International Journal of
Geo-Information

Article

An Improved DBSCAN Algorithm to Detect Stops in Individual Trajectories

Ting Luo [1,2], Xinwei Zheng [1], Guangluan Xu [1], Kun Fu [1,*] and Wenjuan Ren [1]

1 Key Laboratory of Spatial Information Processing and Application System Technology, Institute of Electronics, Chinese Academy of Sciences, Beijing100190, China; 15110288226@163.com (T.L.); zxw_1020@163.com (X.Z.); gluanxu@mail.ie.ac.cn (G.X.); renandliang@sina.com (W.R.)
2 School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100190, China
* Correspondence: kunfuiecas@gmail.com; Tel.: +86-10-5888-7208 (ext. 8931)

Academic Editor: Wolfgang Kainz
Received: 18 December 2016; Accepted: 21 February 2017; Published: 25 February 2017

Abstract: With the increasing use of mobile GPS (global positioning system) devices, a large volume of trajectory data on users can be produced. In most existing work, trajectories are usually divided into a set of stops and moves. In trajectories, stops represent the most important and meaningful part of the trajectory; there are many data mining methods to extract these locations. DBSCAN (density-based spatial clustering of applications with noise) is a classical density-based algorithm used to find the high-density areas in space, and different derivative methods of this algorithm have been proposed to find the stops in trajectories. However, most of these methods required a manually-set threshold, such as the speed threshold, for each feature variable. In our research, we first defined our new concept of move ability. Second, by introducing the theory of data fields and by taking our new concept of move ability into consideration, we constructed a new, comprehensive, hybrid feature–based, density measurement method which considers temporal and spatial properties. Finally, an improved DBSCAN algorithm was proposed using our new density measurement method. In the Experimental Section, the effectiveness and efficiency of our method is validated against real datasets. When comparing our algorithm with the classical density-based clustering algorithms, our experimental results show the efficiency of the proposed method.

Keywords: trajectory data; stops and moves; improved DBSCAN algorithm; temporal and spatial properties

1. Introduction

In recent years, miniaturized GPS (global positioning system) devices have become more widely used in daily life and large amounts of target trajectory data can be easily recorded. For instance, people's daily activity trajectories can be recorded by car GPS equipment and GPS-enabled mobile phones. A common trajectory of a person's daily life is illustrated in Figure 1. Useful information can be extracted from these trajectories and they can be used to benefit daily life. As a result, many location-based services, such as position-based recommender systems and destination prediction systems, are receiving increasing attention from both users and developers. The primary concern of location-based applications is how to understand the semantic meaning of a trajectory, and not just to consider trajectory as a combination of recorded points. The work in Reference [1] proposed a conceptual model to present trajectories with semantic annotations, allowing one to assign semantic information, such as moves and stops, to specific parts of trajectories. Stops in trajectories represent the trajectory segments corresponding to a person's stay in certain locations. Moves correspond to the trajectory segments created by the motion of a target between stop locations.

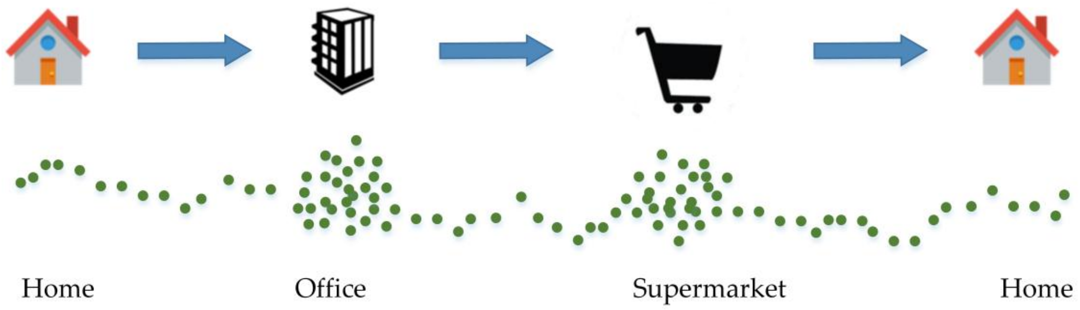


Figure 1. An example of a trajectory.

Stop locations in a trajectory are an indispensable part of various applications, such as purpose prediction services, navigation services, and generic or personalized recommendations. In this paper, the problem of how to extract stop locations from trajectories is called stop detection. In the literature, many models have been proposed to divide a trajectory into stop parts and move parts. Research on stop detection can be divided into two categories: static methods and dynamic methods. Important positions are defined in advance for static techniques [2,3], while no prior knowledge regarding stops is given for a dynamic approach. Recently, several papers have studied the dynamic solution by considering different aspects of mobility characteristics, such as velocity characteristics. Typically, general clustering algorithms, which are able to cluster one's stop locations by assigning different constraints to different features, are adopted in the dynamic solution.

In general, most of the existing clustering methods used in stop detection suffer from their respective drawbacks. First, the value of commonly used characteristics in these clustering methods, such as speed, intensely fluctuates when dealing with a real trajectory. We provide a qualitative analysis about the speed feature in Section 3. Furthermore, this problem further leads to the second set of drawbacks, namely that, in most cases, the algorithms need to be given manually-set parameters for different features, which is a difficult task for users due to the fluctuations described above. Finally, most of the clustering-based algorithms take the number of GPS points within a given distance as a measurement of density. As a result, these methods ignore sequential features and the results of these works are dramatically affected by distance parameters. Additionally, this will be worse when multiple features are considered together, since users have to specify a parameter for each feature, respectively.

In this paper, taking the aspects described above into consideration, we constructed a new, comprehensive, hybrid feature–based density measurement method. In our method, we define a new concept for move ability and apply data field theory, proposed in Reference [4], to measure the density around a GPS point; the new concept of move ability is considered by giving density a move ability–dependent weight. In our work, the density threshold is automatically determined when calculating core points. After that, we use our density measurement method to improve the original DBSCAN (density-based spatial clustering of applications with noise) algorithm.

The rest of our paper is organized as follows. Some common stop detection algorithms are presented in Section 2. In Section 3, we give the definitions of some basic concepts, for example, detailed definitions of GPS trajectory and stops. After describing our improved DBSCAN algorithm in detail in Section 4, we validate our method with real datasets both in terms of feasibility and efficiency by comparing it with the four other algorithms in Section 5. We conclude our work in Section 6.

2. Related Works

In this section, we provide a survey of the clustering algorithms described or analyzed in the literature. Various of methods can be used to extract the stop locations in GPS trajectories. In general, the approaches for stop detection can be summarized into two categories: static methods and dynamic methods. In static techniques [2,3], important positions, such as gas stations, are defined in advance. When extracting stops from trajectories, if targets enter into a predefined region and the stay duration exceeds the duration threshold, this previously defined region is regarded as a stop location in the trajectory. The main drawback of static algorithms is that users need to specify their respective places of interest. As a result, some interesting and personalized stop locations will not be found if they are not provided by users beforehand.

