both variables, though rather more in the direction of x_2 than x_1. If we transform to PCs z_1, z_2, we obtain the plot given in Figure 3 Plot of the 50 observations from Figure 1. with respect to their PCs z_1, z_2..

It is clear that there is greater variation in the direction of z1 than in either of the original variables, but very little variation in the direction of z_2. More generally, if a set of p (> 2) variables has substantial correlations among them, then the first few PCs will account for most of the variation in the original variables. Conversely, the last few

PCs identify directions in which there is very little variation; that is, they identify near-constant linear relationships among the original variables.

Having defined PCs, we need to know how to find them. Consider, for the moment, the case where the vector of random variables x has a known co-variance matrix Σ. This is the matrix whose (i, j)th element is the (known) co-variance between the ith and jth elements of x when i = j, and the variance of the jth element of x when i = j. The more realistic case, where Σ is unknown, follows by replacing Σ by a sample co-variance matrix S. It turns out that for $k = 1,2, \ldots, p$, the kth PC is given by $z_k = a'_k x$ where α_k is an eigenvector of Σ corresponding to its kth largest eigenvalue λ_k. Furthermore, if α_k is chosen to have unit length ($a'_k a_k = 1$), then $\text{var}(z_k) = \lambda_k$ where $\text{var}(z_k)$ denotes the variance of z_k.

After additional calculations it concludes that the full principal components decomposition of X can therefore be given as

$$T = XW$$

where W is a p-by-p matrix of weights whose columns are the eigenvectors of $X^T X$. The transpose of W is sometimes called the whitening or sphering transformation. Columns of W multiplied by the square root of corresponding eigenvalues, that is, eigenvectors scaled up by the variances, are called loadings in PCA.

The transformation T = XW maps a data vector $x_{(i)}$ from an original space of p variables to a new space of p variables which are uncorrelated over the dataset. However, not all the principal components need to be kept. Keeping only the first L principal components, produced by using only the first L eigenvectors, gives the truncated transformation

$$T_L = XW$$

where the matrix T_L now has n rows but only L columns. In other words, PCA learns a linear transformation $t = W^T x, x \in R^p, t \in R^L$, where the columns of $p \times L$ matrix W form an orthogonal basis for the L features (the components of representation t) that are decorrelated [38].

After explaining the basic concept of PCA we can understand how this algorithm will be applied in the results of the segmentation process we previously discussed. The reduction of each of the segments from a variable length of λ to length L PCs, where $L \ll \lambda$ will significantly reduce the size of the dataset. This will affect the storage of the dataset as well as its processing speed during the clustering process.

2.4 Clustering algorithm

Clustering is one of the major data mining methods for knowledge discovery in large databases. It is the process of grouping large data sets according to their similarity. Cluster analysis is a major tool in many areas of engineering and scientific applications including data segmentation, discretization of continuous attributes, data reduction, outlier detection, noise filtering, pattern recognition and image processing. In the field of Knowledge Discovery in Databases (KDD), cluster analysis is known as unsupervised learning process, since there is no A priori (prior) knowledge about the data set.

The Density-based notion is a common approach for clustering. Density-based clustering algorithms are based on the idea that objects which form a dense region should be grouped together into one cluster. They use a fixed threshold value to determine dense regions. They search for regions of high density in a feature space that are separated by regions of lower density.

DBSCAN is designed to discover arbitrary-shaped clusters in any database D and at the same time can distinguish noise points. More specifically, DBSCAN accepts a radius value Eps(ε) based on a user defined distance measure and a value MinPts for the number of minimal points that should occur within Eps radius. Some concepts and terms to explain the DBSCAN algorithm can be defined as follows [39] [40].

