applied sciences

Article

Clustering Moving Object Trajectories: Integration in CROSS-CPP Analytic Toolbox

Alberto Blazquez-Herranz [1], Juan-Ignacio Caballero-Garzon [2,*], Albert Zilverberg [3], Christian Wolff [3], Alejandro Rodríguez-Gonzalez [1,4] and Ernestina Menasalvas [1,4]

1 Centro de Tecnología Biomédica, Universidad Politécnica de Madrid, 28223 Madrid, Spain; abherranz@upm.es (A.B.-H.); alejandro.rg@upm.es (A.R.-G.); ernestina.menasalvas@upm.es (E.M.)
2 Escuela Técnica Superior de Ingenieros Caminos, Canales y Puertos, Universidad Politécnica de Madrid, 28040 Madrid, Spain
3 ATB Bremen, 28359 Bremen, Germany; zilverberg@atb-bremen.de (A.Z.); wolff@atb-bremen.de (C.W.)
4 Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, 28660 Madrid, Spain
* Correspondence: juanignacio.caballero@upm.es

Abstract: Mobile devices equipped with sensors are generating an amount of geo-spatial related data that, properly analyzed can be used for future applications. In particular, being able to establish similar trajectories is crucial to analyze events on common points in the trajectories. CROSS-CPP is a European project whose main aim is to provide tools to store data in a data market and to have a toolbox to analyze the data. As part of these analytic tools, a set of functionalities has been developed to cluster trajectories. Based on previous work on clustering algorithms we present in this paper a Quickbundels algorithm adaptation to trajectory clustering . Experiments using different distance measures show that Quickbundles outperforms spectral clustering, with the WS84 geodesic distance being the one that provides the best results.

Keywords: trajectory mining; mobile data; clustering; anomaly detection

check for updates

Citation: Blazquez-Herranz, A.; Caballero-Garzon, J.I.; Zilverberg, A.; Wolff, C.; Rodríguez-González, A.; Menasalvas, E. Clustering Moving Object Trajectories: Integration in CROSS-CPP Analytic Toolbox. *Appl. Sci.* **2021**, *11*, 3693. https://doi.org/10.3390/app11083693

Academic Editor: Carla Raffaelli

Received: 24 March 2021
Accepted: 15 April 2021
Published: 20 April 2021

1. Introduction

Nowadays, there is a growing number of devices connected to the internet, and an exponential rise is expected, along with the development of 5G communication standards; see [1]. The availability of data and the increase in computation capacity have boosted the proliferation of new applications. Geolocation services have attracted great interest due to the easy collection of geopositional data through all kind of devices. This type of data can proceed from vehicle services http://vis.cs.kent.edu/demo/list_of_trajectory_datasets.pdf (accessed date 17 April 2021), electrical micro-mobility (bikes and scooters) https://www.kaggle.com/pronto/cycle-share-dataset/code, 17 April 2021, people-collected data https://archive.ics.uci.edu/ml/datasets/GPS+Trajectories (accessed date 17 April 2021), and public transport https://data-crtm.opendata.arcgis.com/ (accessed date 17 April 2021). These new services have realized the relevance of analyzing moving patterns of their customers as a key characteristic to improve their value proposition and competitivity. Obtaining knowledge from moving agents' data has applications such as bus routes optimization [2], prediction of collective behavior [3,4], customized trips, or fleet management in the mobility field [5,6]. Additionally, the maturity of the mentioned services is impacting mobility in cities. The study of citizens' movements patterns may push forward more customized transportation [7]. This is a part of the more global concept of smart transportation and smart cities [8].

The surge in data from moving objects faces several challenges. One of them is how to extract useful insights of the data. Clustering of trajectories refers to grouping them based on a similarity measure [9]. This is a task that serves as a exploratory step in the discovery of patterns in moving objects and enables one to synthesize the information by establishing a representative trajectory for each cluster. By doing so, it serves as a base from which to start a deeper analysis of the characteristics of the trajectories.

A second challenge is related to quality assurance and trust. Among the diversity and randomness typical of the data of moving objects, it is vital to establish the data that correspond to real behaviors and those that are due to errors in the sensors or systems. This task is referred to as outlier/anomaly detection. It is domain-specific and is also used for detection of fraudulent activities, since these activities usually fall away from expected common behavior.

Another challenge in the application of trajectory data analysis is the development of user-friendly tools that enable the development of services to extract knowledge from integrated data sources. An obstacle is the dispersion of libraries due to most of them being domain-specific. Moradi et al. [10] mentioned more than 20 different packages in R, mostly focused on animal trajectories. In their work, they propose a set of data structures, classes and methods to deal with generic trajectory data. Another option is the MovingObjects library [11] implemented in python, which is based in the Geopandas data structures. The mentioned libraries do not offer no-code options, and thus, they present a learning curve for non-tech users. The open-source TrajAnalytics software [12] offers a web-based visualization interface that simplifies the analysis process for these users. In this work, we present the CROSS-CPP projecthttps://www.cross-cpp.eu/, 17 April 2021; its main aim is the development of a system for the integration and analytics of data streams coming from high volume (mass) products with cyber physical features. Its purpose is to offer new cross-sectorial services and focus on the commercial confidentiality, privacy and ethical issues using a context-sensitive approach. In the framework of this project, a toolbox has been developed to provide the analysis services. It has been designed as a easy-to-use tool thanks to its user interface that enables to analyze moving objects' anonymized data. The toolbox includes algorithms for different machine learning tasks, on both stream and batch mode. In particular, we focus in this paper on the clustering of moving objects.

