International Journal of Geo-Information

MDPI

Article

Extracting Stops from Noisy Trajectories: A Sequence Oriented Clustering Approach

Longgang Xiang [1,*], Meng Gao [1] and Tao Wu [2]

[1] State Key Laboratory of LIESMARS, Wuhan University, Wuhan 430079, China; gmshepard@whu.edu.cn
[2] School of Geosciences and Info-physics, Central South University, Changsha 410083, China; blackender@163.com
* Correspondence: geoxlg@whu.edu.cn; Tel.: +86-276-877-8751

Academic Editor: Wolfgang Kainz
Received: 26 January 2016; Accepted: 29 February 2016; Published: 9 March 2016

Abstract: Trajectories, representing the movements of objects in the real world, carry significant stop/move semantics. The detection of trajectory stops poses a critical problem in the study of moving objects and becomes even more challenging due to the inevitable noise recorded along with true data. To extract stops with a variety of shapes and sizes from single trajectories with noise, this paper presents a sequence oriented clustering approach, in which noise points within the sequence of a stop can be identified and classified as a part of the stop. In our method, two key concepts are first introduced: (1) a core sequence that defines sequence density based not only on proximity in space but also continuity in time as well as the duration over time; and (2) an Eps-reachability sequence that aggregates core sequences that overlap or meet over time. Then, three criteria are presented to merge Eps-reachability sequences interrupted by noise. Further, an algorithm, called SOC (Sequence Oriented Clustering), is developed to automatically extract stops from a single trajectory. In addition, a reachability graph is designed that visually illustrates the spatio-temporal clustering structure and levels of a trajectory. Finally, the proposed algorithm is evaluated against two baseline methods through extensive experiments based on real world trajectories, some with serious noise, and the results show that our approach is fairly effective in recognizing trajectory stops.

Keywords: trajectory stop; core sequence; reachability graph; sequence oriented clustering

1. Background

A trajectory represents the evolving locations of a moving object in geographical space over a given time interval. From the viewpoint of the computer world, a trajectory is a discrete record structure containing information about the evolving positions of a moving object in geographical space during a given time interval. Such a structure is composed of spatio-temporal points, each of which contains at least two components: an x-y position and a timestamp. The formal definition for a trajectory is given below.

Definition 1 (Trajectory). A trajectory $T = (tid, <p_0, p_1, \ldots, p_N>)$ is a two-tuple structure, where *tid* is a unique trajectory identifier. We have: (1) $p_i = (x_i, y_i, t_i)$, $i = 0, \ldots, N$, $x_i, y_i, t_i \in \mathrm{R}$, as a spatio-temporal point; and (2) $\forall 0 \leqslant i < j \leqslant N$, $t_i < t_j$.

Here, we present a trajectory point as p = (x, y, t), instead of p = (x, y, z, t), because: (1) the z-part, *i.e.*, elevation, is not always available in a trajectory dataset; (2) in our study and similar works, only latitude (the y-part), longitude (the x-part) and timestamp (the t-part) are required to compute space (using x and y) closeness and time proximity (using t); and (3) the changes of the z-part are very small, especially for trajectories recorded within cities, and therefore it is not necessary to apply the z-part on the computation of geographical distances.

Any subset of continuous spatio-temporal points in a trajectory is called a sequence. Due to the unavoidable errors in the measurement and sampling of GPS signals, spatio-temporal points will inevitably deviate from their real positions. Such deviation is what we called noise throughout this paper. When recording in an open area, this deviation is usually within 5–10 m, while acquiring in a semi-enclosed or totally closed area, spatio-temporal points may deviate away by tens or even hundreds of meters.

During a trajectory, the moving object does not always change its position; sometimes it purposely remains at some location for some time, during which the object's position stays fixed or varies only slightly. The stop–move model [1] of trajectory was developed from this idea. A trajectory, according to the stop–move model, is composed of a series of stops and moves, in which a move proceeds from one stop to the next. Here, the start and end points of a trajectory are considered as two special stops.

A stop implies that some important activity during a trajectory has been intentionally carried out at some location for a minimal amount of time, e.g., a 15-minute break for lunch at a fast-food restaurant, so a temporary stationary or slow moving, such as waiting for traffic lights and turning at corner, should not be considered as a stop. From the viewpoint of data, a stop is embodied by a sequence with a velocity of zero or a very low velocity, such as below three kilometers per hour.

A move connects two consecutive stops, where one or multiple means of locomotion may be used to move from one stop place to the next. As a reasonable moving behavior, the object should travel at least some minimum distance in space and continue for at least some minimum duration over time. A move also corresponds to a sequence, but with relatively high speeds.

Table 1 summarizes the stop-related terms used in this paper, in which separated stop and undetected stop are the two situations to cause true negative stops. When applying clustering algorithms to a trajectory, each output cluster corresponds to a sequence. However, it is unlikely for clusters to directly form stops because of the existence of noise. Therefore, a well-designed algorithm for stop extraction should be robust to noise as much as possible. In other words, it should avoid creating false positive stops and true negative stops, and therefore improve the ratio of effective stops.

Table 1. Meanings of stop-related terms.

Term	Meaning
Cluster	A set of points generated by some clustering algorithm
False positive stop	A sequence recognized as a stop, but not true
True negative stop	A sequence that forms a stop, but not recognized
Effective stop	A stop that is successfully detected
Separated stop	A stop, but detected as multiple separated sequences
Undetected stop	A stop, but failed to be detected

2. Introduction

Due to the rapid advances of portable GPS technology, more and more moving objects (e.g., people, cars, animals, *etc.*) now carry devices equipped with GPS chipsets. As a result, there has been an explosion in the collection of trajectory data in the past few years, which has given rise to a large number of applications that use trajectory data as input for various research purposes. Examples of such applications include personal mobility studies, city transportation management and animal behavior analyses. Existing studies on trajectory data have mainly focused on three topics: data management [2,3], querying techniques [4,5] and data mining [6,7], which aimed to extract knowledge by applying or refining traditional database methods directly to raw trajectory data. However, these works considered trajectories as just another type of spatio-temporal data and therefore failed to make use of the rich potential of geographical semantics.

The stop–move model, introduced by Spaccapietra *et al.* [1], views a trajectory as a sequence of stop/move objects, which then can be annotated with important geographical semantics. Figure 1 demonstrates an example trajectory in the form of a stop–move model. One can see that a stop implies

that the moving object stays at some location for some time to carry out some activity, while a move connects two consecutive stops by some means of locomotion. The stop–move model supports more powerful trajectory analyses [8] than do the raw point-based models [2,9], which often represent a trajectory as a geometry of line; thus, an important task here is to find the stops in trajectories effectively.

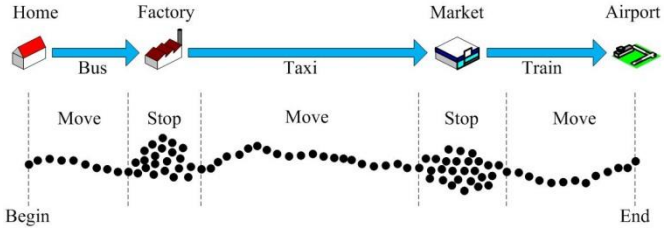

Figure 1. An example trajectory in the form of a stop–move model.

Under ideal conditions, the positioning accuracy of GPS devices is between five and ten meters [10]. In reality, however, due to reflection/blocking of GPS signals, acquired positions may jump away from their actual locations by tens or even hundreds of meters. On the other hand, a trajectory stop can either be an indoor stay or occur at any outdoor site, during which the GPS device may be intentionally powered off or keep recording. In addition, a trajectory may be sampled irregularly and could be composed of multiple segments that include different modes of locomotion. As a result, trajectory stops may exhibit diverse characteristics: a large number of points scattered around some location (usually taking place indoors), a set of points distributed within a small area (often occurring outdoors), a single point with a very large time interval (caused by turning the device off or absence of signal), or even a mixture of these.