As for dynamic approaches, no prior knowledge regarding stops is given and personalized stop locations can be discovered. Multiple sources from the literature have studied the dynamic solution by considering different aspects of mobility characteristics [5–10]. Considering only the spatial characteristics, several classical clustering algorithms are introduced to extract stops from a trajectory. A predictive model, based on automatically detected stop positions, is proposed in Reference [11], and the authors adopted a variation of the traditional K-Means methods in order to detect stop locations. The selection of the value of parameter *K* and the initial clustering center is the main issue, and will directly affect the final results. The DBSCAN [12] algorithm is used in Reference [13] to extract significant locations. In Reference [14], a modified DBSCAN algorithm, DJ-Cluster (density and join-based clustering algorithm), is proposed to detect personal meaningful places. These density-based clustering algorithms can overcome many limitations of the K-Means approach [15]; however, they only take spatial dimensions into consideration and the temporal sequential features are ignored.

Compared with the algorithms described above, many studies have taken both the spatial and temporal characteristics into consideration. Different derivative methods of the DBSCAN method, with temporal sequential characteristic being considered, have been adopted by many researchers in order to extract stop positions [5,6,14,16,17]. In Reference [5], an improved DBSCAN algorithm with gap treatment was proposed to detect stop episodes in a trajectory. The CB-SMoT (clustering-based stops and moves of trajectories) algorithm was proposed in Reference [6] to extract known and unknown stops. As it considers temporal speed and spatial features, CB-SMoT is a density-based clustering algorithm. In detail, clusters are generated by evaluating trajectory sample points at a slower speed than the velocity threshold. In addition, one of the major parameters in Reference [6], namely *Eps* (a given distance threshold around which the points are regarded as neighbors), is obtained using a quantile function. As is described in Reference [16], the quantile function in Reference [6] does not always work in estimating the appropriate value for the parameter *Eps*, making it difficult to determine an appropriate threshold for the parameter. The method proposed in Reference [16] improves the CB-SMoT algorithm by proposing an alternative for calculating the *Eps* parameter, but it is still difficult to calculate it as it depends on users to distinguish the low speed part and high speed part. Additionally, by assigning different thresholds to different characteristics, some clustering approaches have been proposed [18–21]. Especially, information from satellites is introduced in the TDBC (a spatio-temporal clustering method used to extract stop points from individual trajectory) algorithm [21]. Additionally, a time-based clustering algorithm was proposed in Reference [18] and both the clustering distance threshold and the time threshold are needed.

The methods mentioned above can obtain a desirable performance in some situations; however, these methods also have their drawbacks. Most of these methods need to assign appropriate threshold values for each parameter. While calculating the density of GPS points, most clustering-based algorithms take the number of GPS points within a given distance into account, without considering their consequential characteristics. In this paper, the density of GPS points will be calculated using the adjacent points over the trajectory, but not the overall spatial points. First, we define the new concept of the move ability feature. To the best of the authors' knowledge, the move ability feature was first proposed in stop detection. After that, by combining the theory of the data field, proposed in Reference [4], and our new concept of move ability, we construct a new, comprehensive, hybrid feature–based, density measurement method. In our method, the density threshold is automatically determined when calculating core points. Finally, we use our density measurement method to improve the original DBSCAN algorithm.

3. Basic Concepts

In this section, we show the definitions of GPS trajectory, stop, and move, based on the general definitions in Reference [1]. These definitions will be used in the rest of this paper. These definitions are given according to the particular application studied in this paper; for example, altitude is not considered in this paper since there are small variations in altitude within urban regions.

Definition 1. *GPS Trajectory:A GPS Trajectory is a list of GPS data points* $\{p_0 = (x_0, y_0, t_0), p_1 = (x_1, y_1, t_1), \ldots, p_n = (x_n, y_n, t_n)\}$, *where* $\forall i \in [1, n]$, $p_i = (x_i, y_i, t_i)$ *and* $t_i < t_{i+1}$, *and* x_i, y_i *and* t_i *represent the longitude, latitude, and timestamp, respectively.*

Stops represent the significant places of a GPS trajectory where a target has spent a minimal amount of time, and, essentially, with a higher density of GPS points. A move represents the trajectory between stops and is equipped with a lower density of GPS points. In Reference [1], Spaccapietra defined some of their characteristics.

Definition 2. *Stop: A stop is a part of a trajectory and the features are as follows: (i) the user has explicitly defined this part of the trajectory to represent a stop; (ii) the temporal extent is a non-empty time interval; (iii) the traveling object does not move as far as the application view of this trajectory is concerned; and (iv) all stops in the same trajectory are temporally disjointed, i.e., the temporal extents of two stops are always disjointed.*

Definition 3. *Move: A move is a part of a trajectory, such that: (i) the part is delimited by two extremities that represent either two consecutive stops, or* t_{begin} *and the first stop, or the last stop and* t_{end}, *or* $[t_{begin}, t_{end}]$ *(the case when a trajectory has no stops); (ii) the temporal extent* $[t_{begin}, t_{end}]$ *is a non-empty time interval; (iii) the spatial range of a trajectory for the* $[t_{begin}, t_{end}]$ *interval is a spatio-temporal line (not a point) defined by the trajectory, where* t_{begin} *is the initial point of the trajectory and tend is the final one.*

Definition 4. *Distance: The distance between two points* $< p_n, p_m >$ *is denoted by*:

$$Dist(p_n, p_m) = 2R \times arcsin\sqrt{sin^2\left(\frac{lat_m - lat_n}{2}\right) + cos(lat_n) \times cos(lat_m) \times sin^2\left(\frac{lgt_m - lgt_n}{2}\right)} \quad (1)$$

where R represents the radius of the Earth ($R = 6371$ *km),* lat_n *and* lat_m *represent the latitudes of* p_n *and* p_m, *respectively; similarly,* lgt_n *and* lgt_m *represent the longitude.*

Definition 5. *Trajectory curve distance: the curve distance of a sub-trajectory segment,* $traj_{nm}$, *which is composed of a sequence of points* $\{p_n, p_{n+1}, \ldots, p_m\}$, *and is denoted by*

$$TrajCurveDist(traj_{nm}) = \sum_{k=n}^{m-1} Dist(p_k, p_{k+1}) \quad (2)$$

Definition 6. *Trajectory direct distance: the direct distance of sub-trajectory segment* $traj_{nm} = \{p_n, p_{n+1}, \ldots, p_m\}$ *equals the distance between the first point and the last point in the sub-trajectory and is denoted by*:

$$TrajDirectDist(traj_{nm}) = Dist(p_n, p_m) \quad (3)$$

In general, when a target stays at a stop region, the corresponding trajectory direct distance is far less than the trajectory curve distance. On the contrary, the corresponding trajectory direct distance would be close to the trajectory curve distance when the target moves between stop regions. Taking this into consideration, we propose our new concept of move ability.