- Definition 1 (Neighborhood). It is determined by a distance function (e.g., Manhattan Distance, Euclidean Distance) for two points p and q, denoted by $\mathrm{dist}(p, q)$.
- Definition 2 (Eps-neighborhood). The Eps-neighborhood of a point p is defined by $\{q \in D | \mathrm{dist}(p, q) \leq \mathrm{Eps}\}$.
- Definition 3 (Core object). A core object refers to such point that its neighborhood of a given radius (Eps) has to contain at least a minimum number (MinPts) of other points (Figure 4c).
- Definition 4 (Directly density-reachable). An object p is directly density-reachable from the object q if p is within the Eps-neighborhood of q, and q is a core object.
- Definition 5 (Density-reachable). An object p is density-reachable from the object q with respect to Eps and MinPts if there is a chain of objects $p_1, \dots, p_n, p_1 = q$ and $p_n = q$ such that p_{i+1} is directly density-reachable from pi with respect to Eps and MinPts, for $1 \leq i \leq n, p_i \in D$ (Figure 4a).
- Definition 6 (Density-connected). An object p is density-connected to object q with respect to Eps and MinPts if there is an object $o \in D$ such that both p and q are density-reachable from o with respect to Eps and MinPts (Figure 4b).
- Definition 7 (Density-based cluster). A cluster C is a non-empty subset of D satisfying the following "maximality" and "connectivity" requirements:
    1. $\forall p, q$: if$q \in C \wedge p$ is density-reachable from q with respect to Eps and MinPts, then $p \in C$.
    2. $\forall p, q \in C$: p is density-connected to q with respect to Eps and MinPts.
- Definition 8 (Border object). An object p is a border object if it is not a core object but density-reachable from another core object.

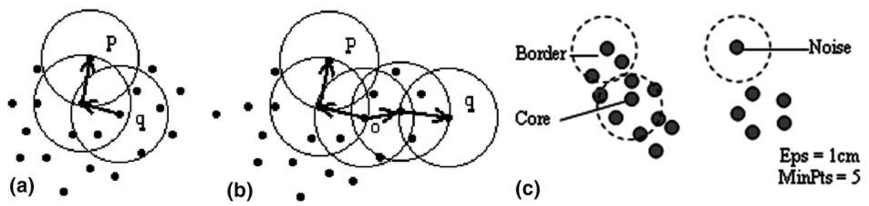

Figure 4 Basic concepts and terms: (a) p density-reachable from q, (b) p and q density-connected to each other by o and (c) border object, core object and noise.

The algorithm starts with the first point p in database D and retrieves all neighbors of point p within Eps distance. If the total number of these neighbors is greater than MinPts, if p is a core object, a new cluster is created. The point p and its neighbors are assigned into this new cluster. Then, it iteratively collects the neighbors within Eps distance from the core points. The process is repeated until all of the points have been processed.

3 Experiments & Results

This section presents the application and results of the techniques that were previously described. It will proceed as follows; Section 3.1 describes the dataset that was used in our experiments. Then, some statistics about it are presented. Section 3.2 presents the application of PCA in the segmented dataset. The results of the dimensionality reduction will be evaluated with some metrics. Section 3.3 discuss the results of clustering. The results are presented with plots of the clusters that were created.

3.1 Description of dataset

The stream event data of a taxi company operating in the city of Porto, Portugal, was used as the case study. The data was originally purposed for a taxi demand predictor [41] so it was stored in an unmanageable form for us to process so some more preprocessing was required. More specifically, the data were continuously acquired using the telematics installed in each one of the 441 running vehicles of the company fleet. This taxi central usually runs in one out of three 8-h shifts, i.e., from midnight to 8 AM, from 8 AM to 4 PM, and from 4 PM to midnight. Each data sample corresponds to one completed trip. It contains a total of 9 (nine) features, described as follows: 1) TRIP_ID: (String) It contains a unique identifier for each trip; 2) CALL_TYPE: (char) It identifies the way used to demand this service. It may contain one of three possible values: 'A' if this trip was dispatched from the central; 'B' if this trip was demanded directly to a taxi driver at a specific stand; 'C' otherwise (i.e. a trip demanded on a random street). 3) ORIGIN_CALL: (integer) It contains a unique identifier for each phone number which was used to demand, at least, one service. It identifies the trip's customer if CALL_TYPE='A'. Otherwise, it assumes a NULL value; 4) ORIGIN_STAND: (integer): It contains a unique identifier for the taxi stand. It identifies the starting point of the trip if CALL_TYPE='B'. Otherwise, it assumes a NULL value; 5) TAXI_ID: (integer): It contains a unique identifier for the taxi driver that performed each trip; 6) TIMESTAMP: (integer) Unix Timestamp (in seconds). It identifies the trip's start; 7) DAYTYPE: (char) It identifies the day type of the trip's start. It assumes one of three possible values: 'B' if this trip started on a holiday or any other special day (i.e. extending holidays, floating holidays, etc.); 'C' if the trip started on a day before a type-B day; 'A' otherwise (i.e. a normal day, workday or weekend). 8) MISSING_DATA: (Boolean) It is FALSE when the GPS data stream is complete and TRUE whenever one (or more) locations are missing; 9) POLYLINE: (String): It contains a list of GPS coordinates (i.e., WGS84 format) mapped as a string. The beginning and the end of the string are identified with brackets (i.e. [and], respectively). Each pair of coordinates is also identified by the same brackets as [LONGITUDE, LATITUDE]. This list contains one pair of coordinates for each 15 seconds of trip. The last list item corresponds to the trip's destination while the first one represents its start.