Quickbundles [13] is known to be a clustering algorithm for tractography segmentation that has shown good performance in the health domain. Tractografies, like moving object trajectories, may be defined as streamlines. Thus, we propose in this paper to adapt Quickbundles for trajectory clustering and to integrate it into CROSS-CPP analytic toolbox. The adaptation is not straightforward, and some challenges have to be faced: (i) Quickbundles assumes streamlines being defined by a settled number of points, and as a consequence, in this paper, we solve the problem to define different trajectories under the same number of points; (ii) for the distance among streamlines, the measure of the distances is adjusted for moving object trajectories.

The paper presents the results of the experiments carried out with two datasets: (i) Porto Kaggle dataset https://www.kaggle.com/crailtap/taxi-trajectory. (accessed date 17 April 2021) and (ii) CROSS-CPP dataset (Due to privacy constrains the dataset is not publicly available). We also show how the integration looks like in CROSS-CPP and how it can be accessed. Consequently, the main contributions of the paper are as follows: (i) the adaption of Quickbundles algorithm for mobile data, (ii) the analysis of its performance in real trajectories datasets and (iii) the integration into the CROSS-CPP environment.

The rest of the paper is organized as follows: Section 2, Related Work, reviews the approaches for clustering trajectories, the used distance measures and models of the Earth surface. Section 3, Preliminaries , include the theory related with the topic and in particular trajectory definition, distance measures and comparison of trajectories. Section 4 contains the datasets' description, the preprocessing of the trajectories data, the validation metrics and results. In Section 5, it is explained how a non-tech person can perform the end-to-end process of harvesting the data and perform a clustering analysis in the CROSS-CPP platform. Thanks to a graphic user-interface, the learning-curve is lower for non-tech professionals. The last Section 6 discusses the results, analyzes the pros and cons of the proposed approach, highlights future work and extracts conclusions.

2. Related Work

As mentioned above, the proposed approach adapts the algorithm presented in [13] for clustering and performs anomaly detection on that basis. Consequently, in what follows, we review previous approaches for trajectory clustering and detection of anomalies.

2.1. Clustering of Trajectories

Trajectory clustering is a common practice in trajectory data mining [9,14]. It consists in grouping the trajectories based on a similarity measure. The nature of trajectories as multidimensional data enables different clustering approaches that may be domain- or application-specific.

2.1.1. Distances between Trajectories

An important aspect in the clustering of trajectories is the computing of the distances among them. The first point to be accounted for is that moving objects travel the Earth in most cases. Therefore, the distance metric may consider the curvature of its surface. This is the case for the great circle and ellipsoids geodesic distance. In other cases, the effect of the curvature of the Earth is negligible, so the most used metric is the euclidean, but there are others like the Hausdorff or the longest common subsequence [9].

Distances between two points on the Earth: in geometry, the ancient Greeks defined the geodesic as the shortest distance between two points on a surface. There are several formulas to calculate the geodesic for the surface of the Earth, with lesser or greater error depending on the position on the Earth's surface. In this regard, the most simple model is to approximate the Earth as a sphere; then, the geodesic distance is also called the great-circle or orthodromic distance [15]. Another approach is to consider an ellipsoidal model of the Earth; then again, there are several models of ellipsoids with variable error depending on the position. The WGS-84 ellipsoid is accepted as the most globally accurate.

In many cases, trajectories are constrained to spatial locations of the size of an area small enough to assume it is flat; these cases neglect the curvature of the planet. This case the most common in the literature with examples like taxi routes analysis; see [6] for a review of different approaches.

2.1.2. Spatial Clustering of Trajectories

Once the distances among trajectories are computed and stored, different clustering algorithms may be applied. Each of them offers pros and cons, and the best solution is domain-specific.

This project focuses on geometric properties of the trajectories, defined by characteristics like spatial distribution, proximity or length. The predominant algorithms are those based on density such as DBSCAN [16] or K-means [17], where the proximity is the most important feature. One of the drawbacks of K-means is its tendency to form spherical clusters, which is inadequate for clustering several streamlines as in the present case.

For clusters with irregular or non-spherical shapes, one of the most efficient algorithms is the spectral clustering [18]; it uses the eigenvalues, also referred to as the spectrum.

Quickbundles [13], presented before, is a clustering algorithm developed to reduce the dimensionality of brain streamlines datasets provided by tractography. It stands out due to its linear complexity with the number of streams and its interpretability.

3. Preliminaries

Prior to presenting the approach to cluster trajectories, some preliminary definitions are introduced.

A trajectory is the path followed by a moving object. Formally, a trajectory s is defined as a chronological sequence of multi-dimensional locations, which is denoted by $s = P1, P2, ..., P_n$, with n being the number of points. Each point P_n is represented as $Location_j$-tuple at time T_j [9].

The concept of distance among trajectories refers to the criteria to measure the similarity of trajectories. In this paper, the similarity focuses on the geometric patterns of trajectories. Two trajectories are identical if they have the same length and are defined by the same positions independently of the time. This kind of similarity is well-suited for analysis of routes in cities and countries, such as commuting routes from homes to work places, hot-spot areas and holiday road trips.