Definition 7. *Move ability: the move ability of a sub-trajectory segment* $traj_{nm} = \{p_n, p_{n+1}, \ldots, p_m\}$ *is denoted by:*

$$MoveAbility(traj_{nm}) = \frac{TrajDirectDist(traj_{nm})}{TrajCurveDist(traj_{nm})} \tag{4}$$

Figure 2 illustrates the concept of move ability. In the figure, there are three sub-trajectories, each of which contains six points. In detail, the coordinates of each point illustrate the spatial longitude and latitude in the real world. In addition, for simplicity, the Euclidean distance is used in this illustration to calculate the move ability features. These three sub-trajectories represent real trajectories corresponding to different situations: Figure 2a represents the activity at a stop; Figure 2b represents the movement on curved roads; Figure 2c represents a linear motion in reality. Comparing the move ability of each sub-trajectory in Figure 2, the results are consistent with our reasoning, described above.

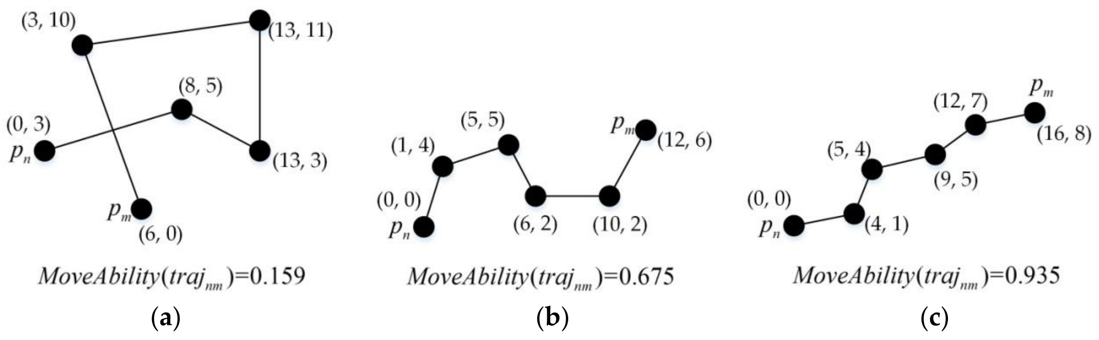


Figure 2. Examples of move ability. (**a**) The activity at a stop; (**b**) movement on curved roads; (**c**) linear motion.

Furthermore, we find that our new concept of move ability is more suitable for distinguishing move and stop episodes. A qualitative comparison between the velocity feature and the move ability was done. Taking a real track as an example, the velocity curve after Gaussian smoothing is shown in Figure 3a. The velocity curve shows that the speed of moving objects can vary dramatically and there are many short, slow-speed segments during high-speed parts, which may be caused by short decelerations in motion. Comparatively, the move ability curve is more stable and discriminatory. The smoothed move ability curve, using the same Gaussian kernel, is shown in Figure 3b. Especially, a low value for move ability is only obtained when the target stays in movement around a certain region, which is likely to be a stop region. In addition, even a low-speed sub-trajectory may achieve a high move ability; for example, when a target moves in an approximately linear fashion with a low speed, this can help to remove some fake stops, such as short-duration traffic jams.

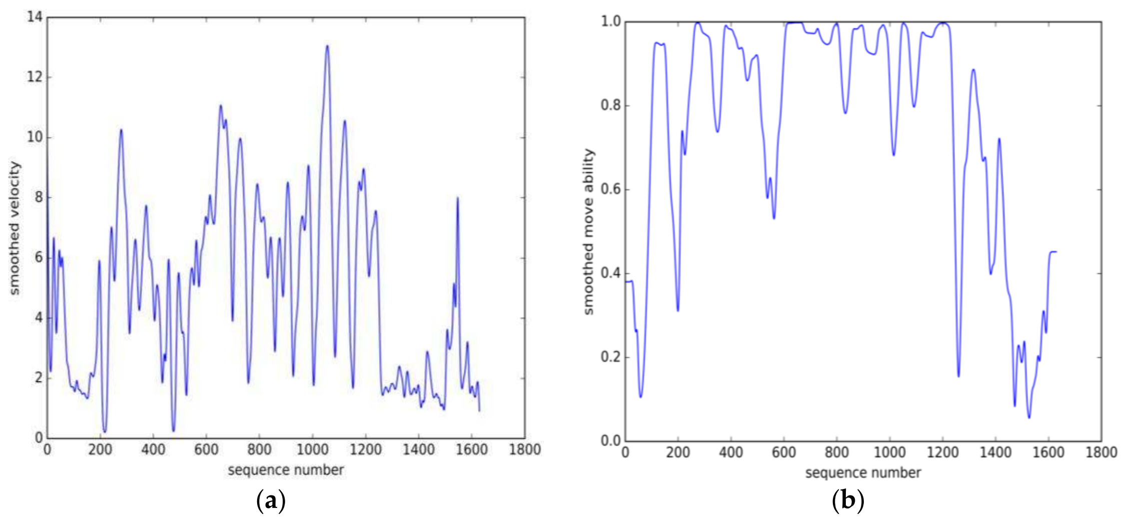


Figure 3. The velocity curve and move ability curve of a real-world track. (**a**) Smoothed velocity curve; (**b**) smoothed move ability curve.

4. Methodology

The basis of our approach is that the stop part of a trajectory should have a lower move ability and a higher density of GPS points; therefore, it is meaningful to find an appropriate method to estimate the density when move ability is considered. Especially, no special treatment is done when dealing with long-duration gaps in a trajectory. These gaps in trajectory may be caused by many things, such as a GPS logger running out of power. It is inappropriate to assume that a target stays in the same place. In our method, trajectory point density is the primary consideration and the short temporal gaps in trajectories have little effect on the results.

Following the reasoning described above, we propose a comprehensive, hybrid feature–based, density measurement method. Furthermore, we improve the original DBSCAN algorithm by using our own density measurement method.