We are using a fragment of the dataset that was provided due to lack processing power After the acquisition of the data some preprocessing needed to be done [42]. Firstly, duplicate entries of trips are removed. Then a circle is created, centered in the center of Porto with a radius of 30 kilometers. All the entries outside the circle are discarded due to the long distance from the city center. After that, some of the smaller trips are filtered out because they will produce noise in our study. It is important to calculate and check the speed and acceleration on each instance of every trip in order to exclude those instances with non-realistic values.

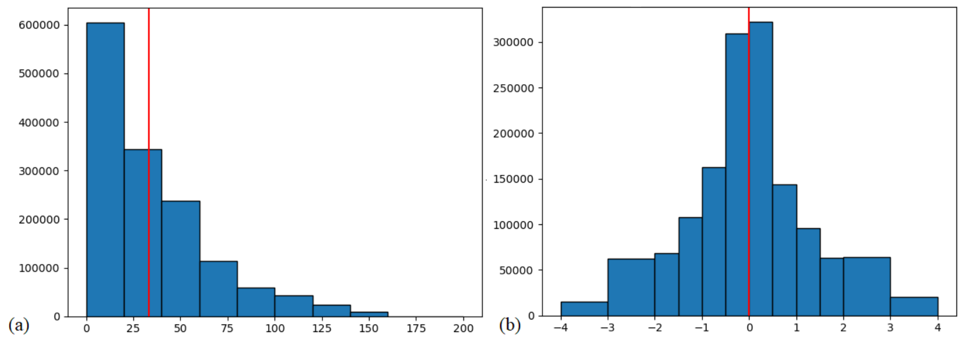

Figure 5 Basic statistics of the data
(a) speed distribution of the vehicles,
(b) acceleration distribution of the vehicles. The red lines are the mean of each graph

Table I Statistics of the dataset

Number of trips	Total number of instances	Number of instances per trip **max**	Number of instances per trip **mean**	Number of instances per trip **min**	Size
28,982	1,435,509	498	50	21	128,661 KB

3.2 Segmentation and compression with PCA

Having our data cleaned of outliers and preprocessed we can proceed with our pipeline to the analysis. The segmentation was applied to the dataset with different values of λ. This happened so we could compare the results of compression and clustering and relate them to λ in order to choose the best and most appropriate one for our data. A bigger value results in a wider range for compression, since the dimension reduction will be in the dimension of λ.

To select the least number of component while maintaining the most information out of our data two metrics were used to determine the PCs. Firstly, we found the explained variance of each principal component (Table II) i.e., the amount of the original data each component represents. This helps us define the number of PCs that are necessary in order to have a good representation. We decided that containing 99.6% of the original data was enough because from there on any number of additional components being added would reduce the compression ratio without significantly enrich the overall accuracy. Secondly, in addition to the cumulative scores of the principal components we are taking into account the root mean square error (RMSE) of each implementation (Table III). All the metrics are calculated with respect to the coordinates i.e., latitude, longitude.