To extract stops with a variety of shapes and sizes from single trajectories containing noise, this paper presents a sequence oriented clustering approach, in which noise points within the sequence of a stop can be identified and classified as a part of the stop. Our method tries to recognize trajectory sequences with high density as stops, where the density function is defined by both spatial distance and temporal duration. It introduces several novel concepts to cluster the trajectory points and then presents three criteria to merge noise-interrupted trajectory sequences. Accordingly, an algorithm called SOC (Sequence Oriented Clustering) is developed; this was inspired by two well-known density-based clustering algorithms, *i.e.*, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [11] and OPTICS (Ordering Points To Identify the Clustering Structure) [12]. In addition, a visualization tool based on reachability distance is designed to illustrate the spatio-temporal clustering structure and levels of a trajectory. Finally, extensive experiments are conducted with real world trajectories. An evaluation of the results shows that our approach is fairly effective compared with two baseline methods in recognizing trajectory stops, especially for noise-prone trajectories.

The rest of this paper is organized as follows. Section 2 outlines the current literature related to the detection of trajectory stops. In Section 3, the underlying main ideas of core sequence and Eps-reachability sequence are developed. Section 4 discusses two sequence merging techniques. Section 5 presents an algorithm to automatically form trajectory stops and describes the reachability graph, which graphically represents trajectory stops. The detecting algorithm is evaluated in Section 6. A discussion and conclusion appear in the last section.

3. Related Works

This paper was inspired by two well-known algorithms: DBSCAN and OPTICS. DBSCAN is a density-based clustering algorithm that can be applied to discover arbitrarily shaped clusters, while OPTICS can be considered a generalization of DBSCAN for multiple ranges, *i.e.*, it creates an augmented ordering of the input points representing their hierarchical clustering structure. These

two well-known clustering algorithms have been used in many applications and studied extensively. For example, ST-DBSCAN [13] extends the DBSCAN algorithm for clustering spatio-temporal data.

For DBSCAN and OPTICS, the definition of a cluster of spatial points is based on the notion of density reachability. Basically, point q is directly density-reachable from point p if p's neighborhood, defined by a given radius (*Eps*), contains at least a minimum number (*MinPts*) of other points and at the same time q lies within that defined neighborhood. Here, p is called a core point. Point q is called density-reachable from point p if there is a chain of points $p_1, p_2, \ldots, p_n$, where $p_1 = p$ and $p_n = q$ such that p_{i+1} is directly density-reachable from p_i.

Both DBSCAN and OPTICS process each input point p once and perform one neighborhood query to test whether p is a core point or not. If p is a core point, a new cluster is created, which will be expanded by recursively adding points that are density-reachable from those points already lying in the cluster. It is not hard to see that the two algorithms require two parameters: *Eps*, which describes the maximum radius to consider, and *MinPts*, which describes the minimum number of points required to form a cluster.

The work of this paper is directly related to two categories of studies oriented toward trajectory data. One category uses density-based clustering to mine important places using multiple trajectories and the other extracts interesting sequences from single trajectories. Our work adopts the idea of clustering, but belongs to the second category.

Generally speaking, movement parameters such as speed and direction present a totally different profile as the status of a movement switches from move to stop. Accordingly, it is very natural to apply some criteria to segment a trajectory to identify the stops [14,15]. Two examples of such criteria are: (1) a velocity of zero (or very low velocity) for at least some defined duration; and (2) the absence of GPS data for longer than a defined duration. However, those criteria work well only under ideal conditions, *i.e.*, those without noise. Rocha *et al.* made use of directional changes to detect stops [16], but that technique works correctly only under certain circumstances, such as when analyzing the trajectories of fishing boats. In [17], contextual geographical information is integrated to generate stops by testing the duration of a trajectory sequence within application-predefined regions of interest.

Stop extraction can to a certain degree be seen as a problem of trajectory segmentation, in which a trajectory is divided into homogenous pieces. Buchin *et al.* developed an algorithm framework to segment a trajectory based on advanced spatio-temporal criteria [18]; Junior *et al.* explored the principle of Minimum Description Length (MDL) and designed an unsupervised iterative procedure for segmenting a trajectory [19]. Still, these segmentation methods give little attention to the existence of noise in a trajectory, and therefore not appropriate for extracting stops.

Clustering is a basic and popular approach in the field of data mining and is also an indispensable tool for exploring information embedded in trajectory data. Specifically, with density-based clustering, two types of studies have been conducted in the literature: one finds significant places [20,21] such as railway stations, where many trajectories leave an abundance of points; the other derives advanced trajectory patterns [22,23] such as flocking behavior, in which a group of trajectories stay close together for a given duration.

Regarding the extraction of trajectory stops, CB-SMoT (Clustering-Based Stop and Move of Trajectories) [24], T-OPTICS (Trajectory-OPTICS) [25] and TrajDBSCAN (Trajectory DBSCAN) [26] are three clustering-based methods available in the literature. To the best of our knowledge, they are also the most closely related and comparable methods to ours. CB-SMoT extends the main idea of DBSCAN in which a core point is computed by testing neighboring points with the average speed. The main problem of CB-SMoT lies in that it is difficult to discover stops when only a few points exceed the speed limit. T-OPTICS works by following the main framework of OPTICS but for the characteristics of a stop on the time dimension, it captures only proximity and does not consider duration. TrajDBSCAN first studies the discovery of stops and then investigates on how to compute shared stops and build stop hierarchy. Compared to CB-SMoT and T-OPTICS, TrajDBSCAN uses geographical distance, instead of travel distance, to develop the concept of core point, and therefore

gets less sensitive to speed. However, when facing trajectories with big noise, it is still difficult for TrajDBSCAN to locate core points and derive stops. Moreover, the three clustering-based methods cannot identify single points with large time intervals as stops (see Figure 2b for an example), and in addition, they fail to provide any mechanism for merging noise-interrupted sequences into stops (see Figure 3b for an example).

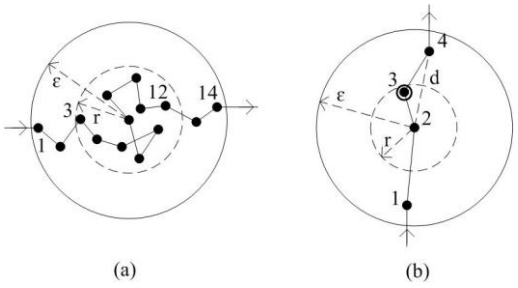

Figure 2. Two typical core sequences: (**a**) a core-sequence generated by continuously recording; and (**b**) a core-sequence caused by pausing recording.

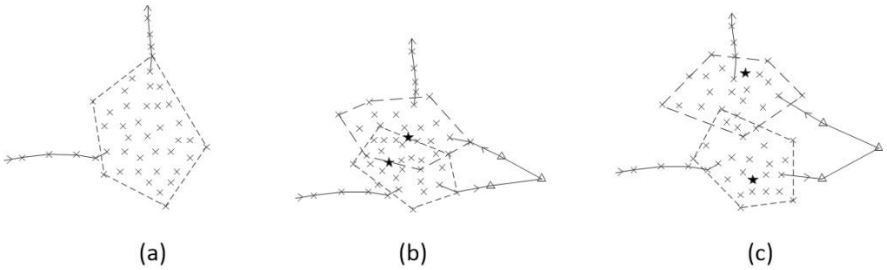

Figure 3. Sequence merging: (**a**) a well clustered stop sequence; (**b**) a stop sequence merged by Criterion 1; and (**c**) a stop sequence merged by Criterion 2.

4. Eps-Reachability Sequence

A stop during a trajectory, practically speaking, is an action such as an indoor stay, an outdoor stop, or wandering around within a small area. Therefore, the inherent characteristics for a trajectory stop are two-fold: (1) points are located closely in space; and (2) the stop lasts for some minimum time duration. To support this point of view for trajectory stops, the concept of core sequence is introduced first. A core sequence is defined as a long stay within an area with a small radius.