The geodesic distance is the shortest distance between two points located on the surface of the Earth, and there are several models to compute this distance:

- Harvesine, or half of the versine, which is the central angle (Θ) between two points in an sphere:
$$\Theta = \frac{d}{r} \tag{1}$$
where d is the distance between two points along a great circle of the sphere and r is the radius of the sphere.
The Haversine $(hav(\Theta))$ is computed directly from the latitude and longitude of the two points:
$$hav(\Theta) = hav(\varphi_2 - \varphi_1) + \cos\varphi_1 \cos\varphi_2\, hav(\lambda_2 - \lambda_1) \tag{2}$$
where φ_2, φ_1 are the latitude of point 1 and 2. λ_2, λ_1 is the longitude of points 1 and 2. The haversine function computes half a versine of the angle θ:
$$hav(\theta) = \sin^2\frac{\theta}{2} \tag{3}$$
Then the distance between the two points:
$$\begin{aligned} d = r\, archav(hav(\Theta)) = 2r\ \arcsin(\sqrt{hav(\Theta)}) \\ d = 2r\ \arcsin(\sqrt{hav(\varphi_2 - \varphi_1) + \cos\varphi_1 \cos\varphi_2\, hav(\lambda_2 - \lambda_1)})) \end{aligned} \tag{4}$$
- Great circle distance [15]: In geometry, the great circle is defined as the intersection between a plane and a sphere that passes through the center of the sphere. This formula was manually implemented, and it is defined as follows:
$$d = R \arccos\left(\sin\varphi_2 * \sin\varphi_1 + \cos\varphi_2 * \cos\varphi_1 * \cos\left(\lambda_2 - \lambda_1\right)\right) \tag{5}$$
where the d is the distance, R is the Earth radius, a constant with $R = 6371$ km, and the same notation as before for the latitude (φ) and longitude (λ) of each point.
- Euclidean: by neglecting the curvature of the Earth, the euclidean distance between the two points is defined:
$$d = \sqrt{(\varphi_2 - \varphi_1)^2 + (\lambda_2 - \lambda_1)^2} \tag{6}$$
- Geodesic WS84 [15]: in this case, the model of the Earth is an ellipsoid; in particular, the WS84 is the most globally accurate model. The used implementation is that included in the GeoPy library.

All the metrics defined above are suitable to calculate distances between two points. To compute distance among trajectories, different approaches have been proposed [9,19,20]. These approaches distinguish between trajectories defined by the same number of points or dissimilar. In the case of trajectories defined by the same number of points and lineal time complexity, there are two distances that have been used in moving objects data:

the euclidean and the Hausdorff [21]. In this paper, we explored the use of Minimum Direct-Flip (MDF) [13], see Equation (7).

$$\begin{aligned} d_{direct}(s,t) = d(s,t) &= \frac{1}{K}\sum |s_i - t_i|, \\ d_{flipped}(s,t) = d(s,t^F) &= d(s^F,t) \\ MDF(s,t) &= min(d_{direct}(s,t), d_{flipped}(s,t)) \end{aligned} \tag{7}$$

where $|s_i - t_i|$ denotes the Euclidean distances between two points, s is a trajectory and t is the mean of the Euclidean distance. Figure 1 provides an insight of the working of the metric.

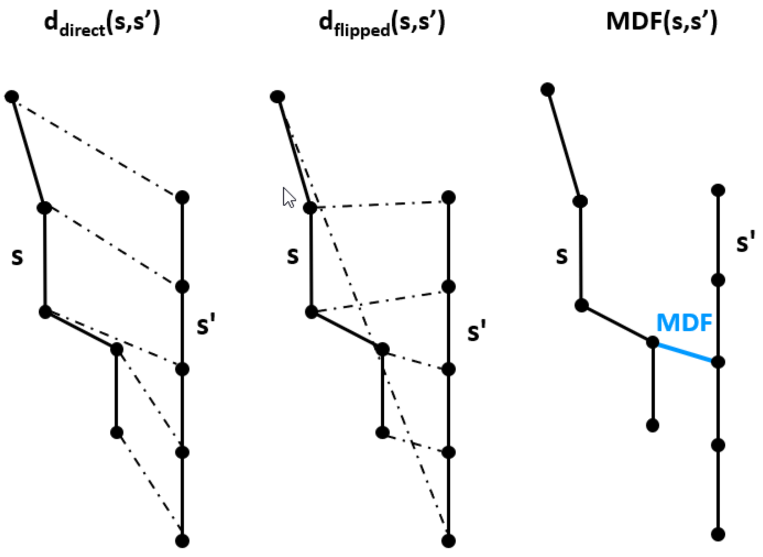


Figure 1. Operation of MDF.

Quickbundles

Quickbundles (QB) [13] is a clustering algorithm with linear time performance with respect to the number of trajectories.

QB computing time depends lineally to the number of samples because (i) it uses MDF with linear complexity and (ii) there is no reassignment of the trajectories clustering. This means that trajectories are evaluated only once when performing the clustering and opposed to K-means, where there is amalgamating phase. Clusters are defined by a triplet $c = (I, h, n)$, where I is a list of indexes of the trajectories in the cluster, n is the number of trajectories in the cluster, and h is the trajectories sum. h is a $Kx2$ matrix that can be updated for new trajectories (see Equation (8)).

$$h = \sum_{n=1}^{n} s_i \tag{8}$$

This way, the centroid of the cluster is defined by v:

$$vs. = \frac{h}{n} \tag{9}$$

Among the strengths of this clustering algorithm, we have its above-mentioned lineal complexity, the easiness to interpret the clusters and the versatility provided by changing the number of points that define each trajectory and the threshold value to consider trajectories being similar. All of these become very useful in the exploration of a dataset of trajectories.

However, QB was developed to overcome the complexity of segmentation of the brain fibers. In order to apply it to cluster trajectories, the following are required:

- All trajectories have be defined by the same number of points.
- A distance threshold to specify the maximum value to which two trajectories are clustered together has to be defined.