Table II Explained and Cumulative variances of latitude obtained by the PCA

Principal components	Explained Variance λ=40	Cumulative Variance λ=40	Explained Variance λ=50	Cumulative Variance λ=50
PC1	0.899495	0.899	0.863018	0.863
PC2	0.084532	0.984	0.114611	0.978
PC3	0.009354	0.993	0.013407	0.991
PC4	0.002987	0.996	0.003844	0.995

Table III Root mean square error of various values of λ (length of each segment)

Value of λ	Root mean square error
20	0.003464
25	0.003794
30	0.004081
35	0.004523
40	0.003505
50	0.004122

The advantage of this process is that each of the coordinates can be stored in the form of principal components as we previously mentioned in Section 2. This has a significant reduction in size (Table IV, Table V) for each of the coordinates although the computation times are not negligible (Table VI). The computation times were similar to each other, so we are going to include just one from each type of process.

Table IV Compression of data with 4 principal components. The original size of the data (latitude, longitude) was 29,128 KB

Value of λ	Compressed size	Compression ratio	Space savings
20	13,321 KB	2.1866	54.27%
25	10,485 KB	2.7781	64%
30	8,842 KB	3.2943	69.64%
35	7,616 KB	3.8246	73.85%
40	6,415 KB	4.5406	77.98%
50	5,187 KB	5.6156	82.19%

Table V Compression of data with 3 principal components. The original size of the data (latitude, longitude) was 29,128 KB

Value of λ	Compressed size	Compression ratio	Space savings
40	6,415 KB	6.0848	83.57%
50	3,872 KB	7.5227	86.71%

Table VI Computation times (time in seconds)

Segmentation	Principal component analysis and dimensionality reduction	Reconstruction of table after PCA
1860 sec	9,52 sec	824 sec

To evaluate the results of PCA we used as a reference point the algorithm proposed by Douglas D, Peucker T [19]. The evaluation was done in terms of compression rate but also the percentage of information retained in the compressed file. Different values of epsilon were used in order to have a correlation of the the value of epsilon with the compression stream. The metric used to evaluate the retained information was as before the root mean square error (RMSE). The results are presented below (Table VII).

Table VII Compression with Ramer-Douglas-Peucker algorithm (RDP)

Value of epsilon	RMSE	Compressed size	Compression ratio	Space savings
0	0.000827	28,590 KB	1.0188	1.85%
0.00001	0.006826	24,478 KB	1.19	15.96%
0.0001	0.017919	15,165 KB	1.9207	47.94%

3.3 Clustering

The traditional application of DBSCAN in GPS data is done for each one of the points and has as a result the densest areas of the given dataset (Figure 6). This provides little to no information about the most common trajectories of vehicles. Also, in big data this technique has very long computational times due to its nature of trying to place each point to the appropriate cluster and then recalculate the cluster centers. In our approach, given that the data-set is segmented and then reduced to a couple of PCs the process is oriented towards the similarity of the trajectory rather than dense areas.

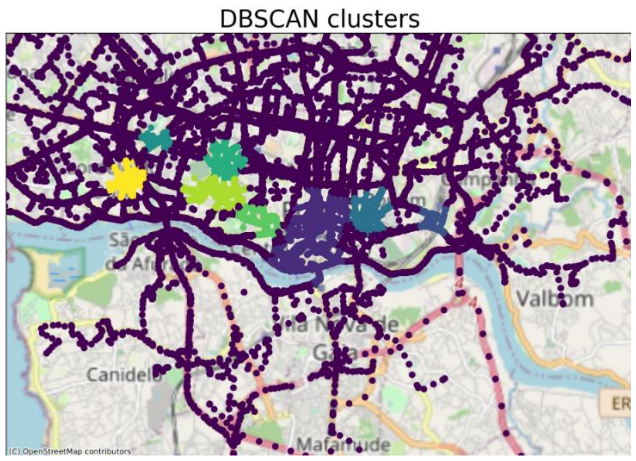

Figure 6 DBSCAN performed in a fragment of our data set (40000 points).