Definition 2 (Eps-sequence). Let T be a trajectory and p be a point of T. The Eps-sequence of p, marked as Seq(p, *Eps*) is the maximum sequence in T that satisfies: (1) $p \in$ Seq(p, *Eps*); and (2) $\forall q \in$ Seq(p, *Eps*) where distance(p, q) $\leqslant$ *Eps*. Here, *Eps* is a given radius.

In the above definition, the distance from point q to point p, *i.e.*, distance(p, q), is a geographical distance, which is calculated throughout this paper by applying the Euclidean distance. Based on Definition 2, the concepts of core sequence and core point can be derived, as shown below.

Definition 3 (Core sequence). Let S be a sequence of a trajectory. S is called a core sequence if it satisfies the following criteria: (1) $\exists p \in S$, where S is an Eps-sequence of p; and (2) the temporal span of S is not shorter than *Tau*. Here, *Tau* is a given temporal duration.

Definition 4 (Core point). Let p be a point of a trajectory. Point p is called a core point with respect to ε and τ if Seq(p, *Eps*) is a core sequence.

Definition 3 says that a core sequence is made up of a set of continuous points that stay within a defined circular area for at least a given amount of time. The ratio 2**Eps*/*Tau*, which indicates the maximum speed limit for crossing the circle with a radius of ε (*i.e.*, moving exactly along a diameter), should be relatively small to prevent a moving sequence from being mistakenly detected as a core

sequence. Therefore, a core sequence clusters points not only in space but also over time, which means that points in a core sequence are not only spatially close but also temporally proximal. Note that *Tau*, a time threshold to define core sequence, should be set significantly larger than the minimum sampling interval during a trajectory.

Figure 2 depicts two typical core sequence structures: Figure 2a shows that the GPS device is still recording positions during a stop event, while in Figure 2b, the GPS device was switched off when the stop occurred (specifically, the GPS device was switched off after capturing point 3). Based on the idea of core sequence, we further develop the concepts of core distance and reachability distance.

Definition 5 (Core distance). Let p be a point of a trajectory. The core distance of point p with respect to *Eps* and *Tau* is defined as $\min\{r \mid r \leqslant Eps \cap \text{Seq}(p, r)$ is a core sequence$\}$ if $\text{Seq}(p, Eps)$ is a core sequence; it is UNDEFINED otherwise. Here, UNDEFINED is a predefined value greater than *Eps*.

The core distance represents the spatial closeness of a trajectory point to its temporal neighbors. In Figure 2, the core distance in either case is smaller than the maximum distance between the centered point and other points. The core distance of point 3 in Figure 2b is zero because it is a core point with a large time interval.

Definition 6 (Reachability distance). Let p_i be a point of a trajectory. The reachability distance of p_i with respect to *Eps* and *Tau* is defined as $\max\{$core distance(p_k), distance$(p_k, p_i)\}$ if $k = \max\{j \mid j \leqslant i \cap p_i \in \text{Seq}(p_j, Eps) \cap \text{Seq}(p_j, Eps)$ is a core sequence$\}$; it is UNDEFINED otherwise.

Obviously, if point p is not included in any core sequence defined either on p or the points preceding p, the reachability distance of p will take a predefined value greater than *Eps*. The reachability distance of a trajectory point implies a spatio-temporal clustering level with respect to core sequence. In Figure 2b, the reachability distance for point 2 is r, while for point 4, it is the distance between point 4 and point 3 because point 3 is also a core point that is temporally closer to point 4. After the computation of reachability distance for all trajectory points, the Eps-reachability sequence can be derived, which is composed of those continuous points with a reachability distance not larger than *Eps*.

Definition 7 (Eps-reachability sequence). Let $S = \{p_s, p_{s+1}, \ldots, p_{e-1}, p_e\}$ be a sequence of trajectory T and $0 \leqslant s \leqslant e < |T|$. S is called an Eps-reachability sequence with respect to *Eps* and *Tau* if it satisfies these conditions: (1) the reachability distance for any point in S is smaller than *Eps*, or equal to it; and (2) the reachability distances for both p_{s-1} and p_{e+1} are greater than *Eps*.

According to Definition 6, a trajectory can be divided into a series of Eps-reachability sequences delimited by the points with reachability distances greater than *Eps*. One can see that the length of an Eps-reachability sequence may be as short as one, *i.e.*, an Eps-reachability sequence formed by a single point. This often occurs when a point has a relatively large time interval, and at the same time, the distance from this point to the next point is relatively small. Such a point is called a "big point" in this paper and could be caused by turning the GPS device off or by signal absence. Big-point handling is one special aspect that should be addressed when extracting trajectory stops.

If trajectory points always recorded their actual positions in the geographical space, each Eps-reachability sequence in a trajectory could be considered as a stop. However, due to recording noise, a stop may be interrupted by noise points; therefore, consecutive Eps-reachability sequences may need to be merged to generate stops.

5. From Eps-Reachability Sequence to Stop

Due to the GPS signal measurement and sampling errors in mobile devices, recorded position deviations are not rare. Usually, trajectory data are imprecise and carry noise even after pre-processing procedures, e.g., data cleaning and data smoothing [27]. When processing such error-prone trajectory data, a stop may be mistakenly detected when multiple Eps-reachability sequences are separated by noise points.

Criterion 1: Let S_1 and S_2 be two consecutive Eps-reachability sequences of a trajectory. If distance(S_1.center, S_2.center) $\leqslant$ 2**Eps* and interval(S_1.last, S_2.first)<*MinMov*, then S_1 and S_2 should be merged into one sequence. Here, *MinMov* is the minimum movement duration.

Criterion 1 states that if two consecutive Eps-reachability sequences are distributed closely in space and separated by a too short duration to be considered a reasonable moving action, they should be merged into one sequence. *MinMov* is the minimum duration that a normal moving behavior should last. Note that in Criterion 1, the center of a sequence is not the geometric center but the spatio-temporal center. Given a sequence S = $\{p_s, p_{s+1}, \ldots, p_{e-1}, p_e\}$, its spatio-temporal center is calculated by weighting each involved point with its time interval, as shown in Formula (1).

$$\begin{aligned} \text{S.center}.x &= \frac{\sum_{i=s}^{e} p_i.x * (p_{i+1}.t_{i+1} - p_i.t_i)}{\sum_{i=s}^{e} (p_{i+1}.t_{i+1} - p_i.t_i)} \\ \text{S.center}.y &= \frac{\sum_{i=s}^{e} p_i.y * (p_{i+1}.t_{i+1} - p_i.t_i)}{\sum_{i=s}^{e} (p_{i+1}.t_{i+1} - p_i.t_i)} \end{aligned} \tag{1}$$

Criterion 2: Let S_1 and S_2 be two consecutive Eps-reachability sequences of a trajectory. If the convex hulls of S_1 and S_2 are overlapped and interval (S_1.last, S_2.first)<*MinMov*, then S_1 and S_2 should be merged into one sequence.

Criterion 2, compared to Criterion 1, adopts a predicate based on spatial shape instead of spatial distance. Accordingly, it does not need to specify a distance threshold for sequence merging. As a result, Criterion 2 is more flexible and powerful than Criterion 1 on merging Eps-reachability sequences. Figure 3 illustrates an example of sequence-merging with the above two criteria, in which an Eps-reachability sequence's convex hull is presented as a dashed polygon and its spatio-temporal center is drawn as a star. The noise points in Figure 3 are marked as triangles. One can see that either Criterion 1 or Criterion 2 can be applied in the middle case, because for the two separated Eps-reachability sequences, their spatio-temporal centers are close, and besides, their convex hulls are overlapped. However for the right case, only Criterion 2 can be applied as the distance between the two spatio-temporal centers is relatively big.

Given two consecutive Eps-reachability sequences, if their spatio-temporal centers are mapped to the same addressable location, it is very likely that they represent the same stop. To find the addressable location, one effective idea is to apply a reverse geocoder against the spatio-temporal center of the Eps-reachability sequence. Therefore, we have another criterion for merging Eps-reachability sequences.