4. Experiments

4.1. DataSet Description

The experiments presented in this section were performed using the Kaggle dataset of taxi trajectorie https://www.kaggle.com/crailtap/taxi-trajectory, (accessed date 17 April 2021). both It contains trajectory information from 1 July 2013 to 30 June 2014 for 442 taxis running in the city of Porto, in Portugal. In total, 1,704,757 trajectories are available.

Each register contains 9 features, described as follows:

- TRIP_ID: (string) It contains an unique identifier for each trip.
- CALL_TYPE: (char) It identifies the way used to demand the taxi service. It may contain one of three possible values:
  - A: if the trip was dispatched from the central.
  - B: If the trip was demanded on a specific stand.
  - C: Otherwise; i.e., a trip was demanded on a random street.
- ORIGINCALL (integer)
- TAXI_ID (integer):
- TIMESTAMP (integer) Unix Timestamp (in seconds). It identifies the trip's start;
- DAYTYPE (char)
- MISSING_DATA (Boolean)
- 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 s of trip. The last list item corresponds to the trip's destination, while the first one represents its start. Some trips have missing data points, in which the MISSING_DATA column is TRUE.

For this paper, the TIMESTAMP and POLYLINE features were used.

CROSS-CPP Dataset Description

The original unprocessed data included 6094 different trajectories from 480 vehicles collected from May 2019 to the October of 2021. The only attribute of the data is the GPS streamline.

During the preprocessing step, it was found out that 742 trajectories were defined with less than 100 coordinates, and they were removed. In conclusion, 5852 trajectories were taken into account for the final experimentation.

4.2. Parameter Setting

Experiments were carried out to compare the behaviour of two clustering algorithms: Quickbundles and spectral clustering.

In order to validate results, ten different experiments with batches of 2000 trajectories were conducted for each algorithm and distance metric. A total of 80 experiments were run (10 per pair algorithm-distance metric). The mean Silhouette Coefficients of each set of the 10 experiments carried out was analyzed.

To build each experimentation batch, a different approach was set depending on the data source:

- Kaggle data: since there were 1,704,757 trajectories, each experiment run for an algorithm-distance metric pair comprised 2000 different trajectories each time, guaranteeing no repetition of trajectories in each experiment.

- Cross-CPP data: data from only 5852 remained for experimentation after filtering, and each experiment was run over a subset of 2000 randomly selected trajectories, unavoidably repeating trajectories between experiments.

In each experiment, the hyperparameters of QuickBundles were adjusted via a 10-fold cross-validation approach. The adjusted hyperparameters were:

- Resampling factor: number of coordinates to which each trajectory will be resampled (as the algorithm requires all trajectories to have the same number of coordinates).
- Neighboring threshold: distance at which two trajectory points are considered neighbors.

The spectral clustering implementation used the radial basis function (RBF) kernel for its construction. It is to note that, for the Spectral Clustering algorithm, only the number of clusters hyperparameter was checked. Results are displayed for its maximum silhouette coefficient value.

4.3. Validation Metrics

The silhouette was the validation metric used, based on the definition of Rousseeuw [22]. i is a data point in the cluster C_i. a is the mean distance between i and all other points in the cluster:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j) \tag{10}$$

The mean dissimilarity of point i to some cluster C_k is the mean of the distance from i to all points in C_k. b is the smallest distance of i to a cluster different from C_i:

$$b(i) = min_{k \neq i} \frac{1}{C_k} \sum_{j \in C_k j} d(i,j) \tag{11}$$

Then, the silhouette value for one point i is given by:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}, \text{ if } |C_i| > 1 \tag{12}$$

The term silhouette coefficient is defined as the maximum value of the mean $s(i)$ over all data of the entire dataset:

$$SC = \max_k t(k) \tag{13}$$

where $t(k)$ represents the mean $s(i)$ over all data for the number of clusters k.

4.4. Results

The following tables show the average of the silhouette coefficient for the clustering algorithms and the distance measures in each of the datasets (Tables 1 and 2).

Table 1. Table Silhoutte results for clustering algorithms for Kaggle dataset.

Distance Metric	Quickbundles	Spectral
Harvesine	0.753	0.479
Great Circle	0.772	0.332
Euclidean	0.732	0.514
Geodesic WS84	0.812	0.601

Table 2. Table Silhoutte results for clustering algorithms for CROSS-CPP dataset.

Distance Metric	Quickbundles	Spectral
Harvesine	0.685	0.378
Great Circle	0.742	0.401
Euclidean	0.698	0.442
Geodesic	0.753	0.396

One can observe that Quickbundles clearly algorithm ourperforms Spectral clustering. Observe that the geeodesic distance clearly yields the best results even when points to calculate distances in the dataset are not much apart.

5. Integration in CROSS-CPP

As was mentioned in Section 1, the CROSS-CPP is an IT environment for the integration and analytics of data streams coming from high volume (mass) products with cyber physical features. Consequently, in this section, we will firstly review the process that has to be followed to harvest data and then make available for the data analytics toolbox to show how the proposed clustering algorithm has been integrated in Cross-CPP. Once we describe the integration on the CROSS-CPP framework, we will analyze how to run and the results of different executions using real datasets.

In what follows, the process performed to calculate clustering of trajectories of the data available at the platform is analyzed. The process is divided into two phases: Obtaining access to the data (data harvesting) and performing the desired analysis—clustering in this case.