The main advantages of this approach during clustering are the ability to find more abstract and generalized clusters as well as the reduction of computational time. The clusters produced through our process are differentiated due to the different sizes of the segments and the different selection in the number of principal components. This is happening because dividing each trip into smaller sections results in the algorithm focusing on different types of motion when creating clusters. Reducing the dimensions, a generalization to the clusters is achieved. This is because the algorithm looks for clusters using less information (since the data set is compressed) resulting in cluster generalization. Original DBSCAN faces an innate problem with big data. This happens due to the nature of the algorithm; in each repetition, all the distances of the centroids are recalculated. This time is constantly increasing as entries are added to the model since times have to be calculated for them as well. This increase is negligible at first but accumulates and makes clustering for big data deterrent if the appropriate equipment is not available.

Our approach addresses the above problem of computation time by reducing the size of the original data set. Through the segmentation, the length of the data set is reduced but its width is increased keeping practically the same size. After PCA the width of the array decreases from a few million to a few thousand. We have chosen to apply clustering to this table, the table of the eigen vectors i.e., each principal component. Thus, saving significant amounts of time (Table VIII) as well as achieving a result of clusters focusing on the different motion that vehicles make (Figure 7, Figure 8, Figure 9, Figure 10, Figure 11).

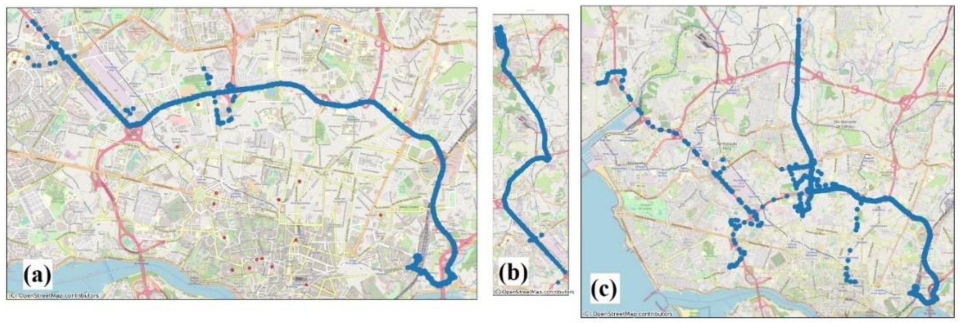

Figure 7 Plots of clusters using the original data without PCA.

a) Coordinate = latitude, λ = 40, min_samples = 20, max_eps = 7.25×10^{-9}
b) Coordinate = longitude, λ = 40, min_samples = 20, max_eps = 3×10^{-8}
c) Coordinate = longitude, λ = 40, min_samples = 20, max_eps = 3×10^{-8}

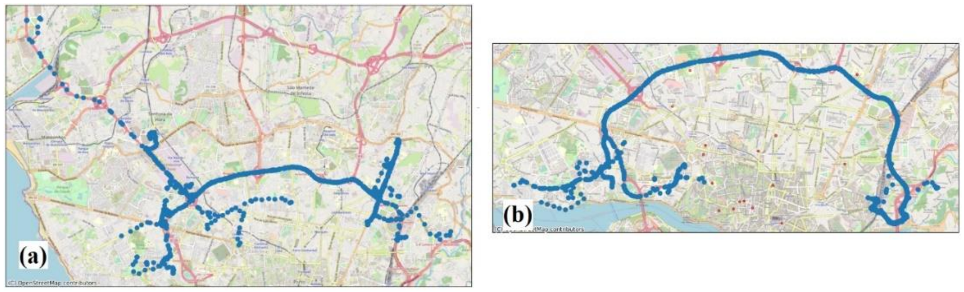

Figure 8 Plots of clusters using the original data without PCA.