Criterion 3 Let S_1 and S_2 be two consecutive Eps-reachability sequences of a trajectory. If S_1.center and S_2.center are reverse geocoded to a same addressable location and interval(S_1.last, S_2.first)<*MinMov*, then S_1 and S_2 should be merged into one sequence.

Note that Criterion 3 applies only under the condition that the reverse geocoded location is addressable. Taking Google Map APIs [28] as an example, the type of returned addresses should be either "precise" or "street address." According to the above three criteria, an Eps-reachability sequence could grow continuously until no new Eps-reachability sequences can be drawn and joined. Such a fully grown sequence is called a Ful-reachability sequence. A Ful-reachability sequence, either merged from multiple Eps-reachability sequences or formed by a single Eps-reachability sequence, always starts from a core point but does not always end with a core point.

For a Ful-reachability sequence, the end points may unfortunately be those that are actually leading away from the stop site but are still reachable from the last core point within the sequence. Therefore, a Ful-reachability sequence should be post-pruned, ensuring it ends with a core point. After merging and post-pruning, the final stops can be generated. Therefore, we have arrived at the point where we can give a formal definition for a trajectory stop.

Definition 8 (Trajectory stop). Let S be a Ful-reachability sequence in a trajectory. A trajectory stop is the prefix sequence of S that ends with the last core point in S.

6. The Extraction and Visualization of Trajectory Stops

This section first describes an algorithm designed to extract trajectory stops and then discusses the reachability graph for visualizing the inner structure of trajectory clustering.

6.1. The SOC Algorithm

Based on the earlier discussions in this paper, a novel algorithm, called SOC, is developed to extract trajectory stops; its pseudo code is presented in Algorithm 1. SOC first sets all points to a reachability distance value of UNDEFINED and then computes the reachability distance by scanning the input trajectory points in chronological order. Next, SOC extracts Eps-reachability sequences according to Definition 7, *i.e.*, continuous points whose reachability distance is not bigger than ε are detected as Eps-reachability sequences. After that, neighboring Eps-reachability sequences, separated by noise points but close in both space and time, are merged into Ful-reachability sequences according to Criteria 1 and 2. Note that if two sequences are merged, the reachability distances for the points between the two sequences will be updated to value *Eps*, even though they are not covered by any core sequence. Next, a pruning procedure is applied to ensure that each Ful-reachability sequence ends with a core point. Finally, if a recognized stop is refined as a false positive stop, it will be removed from the results.

Algorithm SOC (***TrajPt, Eps, Tau, MinStp, MinMov***): ***Stops***
Inputs:

1. *TrajPt* = $\{p_1, p_2, \ldots, p_n\}$, the spatio-temporal point set of a trajectory
2. *Eps*, the geographical distance to define a core sequence
3. *Tau*, the time duration that defines a core sequence
4. *MinStp*, the minimum duration for a stop
5. *MinMov*, the minimum duration for a move

Output:

1. *Stops* = $\{s_1, s_2, \ldots, s_m\}$, the set of trajectory stops, each of which is a sequence in *TrajPt*

Method:

1. Initialize the reachability distance of each point in *TrajPt* to UNDEFINED;
2. **FOR** i = 1 to $|TrajPts|$
   (a) ComputeDistance(*TrajPt*, p_i, *Eps*, *Tau*);
3. **END FOR**
4. *Seqs* = Extract_Sequence(*TrajPt*, *Eps*);//according to Definition 7
5. *Stops* = Merge_Sequence(*Seqs*, *Eps*, *MinMov*);//according to Criterion 1 and 2
6. PruneStops(*Stops*, *Eps*, *Tau*);//ensure every stop sequence ends with a core point
7. FalsePositiveRemove(*Stops*);//remove false positive stops

Function ComputeDistance(*TrajPt*: in out, *CurPt*: in, *Eps*: in, *Tau*: in)
Method:

1. *Seq* = GetEpsSequence(*TrajPt*, *CurPt*, *Eps*);//according to Definition 2
2. **IF** *Seq* is not a core sequence w.r.t. *Tau*, **THEN RETURN END IF**
3. r = GetCoreDistance(seq, *Tau*);//according to Definition 5
4. **FOR** each point q in *Seq*
   (a) **IF** q precedes *CurPt* **THEN CONTINUE END IF**
   (b) q.reachability distance = max{r, distance(*CurPt*, q)};
5. **END FOR**

Compared to DBSCAN and its variation OPTICS, although SOC also benefits from the idea of density clustering, it orients to sequence clustering, considering not only closeness in space but also proximity in time. Another point of difference is that in SOC, noise points within a trajectory stop can be detected and accepted as a part of the stop instead of being marked as noise. SOC has the

same computational complexity as DBSCAN and OPTICS, *i.e.*, O(n*log n). However, for SOC, after a core sequence has been detected, only the points belonging to that core sequence will be checked further. As a result, SOC usually requires less I/O and computation than DBSCAN and OPTICS. One experiment, using a real trajectory of approximately 5000 points, showed that the runtime is 19 s for SOC but 33 s for OPTICS.

6.2. False Positive Elimination

When capturing core sequence and forming Eps-sequences, SOC adopts an idea similar to DBSCAN; therefore, it is capable of generating trajectory stops of arbitrary shapes. Consequently, a line-shaped slow movement, e.g., driving in heavy traffic during rush hours, may be detected as a stop by SOC. To remove this type of false positives, two indexes, straightness and centered-distance, are jointly explored. Given a sequence S = $\{p_s, p_{s+1}, \ldots, p_{e-1}, p_e\}$, the measurements of straightness and centered-distance can be calculated with Formula (2). Here, centered-distance is necessary because for a sequence, if the middle points consume most of the duration but contribute little to the travel distance (e.g., the middle points are distributed over a very small area or consist only of a big point), straightness will have a large value (*i.e.*, close to 1), while centered-distance will have a small value (*i.e.*, significantly smaller than *Eps*). Hence, if both the straightness and centered-distance of a clustered sequence exceed some specified thresholds, the sequence will be identified as a false positive stop and should be filtered out.

$$\text{S.straightness} = \frac{\text{distance}\,(p_s,\ p_e)}{\sum_{i=s}^{e-1}\text{distance}\,(p_i,\ p_{i+1})}$$

$$\text{S.centered} - \text{distance} = \frac{\sum_{i=s}^{e}\text{distance}\,(p_i, \text{S.center}) * (p_{i+1}.t_{i+1} - p_i.t_i)}{\sum_{i=s}^{e}(p_{i+1}.t_{i+1} - p_i.t_i)}$$

When a sequence corresponds to the case of a slow U-turn movement, it is likely to be recognized as a stop because making the turn consumes a significant amount of time. To identify this type of false positives, the concept of "heading direction" is explored. Specifically, when the heading direction for the anterior and posterior of a clustered sequence differs greatly, the sequence should not be recognized as a stop.

In reality, a stop of very short duration may be too trivial to be considered by applications. Accordingly, clustered sequences with a duration shorter than some threshold should not be recognized as a stop. The above three rules are applied together at the last step of the SOC algorithm to recognize and filter out false positive stops.

6.3. Reachability Graph

Similarly to OPTICS, the spatio-temporal clustering of trajectory points can be represented and understood graphically using a reachability graph, where the reachability distance values are plotted for each trajectory point. Note that the points in a reachability graph strictly follow the sequence of points appearing in the input trajectory.

According to SOC, three levels of reachability distance will be generated: values smaller than *Eps* for normal stop points, *Eps* for merged noise points, and UNDEFINED for move points. UNDEFINED can be any predefined value greater than *Eps*, but for ease of illustration, a value slightly bigger than *Eps*, such as 1.2**Eps* is recommended. For a clustered sequence, a smaller reachability distance implies a higher clustering level, *i.e.*, a long-duration sequence confined within a small area, while a bigger value means a relatively low clustering level.