4.1. Density Function

DBSCAN [12] is a classic density-based clustering method. The density of a current point is measured by the number of points within a certain distance from the current point. In our work, taking our new concept of move ability into consideration, we propose a method to measure density by introducing the data field proposed by Li et al. [4]. According to the data field, each trajectory point will receive an interactive impact from other points. Without losing generality, we estimated the impact between points using a Gaussian function, which has acceptable mathematical properties, and the equation is shown as follows:

$$\varphi(p_i) = \sum_{j=1}^{n} e^{-\left(\frac{d_{ij}}{\sigma_1}\right)^2} \tag{5}$$

where $p_i (i = 1, \ldots, n)$ represents a trajectory point, d_{ij} equals the distance between p_i and p_j, and σ_1 represents the standard deviation.

In our work, we regard the summation of the impact from a set of points before and after p_k as an alternative to density. The number of these adjacent points is presented as the input parameter *Nap*. In addition, we take our new concept of move ability into account by multiplying a move ability–related weight function. The final density of points is calculated as follows:

$$\varphi(p_i) = e^{-\left(\frac{MA_i}{\sigma_{MA}}\right)^2} \times \sum_{j \epsilon adj(i)} e^{-\left(\frac{d_{ij}}{\sigma_1}\right)^2} \tag{6}$$

where MA_i represents the move ability of the sub-trajectory, which is denoted by the adjacent points described above, σ_{MA} represents the standard deviation of the weight function, and *adj(i)* represents the adjacent points, before and after p_i, and the other parameters are the same as those in Equation (5).

4.2. Improved DBSCAN

In our method, most of the concepts of DBSCAN are the same as the original definition [12]. Instead of using a minimal number of points to define the core point, we define the core point as follows:

Definition 8. *Core point: A trajectory point $p_i = (x_i, y_i, t_i)$ of a trajectory is called the core point with respect to MinDensity, if $\varphi(p_i) > MinDensity$, where MinDensity represents the threshold of density.*

In our paper, in order to find the appropriate value for *MinDensity*, like the quantitative method in the work of Yuan [22], we consider the 'elbow point' as the threshold. The 'elbow point' refers to the point with the maximum curvature and usually indicates the cut-off point of two states. In our method, a curvature calculation method, KD (a technique to estimate the curvature on curves) curvature [23], is used to determine the 'elbow point'. Taking three consecutive points, $\{p_{i-1},\ p_i, p_{i+1}\}$, as an example, the corresponding KD curvature is calculated as follows:

$$K_d(p_i) = \frac{\pi - \gamma_i}{\eta} \tag{7}$$

where $\gamma_i = \angle(p_{i-1}p_i,\ p_{i+1}p_i)$ and $\eta = (|p_{i-1}p_i| + |p_{i+1}p_i|)/2$.

Furthermore, the density sequence in practice fluctuates frequently and is not smooth enough, making it difficult to calculate the 'elbow point'. In order to get the exact 'elbow point', we smooth the density curve with a Gaussian kernel. After that, considering horizontal and vertical coordinates as being equally important, a normalization procedure is adopted to normalize the density sequence. In addition, we discretize the density curve by sampling with a given length interval, making it easier to calculate the 'elbow point'. Taking one track in our real dataset as an example, the non-smoothed original density sequence is shown in Figure 4a; the density sequence after smoothing is illustrated in Figure 4b.The figures show that there are indeed fluctuations and gaps in the sequence. In our algorithm, the normalized density sequence, after sampling with a length interval $\Delta l = 0.1$, is illustrated in Figure 5a. The asterisk in Figure 5b refers to the final 'elbow point' in the normalized density sequence, which is noted with a black arrow.

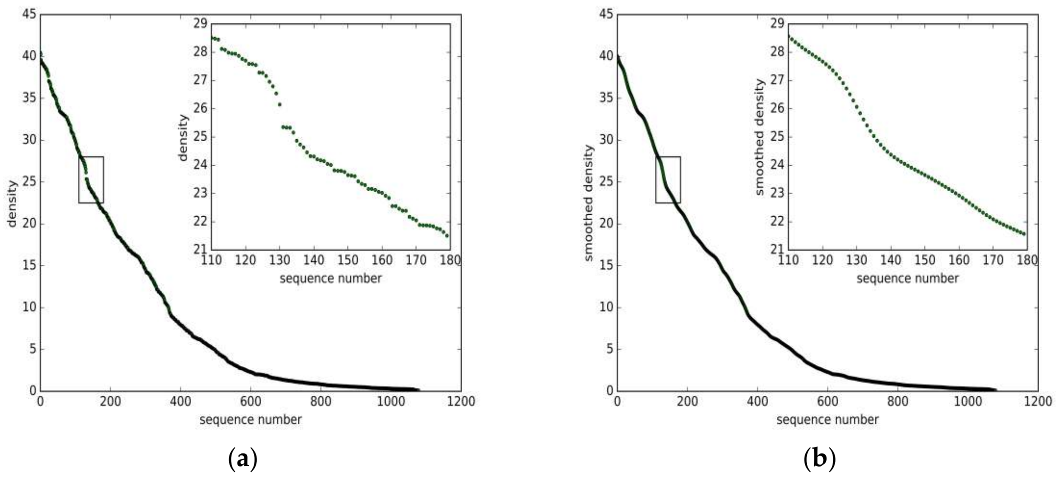


Figure 4. Density sequence. (**a**) Non-smoothed density sequence and (**b**) smoothed density sequence.

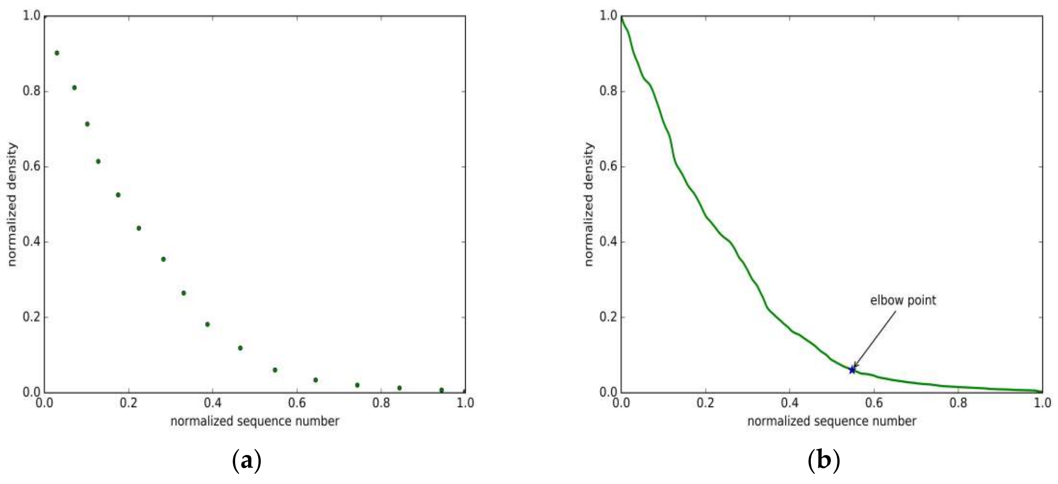


Figure 5. Sampled density sequence and 'elbow point'. (**a**) Density sequence after sample and (**b**) the final 'elbow point'.