a) Coordinate = longitude, λ = 50, min_samples = 20, max_eps = 5.13×10^{-9}

b) Coordinate = latitude, λ = 50, min_samples = 20, max_eps = 1.2×10^{-9}

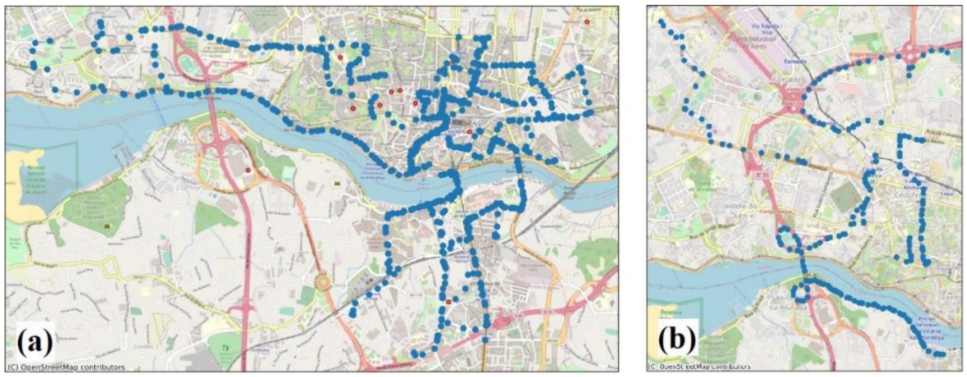

Figure 9 Plots of clusters using the scaled data.

a) Coordinate = latitude, λ = 40, Number of PC's = 4, min_samples = 20, max_eps = 0.0095

b) Coordinate = longitude, λ = 40, Number of PC's = 3, min_samples = 20, max_eps = 0.002

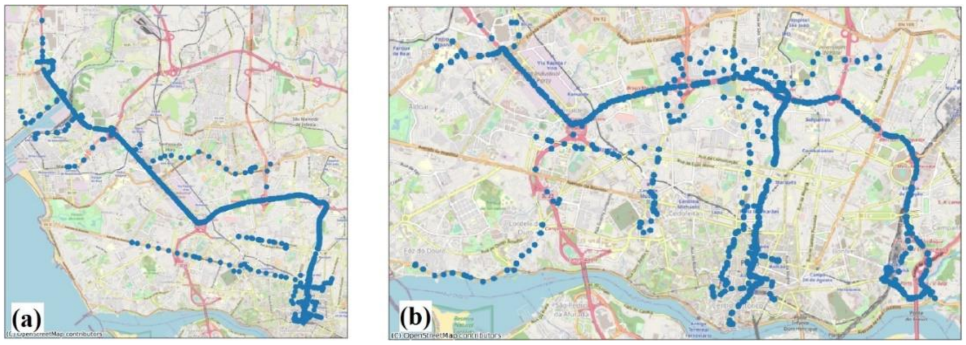

Figure 10 Plots of clusters using weighted data.

a) Coordinate = longitude, λ = 50, Number of PC's = 4, min_samples = 20, max_eps = 0.003

b) Coordinate = latitude, λ = 50, Number of PC's = 4, min_samples = 20, max_eps = 0.0025

Naturally, because the segmentation and PCA are applied separately to each set of coordinates, the clustering algorithm is also applied separately to latitude and longitude respectively. This allows the algorithm to find different patterns depending on the motion in latitude or longitude. Our experiments have shown that due to the local morphology of the city of Porto, clustering with respect to latitude has shown better results with more dense and discrete clusters than longitude.

We decided on not to scale the data of the principal components. This decision came from the fact that some principal components have much more variance within them than others. Specifically, PC1 represents more variance within the dataset than PC4 (~223 times more variance (Table II)). Adopting this approach enable us to do a weighted clustering (Figure 10, Figure 11).

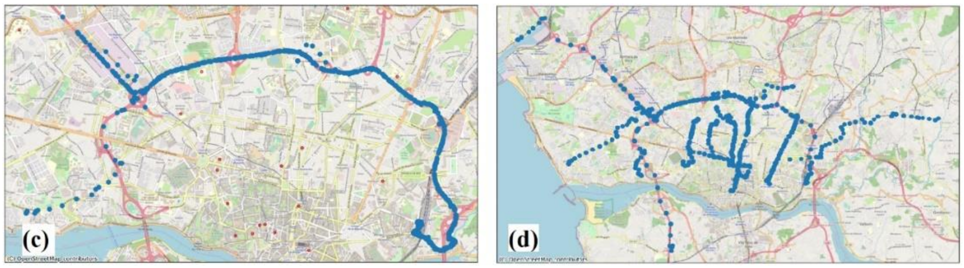

Figure 11 Plots of clusters using weighted data.

c) Coordinate = latitude, λ = 40, Number of PC's = 4, min_samples = 20, max_eps = 0.003

d) Coordinate = latitude, λ = 40, Number of PC's = 4, min_samples = 20, max_eps = 0.0025

This is evident when comparing Figure 9 and Figure 10 with the other Figures. Through the PCA and specifically from the accumulated information contained in the first few principal components we can identify clusters that otherwise would not be visible because they would be integrated into other clusters.