For the reachability graph of a trajectory, the clustering structure as a whole and the clustering levels of individual points will be influenced by the ratio of *Eps*/*Tau*. Roughly speaking, a smaller *Eps* or a bigger *Tau* may prevent more points from being covered by the core sequence; therefore, a stop sequence may shrink or even disappear altogether. In contrast, a bigger *Eps* or a smaller *Tau* enables a

core sequence to include more points; therefore, a stop sequence may expand or even swallow points in neighboring moving sequences. Setting the parameter values for *Eps* and *Tau* will be discussed further in the experimental section.

Figure 4 depicts a reachability graph with *Eps* = 20 m and *Tau* = 50 s for a portion of a trip trajectory collected in Beijing, China. The top part of the figure shows the mapped trajectory. There are four Eps-reachability sequences in total (denoted as I–IV), where the second has a much higher clustering level than others. Note that point 39 is a big point with a time interval of 320 s, which accounts for the fact that point 39 and its preceding point are included in the first Eps-reachability sequence. However, when the core-sequence parameters are adjusted as *Eps* = 30 m and *Tau* = 75 s, the Eps-reachability sequences III and IV will be merged into one Eps-reachable sequence because with a bigger *Eps*, the points between III and IV are reachable by the core points in III.

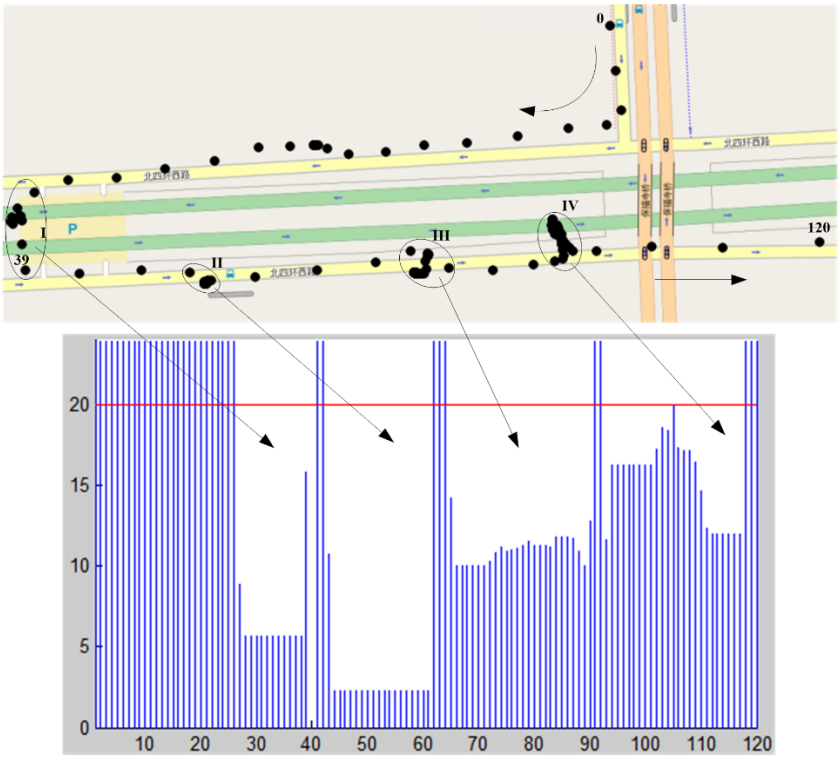

Figure 4. An example reachability graph with respect to *Eps* = 20m and *Tau* = 50s.

7. Experimental Evaluation

In this section, four real trajectory datasets from different sources were used to test the performance of SOC, which was evaluated from two main perspectives: result correctness and parameter sensitivity. The details of the four trajectory datasets are presented in Table 2, in which the "Labeled stops" column gives the number of manually labeled stops in each dataset.

Table 2. Four experimental trajectory datasets.

Dataset No.	Trajectory Amount	Sampling Rate	Average Duration	Average Distance	Labeled Stops
1	50	3 s	27 m	479 m	121
2	10	irregular	53 m	6.7 km	43
3	10	5 s	9 h 34 m	41.5 km	75
4	10	1 s	3 h 15 m	125.1 km	53

We acquired the first two datasets ourselves in Wuhan, China in 2014 using mobile phones and professional GPS recorders, while the last two datasets are, small extracted portions of two well-known free internet GPS archives, OpenStreetMap [29] and GeoLife [30]. Dataset 1 was collected by an application running on Android mobile phones equipped with GPS chipsets. The movements captured in this dataset mainly took place on a campus. Dataset 2 was collected by two types of professional GPS recorders: a Garmin Forerunner and a Holux M-1200E. The movements captured in this dataset recorded commuter routes during rush hours. The third dataset consists of volunteers' daily movements in Beijing, China in 2009, and the last dataset selected was uploaded in 2010 by some Japanese volunteers and mainly recorded hiking routes in Japan.

Dataset 1 was directly formatted in the form of stop–move sequences; therefore, the stop information for each trajectory is very easy to obtain. For the other three datasets, a visual approach based on QGIS (Quantum Geographical Information System) [31] was applied to manually check and mark trajectory stops. Specifically, stops lasting longer than three minutes during each trajectory were carefully labeled. In addition, the Android application used to collecting Dataset 1 is designed to stop logging when it enters an indoor space, while the GPS recorders used in collecting Dataset 2 continued to log points once started, even when indoors.

Dataset 1 was sampled on a campus; dataset 4 was acquired in some rural area. Consequently, the noise in the two datasets is relatively small and the average deviation is about 10 m. However, the other two datasets occurred in big cities with less-than-ideal GPS signals because of signal reflection/blocking, and consequently, the noise gets pervasive and serious. For example, a trajectory from dataset 2 entered a building with some samples jumping away from their real positions by several hundred meters; a trajectory from dataset 3 passed under a viaduct with some samples deviating more than 50 meters.

7.1. Experimental Setting

As declared in Algorithm 1, the SOC algorithm depends on three key parameters, which are summarized in Table 3. Among them, *Eps* and *Tau* are used to generate Eps-reachability sequences, while *MinMov* is used to merge Eps-reachability sequences. Moreover, four additional threshold values (straightness, centered-distance, direction difference and minimum stop duration) are required to filter three types of false positive stops. After testing with example trajectories, the threshold value parameters were set to default values of 0.5, 2**Eps* meters, 90 degree and three minutes, respectively. Unless explicitly specified, all parameters assume default values for all experiments.

Table 3. Three key parameters of SOC.

Parameter	Description	Default Value
Eps	The minimal radius to define a core sequence	30 m
Tau	The minimal duration to define a core sequence	75 s
MinMov	The minimal duration for a normal move	180 s

With the default setting, the reachability graph of an outdoor trajectory from Dataset 2 is illustrated in Figure 5 (the left part is the mapped trajectory). We can clearly see that there are four stops during the trip, which conforms closely to reality. We also observe that the first and last stops have small reachability distances, which means the points are well spatio-temporally clustered. Actually, the two stops correspond to simply standing still outdoors.

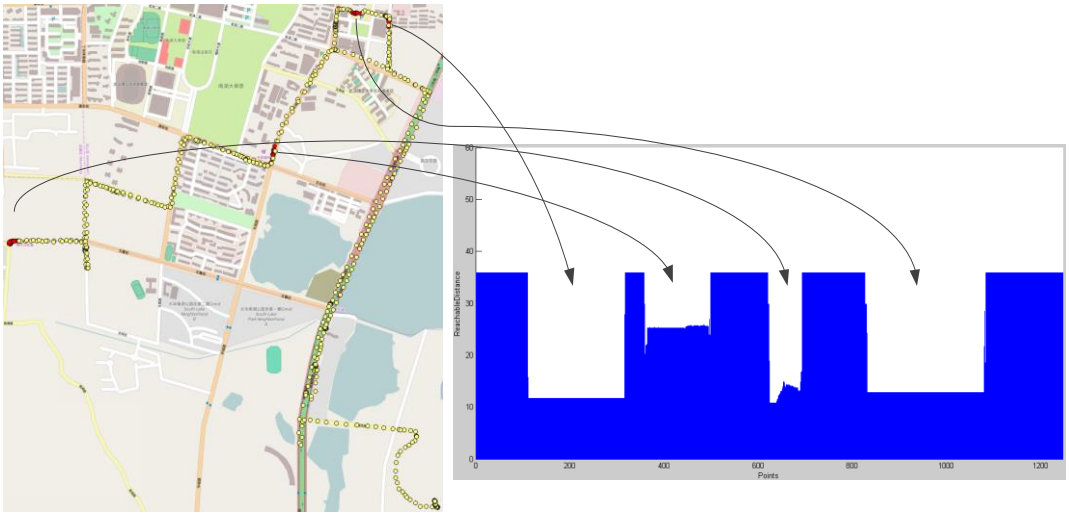

Figure 5. A trajectory from Dataset 2 and its reachability graph.