Data Harvesting

The accessing data phase started by logging into the Agora website at the "Home" tab and choosing vehicles' origin data. Figure 2 shows the screen where basic statistics are displayed for the selected data:

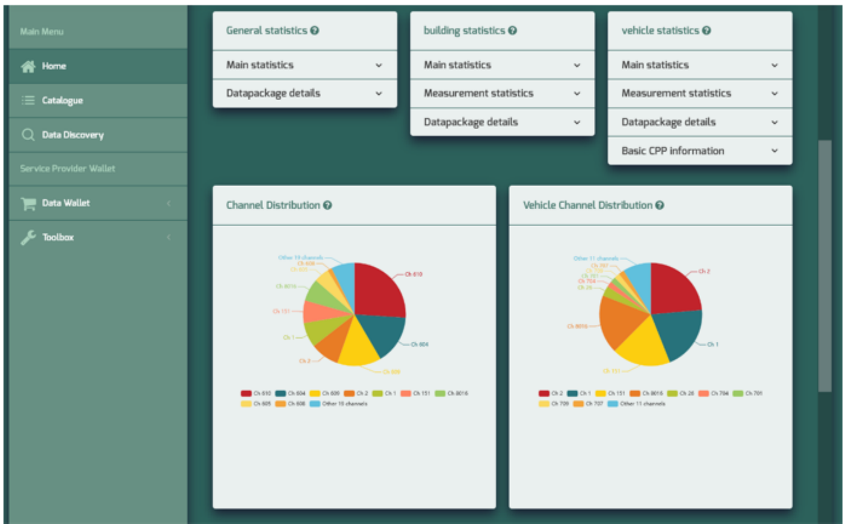
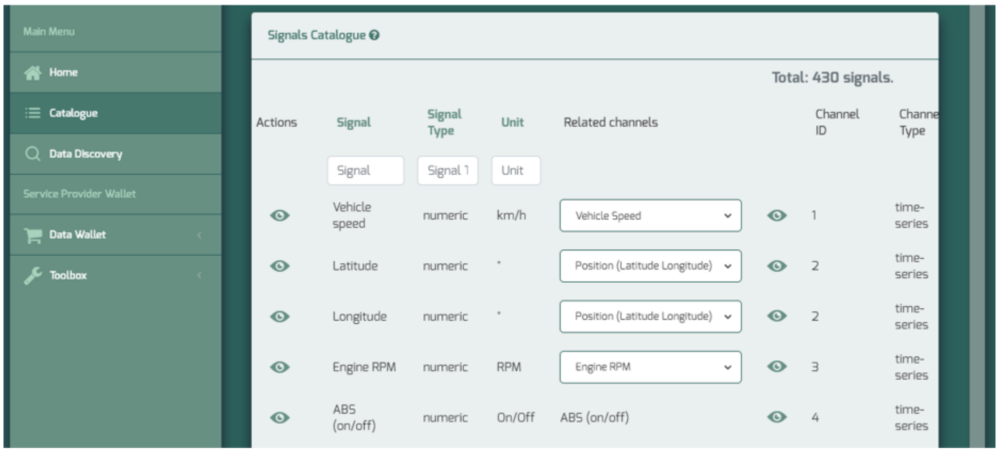


Figure 2. Data selection.

The process continued by moving to the "Data Discovery" tab and selecting the type of signal to be used. Filtering by vehicles, each signal contains the values for a vehicle features, i.e., vehicle speed, latitude, and temperature. This information is contained in a channel with an unique identifier. The "position", channel number two, signal was chosen, containing the latitude and longitude as can be seen in Figure 3.



Signals Catalogue

Total: 430 signals.

Actions	Signal	Signal Type	Unit	Related channels		Channel ID	Channe Type
	Signal	Signal 1	Unit				
👁	Vehicle speed	numeric	km/h	Vehicle Speed	👁	1	time-series
👁	Latitude	numeric	°	Position (Latitude Longitude)	👁	2	time-series
👁	Longitude	numeric	°	Position (Latitude Longitude)	👁	2	time-series
👁	Engine RPM	numeric	RPM	Engine RPM	👁	3	time-series
👁	ABS (on/off)	numeric	On/Off	ABS (on/off)	👁	4	time-series

Figure 3. Available data attributes, referred as signals in the platform.

Once the channel was selected, no geographical or temporal filters were added in the example. Then, the platform provided useful insights in the form of a heatmap for geographical reference and a slide bar for the time distribution of the trajectories as shown in Figure 4.