Table VIII Computation times of clustering

Type of dataset	Original dataset clustering	Clustering after segmentation and PCA
Time	65 seconds	50 seconds

4 Conclusion

This thesis has presented an approach on compressing and clustering trajectory data in the spectrum of big data. This was done by transforming the dataset of trajectories, from taxis of the city of Porto, into smaller segments and applying Principal Component Analysis to them. Furthermore, the dataset was clustered using the first few principal components therefore discovering unique clusters as well as reducing processing times of the algorithm. In addition, the reduced data had achieved a maximum compression ratio of 7.52 while retaining 99.1% of the original data.

4.1 Future work

The present work may acquire future extensions that can be divided into two categories, in terms of the type of data or the techniques used (dimensionality reduction - clustering).

In terms of data type, data from the trajectories of other types of vehicles could be used, such as buses of ordinary cars or even commercial ships or airplanes. With these different data we will be able to have different compression yields if they have similar trajectories. This is most pronounced in the airplane and ship trajectories mentioned above, as they often have predetermined trajectories.

Other techniques could have been applied in terms of dimensional reduction as well as clustering. PCA is the most common technique in the literature. alternatively, SVD (singular value decomposition) could be used which has a lot in common with PCA as well as ICA (independent component analysis) which would give more separate components and thus more important. Additionally, PCA using kernel functions would give different results.

Regarding clustering, different clustering algorithms based on diversity such as optics could be used. Tests and comparisons could also be made by clustering via a neural network

References

1. Worldometers (2020). Current world population. Last accessed on Dec 03, 2016, http://www.worldometers.info/world-population/.
2. Statista (2020). Curent smartphone users https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
3. Yaqoob, I., Hashem, I. A. T., Gani, A., Mokhtar, S., Ahmed, E., Anuar, N. B., et al. (2016). Big data: From beginning to future. International Journal of Information Management, 36(6), Part B), 1231–1247.
4. Kreps, D., & Kimppa, K. (2015). Theorising web 3.0: ICTs in a changing society. Information Technology & People, 28(No. 4), 726–741.
5. Li, J., Tao, F., Cheng, Y. et al. Big Data in product lifecycle management. Int J Adv Manuf Technol 81, 667–684 (2015). https://doi.org/10.1007/s00170-015-7151-x
6. Meratnia N, de By RA (2004) Spatiotemporal compression techniques for moving point objects. In: Proceedings of the 9th international conference on extending database technology (EDBT), pp 765–782
7. Abdelguerfi M, Givaudan J, Shaw K, Ladner R (2002) The 2-3TR-tree, a trajectory-oriented index structure for fully evolving valid-time spatio-temporal datasets. In: Proceedings of the 10th SIGSPATIAL international conference on advances in geographic information systems (ACM-GIS), pp 29–34
8. Agarwal PK, Guibas LJ, Edelsbrunner H, Erickson J, Isard M, Har-Peled S, Hershberger J, Jensen C, Kavraki L (2002) Algorithmic issues in modeling motion. ACM Comput Surv 34:550– 572
9. Zhu H, Su J, Ibarra OH (2002) Trajectory queries and octagons in moving object databases. In: Proceedings of the 11th conference on information and knowledge management (CIKM), pp 413–421
10. Giannotti F, Nanni M, Pinelli F, Pedreschi D (2007) Trajectory pattern mining. In: Proceedings of the 13th international conference on knowledge discovery and data mining (ACM-KDD), pp 330–339
11. Python 3.8 (2020) programing language https://www.python.org
12. Pandas 1.1.5 (2020) https://pandas.pydata.org
13. Geopandas 0.8 (2020) https://geopandas.org/#
14. Matplotlib 3.3.3 (2020) https://matplotlib.org/
15. Scikit-learn 0.23.2 (2020) https://scikit-learn.org/
16. Psycopg2 2.8.6 (2020) https://www.psycopg.org/docs/
17. PostgreSQL 11.10 (2020) relational database management system https://www.postgresql.org/
18. Postgis 2.5.1 (2020) extension of postgreSQ https://postgis.net/
19. Douglas D, Peucker T (1973) Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Can Cartogr 10(2):112–122