During the sequence-merging procedure, reverse geocoding is invoked to obtain a street address. Practically, we chose to use two different reverse geocoding services, namely, Google Maps and Baidu Maps [32]. Because Baidu Maps has more abundant address information in China, the algorithm calls the reverse geocoding service of Baidu Maps when the spatio-temporal center of a sequence falls into China; otherwise, it calls the Google Map service. As the returned addresses are formatted as string structures, SOC simply explores standard string comparison to check if sequences share a same address or not. Note that the real latitude and longitude for the addresses are encrypted in the call to the Baidu Maps service. Accordingly, before performing reverse geocoding using Baidu Maps, the spatio-temporal centers, which were initially coded using the WGS (World geodetic system) 84 specification, must be translated to the coordinate system used by Baidu Maps. This can also be performed by calling an external API of Baidu Map.

In our current implementation, we could accept trajectory data of TXT or GPX format, which will be read into memory before processing. The resulting stops are outputted as a text file and visualized with MATLAB. Since a single trajectory is usually small in size, such pre-processing will not cause trouble. For example, the storage size of a 24-hour trajectory with a sampling rate of 1 s is smaller than seven megabytes (here, 10 information fields such as latitude, longitude and timestamp are assumed to be recorded). Even facing a very large trajectory that cannot be fit into memory, only the parts of point reading and neighborhood locating are needed to be updated to work on external memory like file system or DB system.

7.2. Effectiveness Evaluation

To validate the effectiveness of SOC, two baseline methods were used here: speed-testing and CB-SMoT. In the speed-testing method, 3 km/h was selected as the limit for testing stop points; in the CB-SMoT method, the same *Eps* and *Tau* were used to generate clusters. The two baseline methods, similar to SOC, also prune stops that last no longer than three minutes. The results are shown in Table 4, in which "SOC without merging" means that the sequence merging function was disabled for that test. For a given trajectory sequence S, there are four conditions: (1) if S is a labeled stop and was detected as a single stop, it is counted as an effective stop; (2) if S was not labeled but detected as a stop, it is counted as a false positive stop; (3) if S was labeled but detected as multiple stops, it is counted as a separated stop; and (4) if S was labeled but not detected as a stop, it is counted as an undetected stop. Note that we have a formula: "Labeled stops" = "Effective stops" + "Separated stops" +"Undetected stops". The last column denotes the average ratio of the number of a measure to the number of labeled stops.

We can see that SOC performs better than the two baseline methods in all cases in recognizing effective stops. Even without the merging step, the average recognition accuracy of effective stops for SOC is 83.5%, while it is 75.4% and 65.5% for CB-SMoT and speed-testing, respectively. Figure 6a demonstrates a common scenario—an indoor stop that SOC recognized successfully but that speed-testing and CB-SMoT failed to recognize. This occurred because SOC uses geographical distance instead of travel distance to define core points and is therefore less sensitive to an object's speed and more robust to noise. After enabling the merging function in SOC, a total of 19 separated stops were successfully recognized as effective stops; accordingly, the average recognition ratio of effective stops improved to 91.3%. We also observe that SOC overwhelms the two baseline methods on the three other measures: false positive stops, separated stops and undetected stops. For example, the average ratio of undetected stops for speed-testing and CB-SMoT are 14.6% and 10.0%, respectively, but only 4.5% for SOC.

Table 4. SOC *versus* two baseline methods on detecting stops.

Methods	Measures	Datasets				Average Ratio
		#1	#2	#3	#4	
Speed-testing	Effective stops	91	24	43	39	65.5%
	False positive stops	7	6	9	5	10.2%
	Separated stops	13	12	21	7	20.0%
	Undetected stops	17	7	11	7	14.6%
CB-SMoT	Effective stops	103	27	53	44	75.4%
	False positive stops	15	11	17	9	15.7%
	Separated stops	5	10	15	6	14.8%
	Undetected stops	13	6	7	3	10.0%
SOC without merging	Effective stops	112	31	59	48	83.5%
	False positive stops	0	3	5	2	3.8%
	Separated stops	0	10	13	4	12.0%
	Undetected stops	9	2	3	1	4.5%
SOC	Effective stops	112	37	69	51	91.3%
	Separated stops	0	4	3	1	3.8%

Because the speed-testing method considers only speed when generating stops, it is very sensitive to noise. Hence, for this straightforward method, a stop with many points is highly likely to be detected as a separated stop when one or a few intermediate points exceed the speed threshold due to noise. For the same reason, the speed-testing method reduces the possibility of detecting slow line-shaped or U-turn movements as false positive stops. As a result, speed-testing has a higher ratio of separated stops (20.0%) but a relatively lower ratio of false positive stops (10.2%) compared with CB-SMoT (15.7% and 14.7%, respectively).

Because Datasets 2 and 3 were sampled in big cities with less-than-ideal GPS signals due to multi-path signal reflection or signal blocking, all three methods perform relatively poorly, detecting many single labeled stops as multiple small stops. For Dataset 1, an interesting point is that SOC detected no false positive or separated stops because the GPS signal is relatively good within the campus environment. However, in this dataset, there are nine sequences that were annotated by students as stops but were not detected by SOC. This is because the nine stops, though successfully detected in previous steps, were eventually filtered out as false positive stops due to their short duration (*i.e.*, less than three minutes).

Staying indoors but continuing to log point data is likely to cause separated stops. Figure 6b shows an example of this condition in which only two small stops (red and green points) were identified. In reality, this corresponds to an indoor stay of about three hours, in which the points scatter around a relatively large area and some points jump more than one kilometer away. Because the two small stops represent significant interruptions in both space and time, they failed to be merged by the two merging criteria. However, the two sequences shown in Figure 6c,d were successfully merged as two single stops by Criterion 1 and Criterion 2, respectively. For the latter case, the separated sequences were

reverse geocoded to the same addressable location. Generally speaking, the longer a GPS device stays indoors, the more widely logged positions tend to scatter, which means there is a higher likelihood that the points will be analyzed as separated stops.

In addition to true negative stops (*i.e.*, separated stops and undetected stops), SOC will inevitably introduce false positive stops, even though three rules have been explored and applied. Consider Figure 6e, which corresponds to a U-turn while driving under a viaduct (the lines shown in light green). Because the movement was slow due to congestion and the position precision was low due to a poor GPS signal, a sequence (the red points) was detected but failed to be filtered, leading to a false positive stop. Unfortunately, when a very slow turn occurs in a location with a poor GPS signal, the recorded sequence is very likely to be recognized as a false positive stop by SOC.