Finally, a merger step is introduced in our improved DBSCAN algorithm. In detail, two consecutive stops with the same geographic location and a short temporal interval are merged. This is consistent with real life, in that people always move around certain regions, resulting in the appearance of several small spatiotemporal similar stops in the same region. In our experiments, two consecutive stops are merged if the distance between them is less than 200 m and the temporal interval is less than one hour.

Compared with the original DBSCAN algorithm, our method contains two main modifications for clustering in single trajectories: (i) we propose a new measurement method for density by suggesting the new concept of move ability and introducing the theory of data field; additionally, we take several points before and after p_i into consideration to measure local density; (ii) by introducing the quantitative method from Yuan [22], our method can automatically select an appropriate value for the density threshold.

5. Experimental Results

In this section, we validate our improved DBSCAN algorithm through experiments on real trajectory datasets. Comparative experiments between our method and two classic, density-based clustering algorithms were conducted. In the following sections, we first discuss the datasets, and then relevant experimental results are shown.

5.1. Datasets Description

In this paper, we use the Geolife dataset and our own collected dataset to perform our experiments. The Geolife dataset was collected by Microsoft Research Asia during their Geolife project. It shows the trajectories of 182 users from April 2007 to August 2012.In total, this dataset contains 17,621 trajectories. Each trajectory in this dataset consists of a sequence of temporal, ordered, time-stamped points; each point contains geographical coordinate information, such as longitude, latitude, and altitude. Additionally, more than 90% of these trajectories were recorded in a dense representation, namely, every 5–10 m or 1–5 s per point.

In our experiment, a software tool was developed based on the Bing Maps APIs, which is a web-mapping service provided by Microsoft. Based on this software, hundreds of trajectories were visually inspected by a group of research assistants. During this work, regions with high densities and long durations were labeled as stops. A labeled real trajectory in our dataset is illustrated in Figure 6, where S_1 and S_2 represent two stops. Considering that there are many short trajectory segments in the Geolife dataset, the trajectories selected for our experiment should be long enough to ensure that there are, indeed, stops in the trajectories. Finally, in order to verify our algorithm, we used 100 labeled trajectories, which were selected from the Geolife dataset, and cover more than 50 users. In detail, our selected trajectories were recorded daily by volunteers and contain different modes of transportation, such as walking and biking. Additionally, all of the trajectories were urban trajectories.

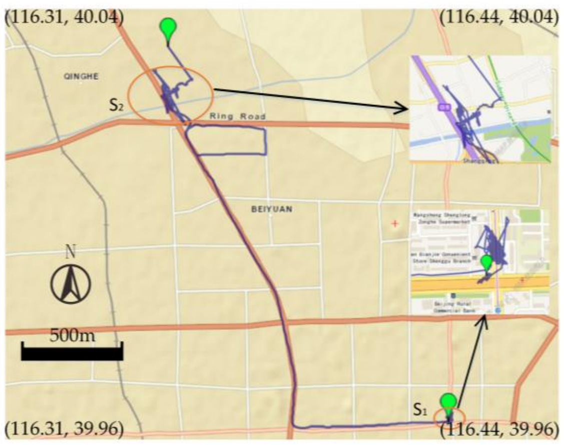
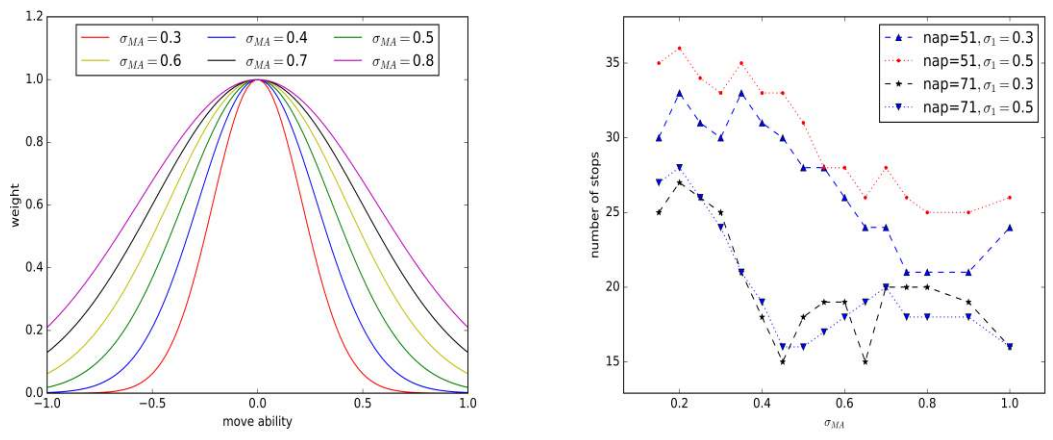


Figure 6. An example of a labeled trajectory.

The second dataset in our experiments was collected by our team, from daily life, using mobile phones. In total, this dataset contained the trajectories of our team for a month and was recorded every 2 s, per point. All stops in these trajectories were recorded in detail. In our experiments, 14 trajectories of this dataset were selected to validate the effectiveness of our new algorithm.

5.2.Parameter Estimation

In our improved DBSCAN algorithm, as shown in Equation (6), there are three parameters to be estimated in order to determine the density calculation model. These are *Nap* (the number of adjacent points), σ_1 and σ_{MA}. In order to find the appropriate value for these parameters, we used real trajectory data in our dataset to carry out on the simulation experiments for each parameter. The estimation results of these three parameters are shown in Figure 7.



(**a**) (**b**)

Figure 7. The estimation simulations of parameter σ_{MA}. (**a**)The variation of move ability weight with parameter σ_{MA}; (**b**) the number of stops found for different σ_{MA}.

In our improved DBSCAN algorithm, parameter σ_{MA} determines the weight assigned to different move abilities. As previously described, when the target stays in a region, the resulting sub-trajectories have a smaller move ability; when the target moves from one region to another, the resulting sub-trajectories have a larger move ability. Figure 7a shows the variations of move ability weight with parameter σ_{MA}. A high weight should be given to a point with a low move ability, and a low weight to a point with a high move ability. We found parameter $\sigma_{MA} = 0.5$ to be favorable as the value of a weight has a large distribution, without causing serious two-stage differentiation.

Parameter σ_1 determines the interactive impact between trajectory points. In our algorithm, the summation of impacts from adjacent points is used to measure the density around a certain trajectory point. Meanwhile, each point receives a stronger impact from closer points and a weaker impact from farther points. A lesser setting of σ_1 results in recieving weaker impact from farther points. When σ_1 is given too small a value, trajectory points will only receive an effective impact from closer points.