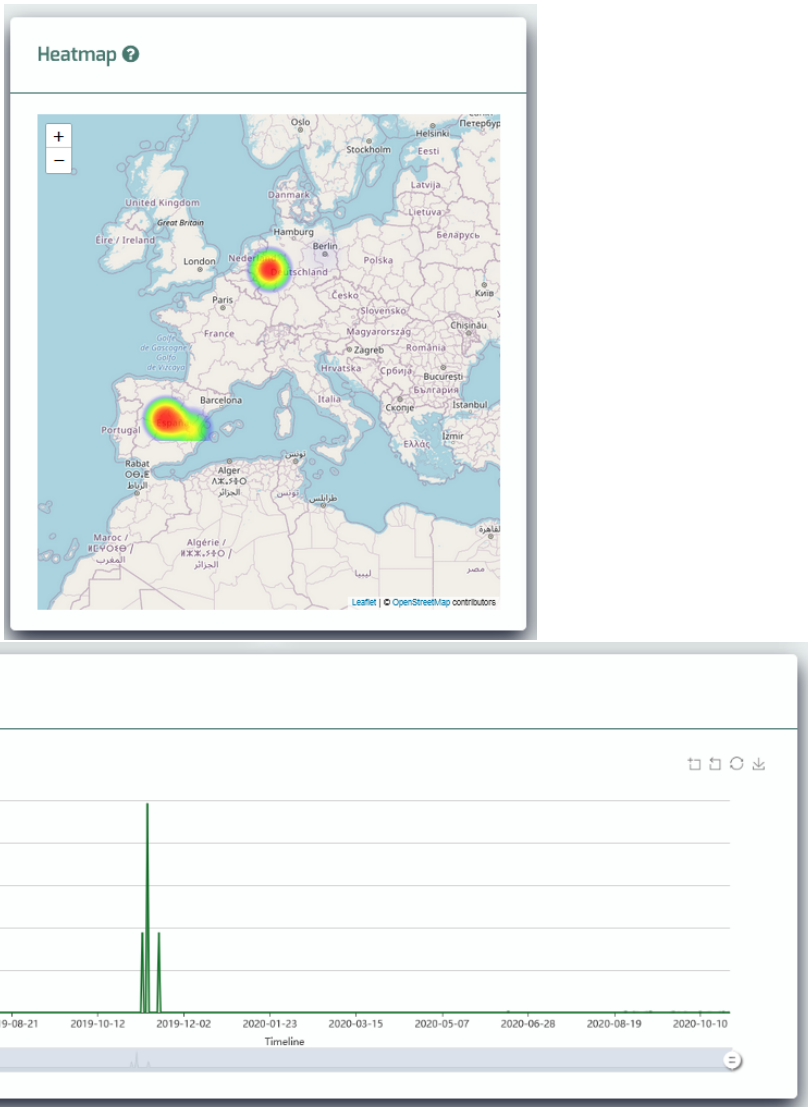
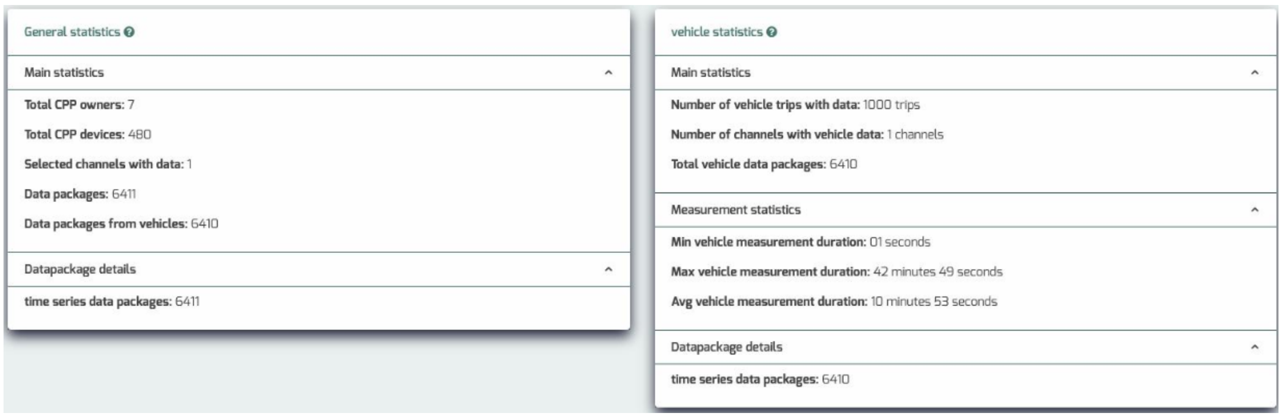


Figure 4. Heatmap of trajectories dataset and time distribution.

In addition, general statistics are also given as can be observed in Figure 5.



General statistics

Main statistics

Total CPP owners: 7

Total CPP devices: 480

Selected channels with data: 1

Data packages: 6411

Data packages from vehicles: 6410

Datapackage details

time series data packages: 6411

vehicle statistics

Main statistics

Number of vehicle trips with data: 1000 trips

Number of channels with vehicle data: 1 channels

Total vehicle data packages: 6410

Measurement statistics

Min vehicle measurement duration: 01 seconds

Max vehicle measurement duration: 42 minutes 49 seconds

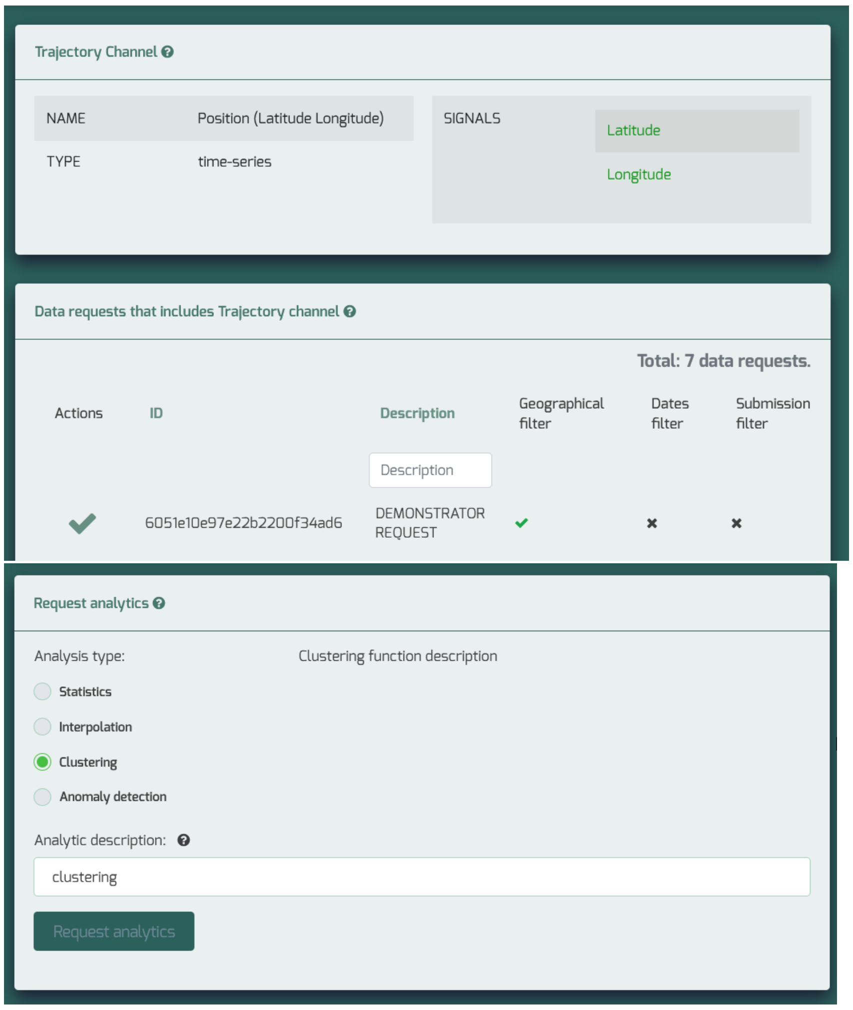
Avg vehicle measurement duration: 10 minutes 53 seconds