20. Hershberger, J., Snoeyink, J. (1992) Speeding up the Douglas-Peucker line-simplification algorithm. In: Proceedings of the 5th SDH. Volume 1., Charleston, South Carolina, USA, University of South Carolina
21. Potamias M, Patroumpas K, Sellis TK (2006) Sampling trajectory streams with spatiotemporal criteria. In: Proceedings of SSDBM
22. Chen Y, Jiang K, Zheng Y, Li C, Yu N (2009) Trajectory simplification method for location-based social networking services. In: Proceedings of LBSN
23. Kellaris G, Pelekis N, Theodoridis Y (2013) Map-matched trajectory compression. J Syst Softw 86(6):1566–1579
24. Giannotti F, Pedreschi D (2008) Mobility, data mining and privacy, geographic knowledge discovery. Springer, Berlin
25. Pelekis N., Theodoridis Y. (2014) Mobility Data Management and Exploration. Springer, New York, NY.
26. Gaffney S, Smyth P (1999) Trajectory clustering with mixtures of regression models. In: Proceedings of KDD
27. Cadez V, Gaffney S, Smyth P (2000) A general probabilistic framework for clustering individuals and objects. In: Proceedings of SIGKDD
28. Nanni M, Pedreschi D (2006) Time-focused clustering of trajectories of moving objects. J Intell Inf Syst 27(3):267–289
29. Pelekis N, Kopanakis I, Kotsifakos E, Frentzos E, Theodoridis Y (2009) Clustering trajectories of moving objects in an uncertain world. In: Proceedings of ICDM
30. Pelekis N, Kopanakis I, Kotsifakos E, Frentzos E, Theodoridis Y (2011) Clustering uncertain trajectories. Knowl Inf Syst 28(1):117–147
31. Lee JG, Han J, Whang KY (2007) Trajectory clustering: a partition-and-group framework. In: Proceedings of SIGMOD
32. Li X, Han J, Lee JG, Gonzalez H (2007) Traffic density-based discovery of hot routes in road networks. In: Proceedings of SSTD
33. Han B, Liu L, Omiecinski E (2012) NEAT: road network aware trajectory clustering. In: Proceedings of ICDCS
34. Sacharidis D, Patroumpas K, Terrovitis M, Kantere V, Potamias M, Mouratidis K, Sellis T (2008) On-line discovery of hot motion paths. In: Proceedings of EDBT
35. Chen Z, Shen HT, Zhou X (2011) Discovering popular routes from trajectories. In: Proceedings of ICDE
36. Georgiou (2017) 1-pass tool - linear interpolation
37. Ian Jolliffe (2005) Principal Component Analysis, Second Edition
38. Bengio, Y.; et al. (2013). "Representation Learning: A Review and New Perspectives". IEEE Transactions on Pattern Analysis and Machine Intelligence, 35 (8): 1798–1828.
39. M. Ester, H.-P. Kriegel, J. Sander, X. Xu (1996) A density-based algorithm for discovering clusters in large spatial databases with noise, In: Proceedings of Second International Conference on Knowledge Discovery and Data Mining, pp. 226–231.
40. Derya Birant, Alp Kut (2006) ST-DBSCAN: An algorithm for clustering spatial–temporal data

41. Luis Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luis Damas (2013): Predicting Taxi–Passenger Demand Using Streaming Data. IEEE Transactions on Intelligent Transportation Systems
42. Taxi service trajectory prediction challenge (2020) http://www.geolink.pt/ecmlpkdd2015-challenge/dataset.html