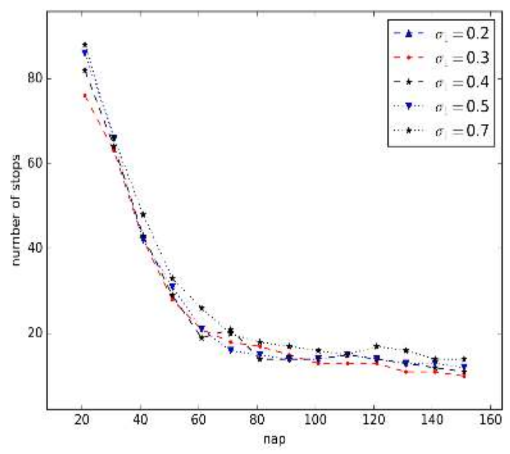
Parameter *Nap* demonstrates the number of adjacent points considered in our density calculations. When parameter *Nap* is given too small a value, the density is determined by a few adjacent points, resulting in a lower robustness to noise. In addition, *Nap* has an effect on the size of clusters. Since people always move at a small scale, even when they stay in a certain stop region, for example students move from place to place when they perform activities on a playground, too small a setting value for parameter *Nap* would reduce the size of the clusters and divide a larger stop region into several smaller stops. However, too large a setting value for *Nap* would make it difficult to detect smaller stops, as the local small changes of move ability would be smoothed.

In order to find the appropriate value for the three parameters in our algorithm, we selected five trajectory segments from the Geolife dataset to carry out simulation experiments for each parameter. By observing the total number of detected stops for different combinations of parameters, the appropriate range of values for each parameter was determined. The merge step in our algorithm was removed in our simulation experiments, as small stops, which are close to each other, may be merged. Figure 7b demonstrates the number of discovered stops for different values for parameter σ_{MA}. For each curve in Figure 7b, parameter *Nap* and parameter σ_1 are set to a fixed value. From the graph in Figure 7b, we can see that the curve decreases dramatically when σ_{MA} is less than 0.5. On the other hand, when the value of parameter σ_{MA} is greater than 0.5, the curve tends to be stable. Therefore, the value of parameter σ_{MA} was set to no less than 0.5 in the experiments.

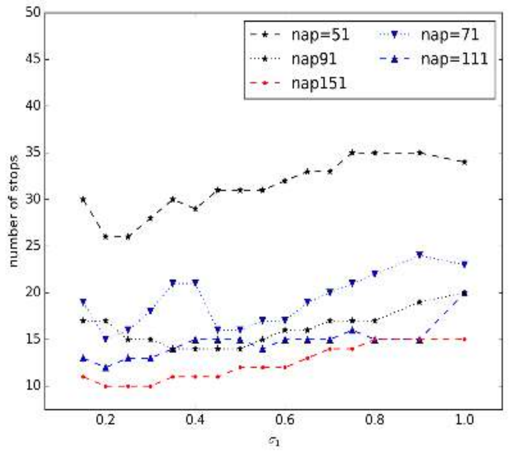
Similarly, with the premise that $\sigma_{MA} = 0.5$, the number of detected stops with different values for *Nap* and σ_1 is shown in Figure 8a,b, respectively. Figure 8a shows that the number of detected stops decreases intensely until $Nap > 50$. When the value of *Nap* is set to be larger than 50, the curves tend to be stable. The variation in the number of detected stops with different values for σ_1 is shown in Figure 8b. The graph shows that the number of detected stops increases slowly with an increase in σ_1. In our experiments, the value of *Nap* was set to be greater than 50. As for parameter σ_1, considering that the value should not be set too small, we set the parameter as $\sigma_1 > 0.3$.

In this paper, using one trajectory segment selected from the Geo-life dataset, we further estimated the values for parameters *Nap* and σ_1 by observing the *sse* (sum of squared error). As described in Reference [24], *sse* is an evaluation of the partitioning of detected locations. The smaller the *sse* value is, the better the clusters are; especially when the number of stops becomes very large, the *sse* tends to be very small. However, this does not mean a good result for the clustering.

In order to estimate the values for parameters *Nap* and σ_1, the parameter σ_{MA} is set to a fixed value ($\sigma_{MA} = 0.5$). Figure 9a shows the variations of *sse* with the growth of *Nap*. The *sse* is small when $Nap < 50$ and it becomes large when $Nap > 50$. Therefore, the value of parameter *Nap* was set as $Nap > 50$ in our experiments. Figure 9b demonstrates the variations of *sse* with the growth of σ_1. We can find that the larger σ_1 is, the larger *sse* becomes. This is consistent with our reasoning that σ_1 should not be set to be too small.

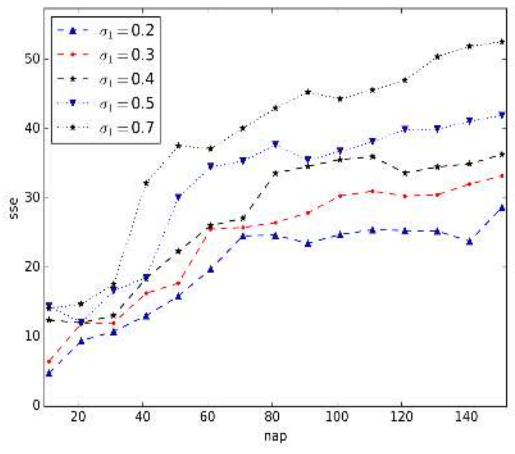


(**a**)

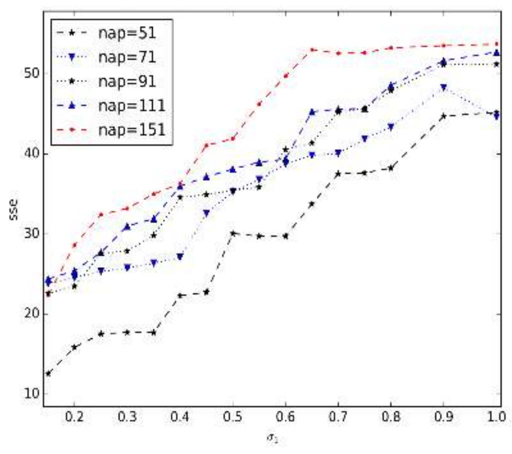


(**b**)

Figure 8. The number of detected stops with different values for parameters *Nap* and σ_1. (**a**) The number of stops found for different *Nap*; (**b**) the number of stops found for different σ_1.



(**a**)



(**b**)

Figure 9. The *sse* (sum of squared error) with different values for parameters *Nap* and σ_1. (**a**) The*sse* for different *Nap*; (**b**) the *sse* for different σ_1.