Datapackage details

time series data packages: 6410

Figure 5. Insights of the trajectories data.

The analytic phase starts upon the approval of the data request. In the "Toolbox" and trajectories tab, the above-mentioned data request can be selected. This provides the option of adding temporal or geographical filters again, although in our example, none of these options were used . On the other hand, in the analysis option, we selected "Clustering", as it can be observed in Figure 6, in order to execute the clustering.



Trajectory Channel

NAME	Position (Latitude Longitude)	SIGNALS	Latitude
TYPE	time-series		Longitude

Data requests that includes Trajectory channel

Total: 7 data requests.

Actions	ID	Description	Geographical filter	Dates filter	Submission filter
		Description			
✓	6051e10e97e22b2200f34ad6	DEMONSTRATOR REQUEST	✓	✖	✖

Request analytics

Analysis type: Clustering function description

- ○ Statistics
- ○ Interpolation
- ● Clustering
- ○ Anomaly detection

Analytic description: clustering

Request analytics

Figure 6. Illustration of the main page to perform analytics.

6. Discussion and Conclusions

In this paper, we have presented results of the integration Quickbundles algorithm for the moving objects' trajectory clustering in the CROSS-CPP platform. Two main aspects must be discussed: (i) integration issues and (ii) clustering algorithm performance.

In relation to integration, the CROSS-CPP marketplace makes it possible for data providers to store data so that service providers can use these data to generate applications based on the knowledge extracted from these data. In order to extract the knowledge from the data, the analytic toolbox has been developed and integrated. One of the main advantages of the analytic toolbox is its flexibility to include new functionalities; in particular, in this paper, we have shown how clustering for moving objects has been integrated. More precisely, the Quickbundles algorithm has been adapted from the health domain to calculate the clustering of trajectories. In the paper we have shown the end-to-end process (see Section 5) from data harvesting to data analysis. In relation to performance, the Quickbundles algorithm has been proposed in this paper for trajectory clustering. We chose this instead of center-based clustering algorithms such as K-means due to the fact that spherical clusters are not appropriate for trajectory clustering. As a consequence, we conducted experiments to compare the behavior of spectral clustering and Quickbundles using different distance measures. The obtained results show that Quickbundles outperforms spectral clustering, with this behavior being coherent in both datasets: the Porto Kaggle and the CROSS-CPP data.

The silhouette metric was used to compare results. The silhouette coefficient metrics are known to be a ratio that depends on both the cohesion, that is to say the similarity inside each cluster, and on the separation, that is, the distance of points belonging to different clusters. From analyzing the obtained results, one could conclude that QB algorithm obtain clusters with a better ratio between cohesion and separation. Spectral clustering may yield clusters with better cohesion or separation metrics, but not both at the same time. However, it would be a future work to analyze these metrics in isolation.

In relation to experimentation, not all of the possible parameters for the compared algorithms have been explored. In particular, for spectral only, the number clusters has been modified in the different executions. For the case of QB, we ran experiments adjusting both the resampling factor and the neighboring threshold. In particular, for QB the experiments raised the fact that the resampling factor does not offer great changes in performance but the neighbouring threshold was shown to be determinant. This is in accordance with what has been already mentioned in the literature.

For experiments run with spectral clustering, we chose the number of clusters yielding the best results. Future work should include a deeper study by using different methods to construct the affinity matrix and fine-tuning the C and σ parameters, among others.

In relation with the distance measure, despite the fact that not high differences in silhouette coefficient were observed, results yielded slightly better results when using the geodesic WS84 in both datasets. Note that this distance improved performance even in the case of Porto, where the difference of distances is not very big.

Regarding time consumption, although experiments were not conducted to measure this issue, we could observe that the spectral algorithm took up to three times less the time to execute in comparison to Quickbundles. Although this should be further explored, results obtained are in accordance to literature: in [13], the authors establish that the time of execution of QB increases linearly with the number of trajectories .

Finally, the clustering was performed in batch mode. However, it is important to note that the CROSS-CPP analytic toolbox allows for data-streaming mode. In future works, it would be interesting to analyze the behavior of the algorithm for that setting.

In conclusion, in this paper, we have presented results of the integration of Quickbundles algorithm for the moving objects trajectories clustering in the CROSS-CPP platform. It has been successfully integrated and is available in the CROSS-CPP project. The reason to choose this algorithm was based on the results shown, in which so far the QB algorithms is shown to outperform the spectral clustering algorithm.