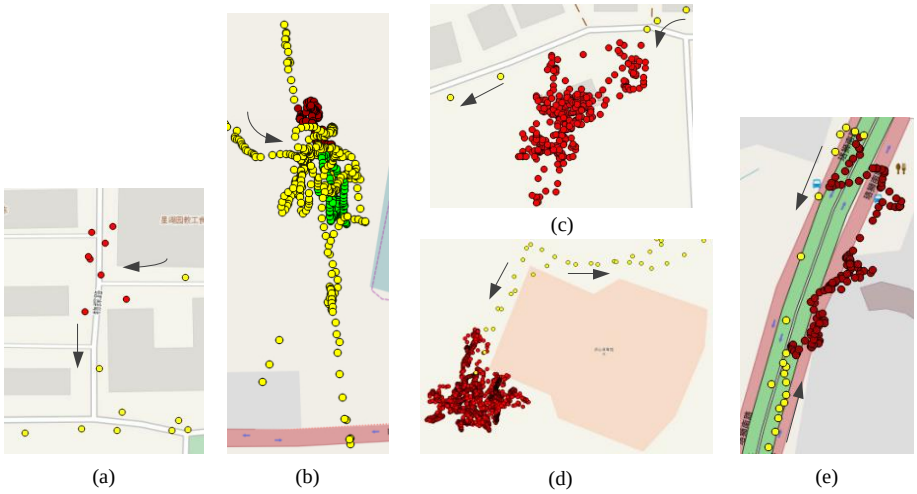

Figure 6. Cases of: (**a**) an effective stop of short indoor stay, detected without merging; (**b**) a separated stop of long indoor stay; (**c**) an effective stop of outdoor wandering, merged by Criterion 1; (**d**) an effective stop of indoor fitness, merged by Criterion 2; and (**e**) a false positive stop of a slow U-turn under a viaduct.

7.3. Parameter Setting Evaluation

A set of experiments against the four trajectory datasets were conducted to evaluate the influence of three key parameters on the performance of SOC. The results are given in Table 5. The first two experiments try to evaluate the influence of *Eps* when *Tau* is fixed to the default. As *Eps* decreases (Experiment 1), the conditions to define a core sequence become stricter. Accordingly, not only do Eps-reachability sequences shrink in size but also decrease in number. Consequently, decreasing *Eps* is very likely to cause fewer false positive stops and more separated/undetected stops, so that the number of effective stops is highly likely to decrease. Conversely, as *Eps* increases (Experiment 2), the opposite results will be observed. The next two experiments aim to evaluate the influence of *Tau* when *Eps* is fixed to the default. As *Tau* decreases (Experiment 3), the strict requirements for creating a core sequence are alleviated. Hence the likelihood of detecting false positive stops increases, but the likelihood of causing separated/undetected stops decreases, thereby likely resulting in more effective stops. Similarly, the opposite result occurs as *Tau* increases (Experiment 4). In the following two experiments, *Eps* varies but *Eps*/*Tau* is fixed to the default. As *Eps* decreases, the core sequence becomes harder to satisfy, and *vice versa*. Therefore, the results will be similar to the first two experiments.

To summarize, *Eps* has a higher influence on detection results than *Tau*. A too-small or too-large *Eps* is detrimental to detecting stops; a small value may introduce separated stops, while a large value may cause false positive stops. The ratio of *Eps*/*Tau*, which implicitly confines a stop to a small average speed, should be relatively small. According to our experiments, values close to the default setting are appropriate.

Finally, we try varying only the *MinMov* parameter. As *MinMov* decreases, the merging capability of SOC decreases; consequently, some labeled stops (usually occurring indoors) may fail to be merged into effective stops and will instead appear as separated stops. For example, when *MinMov* is decreased from 150 s to 90 s (Experiment 8), ten additional labeled stops (four in Dataset 2 and six in Dataset 3) are classified as separated stops. In general, *MinMov* should be large enough to eliminate the influence of noise for trajectories collected in error-prone environments. Nevertheless, a too-large *MinMov* value is not advisable because a curved movement may be mistakenly detected and merged as a stop.

Note that it is not an easy task for users to determine proper input parameters for SOC. Through the above analysis, we found that SOC works well with the default settings in most cases, *i.e.*, it achieves high recognition of effective stops. For trajectories with poor positioning precision (which often occurs in large cities), however, it is more suitable to assign larger values to both *Eps* and *Tau*, such as 50 m and 125 s, respectively. When a trajectory contains a large number of noise points, *MinMov* should be increased accordingly, for example, to 300 s. In addition, we chose a default threshold of three minutes to filter trivial stops of short duration.

Table 5. The influence of three key parameters on run results.

Exp.	Eps (m)	Tau (s)	MinMov (s)	Effective/False Positive/Separated/Undetected Stops			
				Dataset 1	Dataset 2	Dataset 3	Dataset 4
1	20	75	180	108/0/0/13	34/2/6/3	64/5/6/5	49/2/1/4
2	50	75	180	115/0/0/6	39/3/2/2	71/5/2/2	52/2/1/0
3	30	50	180	113/0/0/8	37/3/4/2	70/5/2/3	52/2/1/0
4	30	100	180	109/0/0/12	37/3/4/2	66/5/4/5	50/2/1/2
5	20	50	180	110/0/0/11	36/3/5/2	68/5/4/3	51/2/1/1
6	50	125	180	113/0/0/8	38/3/3/2	70/5/3/2	51/2/1/1
7	30	75	150	112/0/0/9	36/3/5/2	67/5/5/3	51/2/1/1
8	30	75	90	112/0/0/9	33/3/8/2	63/5/9/3	50/2/2/1

8. Discussion and Conclusion

8.1. Discussion

A stop implies some purposive activity, and therefore, it should last a minimum amount of time. From this point of view, slow U-turns are considered as false positive stops in this paper, because a U-turn is just a pure moving without other activities. This also accounts for why SOC introduces parameter *MinStp*, with which temporary stationaries such as waiting for traffic lights can be detected as false positive stops and filtered out. It should be pointed out that *MinStp* is actually an application-dependent parameter, though it is fixed as three minutes in this paper. For example, some applications may view a five minutes of friend meeting on the street as a stop, but other applications may be only interested in those stops that last longer than half an hour.

Three criteria are applied in SOC to handle stops interrupted by noise points. The first two criteria explore the spatio-temporal proximity of points within a stop, while the last criterion makes use of location information behind stops. Of course, if each Eps-reachability sequence can be reverse geocoded to an addressable location, the last criterion can take over the whole function of sequence merging because it is not only more accurate but also carries semantics. In reality, however, stops can occur at places without precise addresses. Moreover, the collected address archive in reverse geocoders is limited even for cities. Hence, the first two Criteria are the first choice of SOC for sequence merging. Note that with reverse geocoding, the output stops can have address semantics attached to them, through which advanced information such as the purpose of the stop can be further inferred. However, that is beyond the scope of this paper.

The definition of core sequence does not set any speed limitations on individual points but implicitly imposes a restriction on average speed using the ratio of 2**Eps*/*Tau*. As declared in the experimental section, this should be close to the default setting (*i.e.*, 2*(30/75)*3.6 = 2.88 km/h), which

is obviously slower than the average human walking speed (*i.e.*, 5 km/h). Because stops show diverse manifestations (they may be single points or sequences containing different numbers of points and degrees of noise), heading direction, another important movement parameter, is not appropriate for identifying stops. Instead, it is employed by SOC to filter false positive stops caused by slow U-turns.

It should be noted that some GPS receivers log not only time-stamped positions but also additional information, such as the number of satellites in view and the position dilution of precision (PDOP). When such information is recorded detecting indoor stops becomes relatively easy because when staying indoors, the number of satellites in view will be less than four and PDOP will have a high value [33]. However, these additional measures are not always available in trajectories, which explains why we developed a clustering-then-merging strategy to meet the challenge of recognizing stops, particularly indoor stops.

8.2. Conclusion

In this paper, we proposed a novel approach to extract stops from single trajectories with noise. Our proposed approach uses a sequence-oriented clustering method, which considers both spatial proximity as well as continuity and duration over time when clustering trajectory points. The main contributions of this paper are as follows:

(1) To capture the inherent characteristics of a trajectory stop, the concept of core sequence was introduced. A core sequence does not involve the speed of individual points but simply requires that the points of a sequence present spatial proximity and have a relatively long duration. In addition, the concepts used to grow core sequences were defined, and criteria were proposed for merging Eps-reachability sequences.

(2) An algorithm, called SOC, was developed to recognize effective stops and eliminate false positive stops. Moreover, the reachability distances of trajectory points were represented and understood graphically using a reachability graph, which intuitively illustrates the clustering structure and levels of a trajectory.