5.3.Evaluation of Effectiveness

In order to verify the feasibility of our algorithm, we compared our method with four other stop-detection algorithms, the CB-SMoT algorithm [6], DBSCAN algorithm, DJ-Cluster algorithm [14],and time-based clustering [18], using the public dataset and our own collected dataset. In this paper, we validate our algorithm using the same experimental method described in Reference [25], which also used the same Geolife dataset. In our experiments, the computation of precision and recall are as follows:

$$\text{Precision} = \frac{\text{number of correct stops found}}{\text{number of stops found}} \tag{8}$$

$$\text{Recall} = \frac{\text{number of correct stops found}}{\text{number ofcorrect stops}} \tag{9}$$

In addition, as described in Reference [25], the weighted harmonic mean of precision and recall, $\text{F}_{\text{measure}}$, is computed as follows:

$$\text{F}_{\text{measure}} = \frac{2 \times \text{Precison} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

In the first experiment, we conducted a comparative experiment on 100 labeled trajectories, which were selected from the public Geolife dataset. In addition, in order to investigate the experimental effects of different combinations of parameters, we ran the experiment with different combinations of parameters for our improved DBSCAN algorithm. In detail, all the parameters of our method were selected according to the discussion in Section 5.1. In our experiments, parameter $\sigma_{MA} = 0.5$ and remained unchanged.

Table 1 shows the experimental results of our algorithm using different combinations of parameters. The results show that our algorithm works best when $\sigma_1 = 0.3$ and *Nap*=51 in our dataset. Moreover, we found that the precision increased with an increase in parameter *Nap*, but the recall decreased. The main reason for this is that when *Nap* became larger, the local variation of the density was smoothed. As a result, only stops with a longer duration would be found in our algorithm; at the same time, some fake stops are discarded with the increase in *Nap*.

Table 1. The experimental results of the improved algorithm with the Geolife dataset.

No.	œ$_1$	Nap	Precision	Recall	F$_{\text{measure}}$
1	0.3	51	0.92105	0.7749	0.8417
2	0.3	71	0.9698	0.7122	0.8213
3	0.3	91	0.9267	0.6531	0.7662
4	0.5	51	0.7729	0.8413	0.8057
5	0.5	71	0.9167	0.7712	0.8377
6	0.5	91	0.9289	0.7232	0.8133

With respect to the other four algorithms, we ran several experiments using different combinations of the respective parameters. Considering the optimal value of the three measurements, namely the precision, recall and F$_{\text{measure}}$, the t optimal combinations of the parameters were selected to run comparative experiments using the improved algorithm. With respect to the CB-SMoT algorithm, as it is inappropriate to calculate using the quantile function, as described in Reference [16], the value of *Eps* was set manually when calculating the optimal combination of parameters.

Table 2 shows the experimental results of our algorithm and the other four clustering algorithms using the Geolife dataset. As can be seen in from Table 2, the improved DBSCAN method worked better than the other four algorithms in a real-world dataset.

Table 2. Experimental results with the Geolife dataset.

Metrics	OurMethod	CB-SMoT	DBSCAN	DJ-Cluster	Time-Based
Parameters	$\sigma_1 = 0.3$ *Nap* = 51	*Eps* = 50 *MinTime* = 5	*Eps* = 80 *MinPts* = 100	*Eps* = 80 *MinPts* = 80	*Eps* = 80 *Time* = 6
Labeled	271	271	271	271	271
Detected	228	358	241	282	308
Matched	210	225	187	215	212
Precision	0.9211	0.6285	0.7759	0.7624	0.6883
Recall	0.7749	0.8303	0.6900	0.7934	0.7823
F$_{\text{measure}}$	0.8417	0.7154	0.7305	0.7776	0.7323

In this paper, we found that the CB-SMoT algorithm is incapable of dealing with fake stops. For example, some moving points with a lower velocity, such as passing crossroads, are detected as stops by CB-SMoT. The main reason for this is that the CB-SMoT algorithm is velocity-dependent, and, thus, short slow-speed segments lead to fake stops. As for the DBSCAN algorithm, compared with our method, the main drawback is that it only considers spatial density information. Some intersections of a road, which are frequently visited by a user, would be detected as stops. The DJ-Cluster algorithm is a modification of the original DBSCAN method; however, it still only considers spatial information.

The time-based clustering method is sensitive to the time threshold. In this paper, we applied our new concept of move ability, instead of speed, to detect stops. This is an important reason why our method is more effective than the others.

In order to further verify the feasibility of our algorithm, and to reduce the impact of artificially labeled data on the experiment, a second experiment was done using our own dataset, which was collected by the authors. In detail, the dataset contained14 trajectory segments and the stops in each trajectory were recorded. As in the first experiment, after a number of experiments with different combinations of respective parameters, the optimal parameters used were selected. Finally, we matched the detected stops with the recorded stop information. The experimental results are shown in Table 3, and are in accordance with the experimental results of the labeled Geolife dataset. Our improved DBSCAN algorithm worked better than the other four algorithms for stop detection.

Generally, our method and the other four clustering algorithms can be used to detect stops in their respective scenarios. CB-SMoT is applicable to detect some very small stops, such as a short traffic jam, in case the fake stops are not critical. The trajectories of traffic jams, although having a relatively slow speed, are usually a linear motion. Our method would not detect these jams as being stops, as the move ability feature corresponding to these segments is rather large. As for the DBSCAN and DJ-Cluster algorithms, both methods only consider spatial information and can detect most locations with a higher density of trajectory points, for instance, positions visited by a subject frequently. The time-based clustering method is suitable for dealing with trajectories with an unstable recording frequency. Considering the spatial and temporal information, our algorithm works better for detecting activity stops that have a longer duration. Usually, users are only interested in these activity stops. In addition, our method is more robust to fake stops, such as traffic jams.

Table 3. Experimental results using data collected by the authors.

Metrics	Our Method	CB-SMoT	DBSCAN	DJ-Cluster	Time-Based
Parameters	$\sigma_1 = 0.5$ $Nap = 71$	$Eps = 50$ $MinTime = 5$	$Eps = 50$ $MinPts = 100$	$Eps = 50$ $MinPts = 100$	$Eps = 80$ $Time = 6$
Recorded	48	48	48	48	48
Detected	48	51	60	59	49
Matched	46	37	42	41	41
Precision	0.9583	0.7255	0.7000	0.6949	0.8367
Recall	0.9583	0.7292	0.8750	0.8542	0.8542
$F_{measure}$	0.9583	0.7475	0.7778	0.7664	0.8454

5.2. Evaluation of Efficiency

In this paper, we also performed an evaluation of the computational efficiency of our improved DBSCAN algorithm. We ran the algorithms, our method and the other four methods on a set of trajectories with a different number of trajectory points. The running time of this comparative experiment was recorded to evaluate the efficiency. Figure 10a shows the efficiency of the different methods. As we can see from Figure 10a, the curves corresponding to both the DBSCAN and DJ-Cluster algorithms increase sharply. This demonstrates that the DBSCAN and DJ-Cluster algorithms are not suitable for dealing with long trajectories. As for the other three methods, the computational efficiency of the algorithms is approximately linear. Moreover, the gap in the efficiency among these three methods is small enough to be negligible.