As was already mentioned, future work shall include experimenting with streaming clustering algorithms for real-time analysis. Furthermore, deeper studies to analyze the effect of the hyperparameters on the behavior of the algorithms as well as to measure the time consumption should be required to decide the best option depending on the amount and nature of the dataset.

Author Contributions: A.B.-H. and J.-I.C.-G. are responsible for design and implementation issues as well as drafting the manuscript. C.W. and A.Z. are the coordinators of the CROSS-CPP project and consequently, responsible of the coordination of all of the work performed. Their contribution in reviewing the paper has been crucial. E.M. and A.R.-G. are responsible for coordination of the modeling of the algorithms, validation steps and final writing of the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by CROSS-CPP EU grant number 780167.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, D.; Chen, D.; Song, B.; Guizani, N.; Yu, X.; Du, X. From IoT to 5G I-IoT: The Next Generation IoT-Based Intelligent Algorithms and 5G Technologies. *IEEE Commun. Mag.* **2018**, *56*, 114–120. [CrossRef]
2. Zimmer, K.; Kurban, H.; Jenne, M.; Keating, L.; Maull, P.; Dalkilic, M. Using Data Analytics to Optimize Public Transportation on a College Campus. In Proceedings of the 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), Turin, Italy, 1–4 October 2018; pp. 460–469.
3. Schneider, R.J.; Patten, R.S.; Toole, J.L. Case Study Analysis of Pedestrian and Bicycle Data Collection in U.S. Communities. *Transp. Res. Rec.* **2005**, *1939*, 77–90. [CrossRef]
4. Koshak, N.A.; Fouda, A. Analyzing Pedestrian Movement in Mataf Using GPS and GIS to Support Space Redesign. In *Design & Decision Support Systems in Architecture and Urban Planning*; University of Technology Eindhoven: Eindhoven, The Netherlands, 2008; p. 14.
5. Chen, C.; Zhang, D.; Samuel Castro, P.; Li, N.; Sun, L.; Li, S. Real-Time Detection of Anomalous Taxi Trajectories from GPS Traces. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*; Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering; Puiatti, A., Gu, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 104, pp. 63–74.
6. Castro, P.S.; Zhang, D.; Chen, C.; Li, S.; Pan, G. From taxi GPS traces to social and community dynamics: A survey. *ACM Comput. Surv.* **2013**, *46*, 1–34. [CrossRef]
7. Kozievitch, N.P.; Gadda, T.M.C.; Fonseca, K.V.O.; Rosa, M.O.; Gomes, L.C., Jr.; Abkar, M. Exploratory Analysis of Public Transportation Data in Curitiba. Anais do Seminário Integrado de Software e Hardware (SEMISH). *Soc. Bras. Comput. SBC* **2020**, 36–47. [CrossRef]
8. Azgomi, H.F.; Jamshidi, M. A Brief Survey on Smart Community and Smart Transportation. In Proceedings of the 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), Volos, Greece, 5–7 November 2018; pp. 932–939. [CrossRef]
9. Yuan, G.; Sun, P.; Zhao, J.; Li, D.; Wang, C. A review of moving object trajectory clustering algorithms. *Artif. Intell. Rev.* **2017**, *47*, 123–144. [CrossRef]
10. Moradi, M.M.; Pebesma, E.; Mateu, J. trajectories: Classes and Methods for Trajectory Data. *J. Stat. Softw.* **2018**. [CrossRef]
11. Graser, A. MovingPandas: Efficient Structures for Movement Data in Python. *GI_Forum* **2019**, *1*, 54–68. [CrossRef]
12. AL-Dohuki, S.; Kamw, F.; Zhao, Y.; Ye, X.; Yang, J.; Jamonnak, S. An Open Source TrajAnalytics Software for Modeling, Transformation and Visualization of Urban Trajectory Data. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 150–155.
13. Garyfallidis, E.; Brett, M.; Correia, M.M.; Williams, G.B.; Nimmo-Smith, I. QuickBundles, a Method for Tractography Simplification. *Front. Neurosci.* **2012**, *6*. [CrossRef] [PubMed]
14. Fu, Z.; Hu, W.; Tan, T. Similarity based vehicle trajectory clustering and anomaly detection. In Proceedings of the IEEE International Conference on Image Processing, Genova, Italy, 11–14 September 2005; p. II-602.
15. Karney, C.F.F. Algorithms for geodesics. *J. Geod.* **2013**, *87*, 43–55. [CrossRef]
16. Ester, M.; Kriegel, H.P.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Kdd*; AAAI: Menlo Park, CA, USA, 1996; Volume 96, pp. 226–231.
17. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137. [CrossRef]
18. Von Luxburg, U. A Tutorial on Spectral Clustering. *arXiv* **2007**, arXiv:0711.0189.
19. Zheng, Y. Trajectory Data Mining: An Overview. *ACM Trans. Intell. Syst. Technol.* **2015**, *6*, 1–41. [CrossRef]
20. Mazimpaka, J.D.; Timpf, S. Trajectory data mining: A review of methods and applications. *J. Spat. Inf. Sci.* **2016**, 61–99. [CrossRef]

21. Chen, J.; Wang, R.; Liu, L.; Song, J. Clustering of trajectories based on Hausdorff distance. In Proceedings of the 2011 International Conference on Electronics, Communications and Control (ICECC), Ningbo, China, 9–11 September 2011; pp. 1940–1944.
22. Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65. [CrossRef]