(3) We conducted extensive experiments on four real-world trajectory datasets to evaluate the performance of SOC. The results show that it is fairly effective for extracting stops even for trajectories with serious noise levels. In addition, we provided guidelines for setting the input parameters for the SOC algorithm to their proper values.

Geographical data were utilized in this paper but were restricted to only the sequence merging operation. In future work, we will improve our approach by integrating not only contextual geographical data but also application-related information such as road network data and land use data. Integrating such data can contribute greatly to gain more valuable information about stops. In addition, we are also interested in extending SOC to investigate the problem of stop detection at different geographical scales.

Acknowledgments: This research was supported by the National Sciences Foundation of China (Gant Nos. 41471374 and 41001296).

Author Contributions: LX participated in the design of main methods and composition of manuscript. GM provided key suggestions for improving methods of finding stops and designed experimental schemes. TW collected trajectory datasets and was involved in the implementation of coding and experiments. All authors read and approved the final manuscript.

Conflicts of Interest: The authors declare that they have no competing interests.

References

1. Spaccapietra, S.; Parent, C.; Damiani, M.; Macedo, J.; Porto, F.; Vangenot, C. A conceptual view on trajectories. *Data Knowl. Eng.* **2008**, *65*, 126–146. [CrossRef]
2. Cudré-Mauroux, P.; Wu, E.; Madden, S. Trajstore: An adaptive storage system for very large trajectory data sets. In Proceedings of the 26th International Conference on Data Engineering, Long Beach, CA, USA, 1–6 March 2010.

3. Guting, R.H.; Bohlen, M.H.; Erwig, M.; Jensen, C.S.; Lorentzos, N.A.; Schneider, M.; Vazirgiannis, M. A foundation for representing and querying moving objects. *ACM Trans. Database Syst.* **2000**, *25*, 1–42. [CrossRef]
4. Hadjieleftheriou, M.; Kollios, G.; Bakalov, P.; Tsotras, V.J. Complex spatio-temporal pattern queries. In Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, 30 August–2 September 2005.
5. Sakr, M.A.; Guting, R.H. Spationtemporal pattern Queries. *GeoInformation* **2011**, *15*, 497–540. [CrossRef]
6. Giannotti, F.; Nanni, M.; Pinelli, F.; Pedreschi, D. Trajectory pattern mining. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, CA, USA, 12–15 August 2007.
7. Lee, J.G.; Han, J.; Whang, K.Y. Trajectory clustering: A partition-and-group framework. In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, Beijing, China, 11–14 June 2007.
8. Yan, Z.; Chakraborty, D.; Parent, C.; Spaccapietra, S.; Aberer, K. Semantic trajectories: Mobility data computation and annotation. *ACM Trans. Intell. Syst. Technol.* **2012**, *9*, 1–34. [CrossRef]
9. Xu, J.; Guting, R-H. A generic data model for moving objects. *GeoInformation* **2013**, *17*, 125–172. [CrossRef]
10. Wolf, J. Applications of New Technologies in Travel Surveys. Available online: http://www.isctsc.cl/archivos/2004/B4%20-%20Resource%20Wolf.pdf (accessed on 01 June 2015).
11. Ester, M.; Kriegel, H.M.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996.
12. Ankerst, M.; Breunig, M.M.; Kriegel, H.P.; Sander, J. OPTICS: Ordering points to identify the clustering structure. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Philadelphia, PH, USA, 31 May–3 June 1999.
13. Birant, D.; Kut, A. ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data Knowl. Eng.* **2007**, *60*, 208–211. [CrossRef]
14. Krumm, J.; Horvitz, E. Predestination: Inferring destinations from partial trajectories. In Proceedings of the UbiComp 2006: The 8th International Conference on Ubiquitous Computing, Orange Country, CA, USA, 17–21 September 2006.
15. Li, Q.; Zheng, Y.; Xie, X.; Chen, Y.; Liu, W.; Ma, W. Mining user similarity based on location history. In Proceedings of the ACM Sigspatial Conference on Advanced in Geographical Information Systems, Irvin, CA, USA, 5–7 November 2008.
16. Rocha, J.; Time, V.; Oliveira, G.; Alvares, L.; Bogorny, V.; Times, V. DB-SMoT: A direction-based spatio-temporal clustering method. In Proceedings of the IEEE Conference of Intelligent Systems, London, UK, 7–9 July 2010.
17. Alvares, L.O.; Bogorny, V.; Kuijpers, B.; Fernandes, J.A.; Moelans, B.; Vaisman, A. A model for enriching trajectories with semantic geographical information. In Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems, Seattle, WA, USA, 7–9 November 2007.
18. Buchin, M.; Driemel, A.; Kreveld, M.; Sacristan, V. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *J. Spat. Inf. Sci.* **2011**, *3*, 33–63.
19. Junior, A.; Moreno, B.; Times, V.; Matin, S.; Cabral, L. GPASP-UTS: An algorithm for unsupervised trajectory segmentation. *Int. J. Geogr. Inf. Sci.* **2015**, *29*, 46–68. [CrossRef]
20. Zheng, Y.; Zhang, L.; Xie, X.; Ma, W. Mining interesting locations and travel sequences from GPS trajectories. In Proceedings of the 18th International World Wide Web conference, Madrid, Spain, 20–24 April 2009.
21. Cao, X.; Cong, G.; Jensen, C.S. Mining significant semantic locations from GPS data. *Proc. VLDB Endow.* **2011**, *3*, 1009–1021. [CrossRef]
22. Dodge, S.; Weibei, R.; Lautenschutz, A.K. Towards a taxonomy of movement patterns. *Inf. Vis.* **2008**, *7*, 240–252. [CrossRef]
23. Parent, C.; Spaccapietra, S.; Renso, C.; Andrienko, G.; Andrienko, N.; Bogorny, V.; Damiani, M.; Gkoulalas-divanis, A.; Macedo, J.; Pelekis, N.; *et al*. Semantic trajectories modeling and analysis. *ACM Comput. Surv.* **2012**, *45*, 1–32. [CrossRef]
24. Palma, A.T.; Bogorny, V.; Kuijpers, B.; Alvares, L.O. A clustering-based approach for discovering interesting places in trajectories. In Proceedings of the 2008 ACM Symposium on Applied Computing, Fortaleza, Brazil, 16–21 March 2008.

25. Zimmermann, M.; Kirste, T.; Spiliopoulou, M. Finding stops in error-prone trajectories of moving objects with time-based clustering. *Commun. Comput. Inf. Sci.* **2009**, *53*, 275–286.
26. Tran, L.H.; Nguyen, Q.V.H.; Do, N.H.; Yan, Z. Robust and hierarchical stop discovery in sparse and diverse trajectories. Available online: http://infoscience.epfl.ch/record/175473 (accessed on 01 February 2016).
27. Schuessler, N.; Axhausen, K.W. Processing raw data from global positioning systems without additional information. *J. Transp. Res. Board* **2009**, *2105*, 28–36. [CrossRef]
28. Google Maps: Geocoding APIs. Available online: https://developer.google.com/maps/documentation/geocoding (accessed on 10 July 2015).
29. Openstreetmap. Available online: http://www.openstreetmap.org (accessed on 03 June 2015).
30. GeoLife: Building social networks using human location history. Available online: http://research.microsoft.com/en-us/projects/geolife (accessed 01 June 2015).
31. QGIS: A Free and Open Source Geographic Information System. Available online: http://qgis.org/en/site (accessed on 20 September 2015).
32. Baidu Maps: Geocoding APIS. Available online: http://developer.baidu.com/map/index.php?title=webapi/guide/webservice-geocoding (accessed on 10 July 2015).
33. Ogle, J.; Guensler, R.; Bachman, W.; Koutsak, M.; Wolf, J. Accuracy of Global Positioning System for determining driver performance parameters. *Transp. Res. Rec.* **2002**, *1818*, 12–24. [CrossRef]