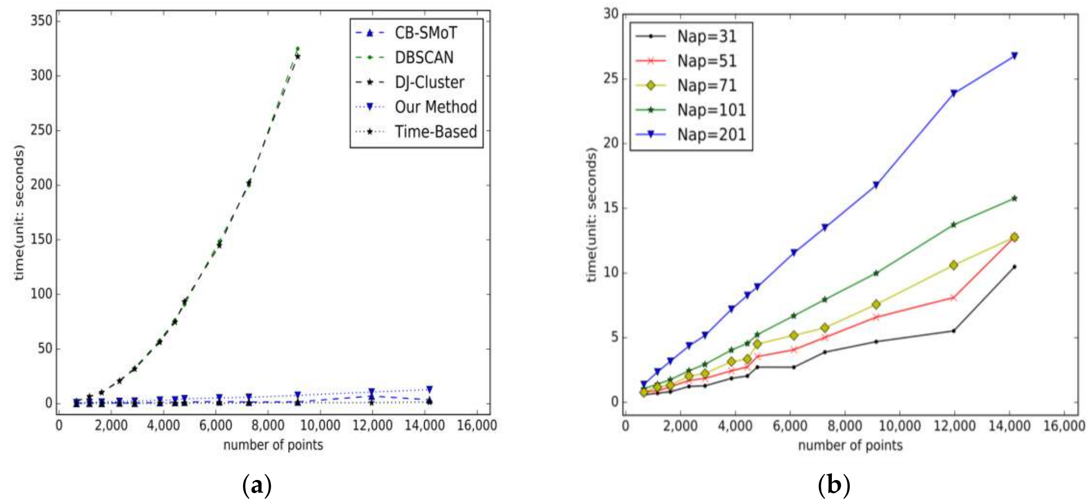


Figure 10. The efficiency evaluation. (**a**) Efficiency comparison; (**b**) effect of parameter *Nap* on the efficiency of our method.

Both our improved DBSCAN method and the CB-SMoT algorithm are derivative methods of the DBSCAN algorithm. The main difference among these derivative methods is how a core point is defined, leading to a difference in complexity. A distance calculation between all pairs of trajectory points is needed in the traditional DBSCAN method to select the core points; thus, the complexity of the DBSCAN method is $O(n^2)$. However, in our method, and in the CB-SMoT algorithm, the distance to only a part of the trajectory points is necessary to determine whether a point is a core point, leading to the complexity being $O(n)$. In more detail, the complexity of these algorithms is proportional to the number of times the distance between pairs of points is calculated. In our method, parameter *Nap* represents the number of adjacent points, and is related to the number of times the distance between pairs of points is calculated. Figure 10b shows the effect of parameter *Nap* on the computational efficiency. The results demonstrate that a larger value for *Nap* leads to a longer running time, which is consistent with our previous analyses.

6. Conclusions

In this paper, we first proposed the new concept of move ability. By introducing the theory of the data field in order to calculate the density of trajectory points, we proposed a new, comprehensive, hybrid feature–based, density measurement method to improve the original DBSCAN method. In our improved DBSCAN method, the move ability was taken into consideration by giving a lower move ability a larger weight. In addition, the density threshold can be automatically determined. In the experiments, we compared our method with four other clustering algorithms, CB-SMoT, DBSCAN, DJ-Cluster, and time-based clustering. The experimental results show that our method works better than the traditional methods, demonstrating the feasibility of our method.

Acknowledgments: This paper was supported by the National Natural Science Foundation of China (No.41501485).

Author Contributions: Ting Luo and Xinwei Zheng conceived and designed the experiments; Guangluan Xu performed the experiments; Kun Fu analyzed the data; Wenjuan Ren wrote the paper.

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

References

1. Spaccapietra, S.; Parent, C.; Damiani, M.L.; Macedo, J.A.D.; Porto, F.; Vangenot, C. A conceptual view on trajectories. *Data Knowl. Eng.* **2008**, *65*, 126–146. [CrossRef]
2. Alvares, L.O.; Bogorny, V.; Kuijpers, B.; de Macedo, J.A.F.; 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; p. 22.
3. Xie, K.; Deng, K.; Zhou, X. From trajectories to activities: A spatio-temporal join approach. In Proceedings of the 2009 ACM International Workshop on Location Based Social Networks, Seattle, WA, USA, 03 November 2009; pp. 25–32.
4. Li, D.; Du, Y. *Artificial Intelligence with Uncertainty*; CRC Press: Boca Raton, FL, USA, 2007.
5. Hwang, S.; Evans, C.; Hanke, T. Detecting stop episodes from GPS trajectories with gaps. In *Seeing Cities through Big Data*; Springer: Cham, Switzerland, 2017; pp. 427–439.
6. 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, 16–20 March 2008; pp. 863–868.
7. Mousavi, S.M.; Harwood, A.; Karunasekera, S.; Maghrebi, M. Geometry of interest (GOI): Spatio-temporal destination extraction and partitioning in GPS trajectory data. *J. Ambient Intell. Humaniz. Comput.* **2016**, 1–16. [CrossRef]
8. Bhattacharya, T.; Kulik, L.; Bailey, J. Automatically recognizing places of interest from unreliable GPS data using spatio-temporal density estimation and line intersections. *Pervasive Mob. Comput.* **2015**, *19*, 86–107. [CrossRef]
9. Kami, N.; Enomoto, N.; Baba, T.; Yoshikawa, T. Algorithm for detecting significant locations from raw GPS data. In Proceedings of the 2010 International Conference on Discovery Science, Canberra, Australia, 6–8 October 2010; pp. 221–235.
10. Cao, X.; Cong, G.; Jensen, C.S. Mining significant semantic locations from GPS data. *Proc. VLDB Endow.* **2010**, *3*, 1009–1020. [CrossRef]
11. Ashbrook, D.; Starner, T. Using gps to learn significant locations and predict movement across multiple users. *Pers. Ubiquitous Comput.* **2003**, *7*, 275–286. [CrossRef]
12. Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*; KDD: Portland, OR, USA, 1996; pp. 226–231.
13. Schoier, G.; Borruso, G. Individual movements and geographical data mining. Clustering algorithms for highlighting hotspots in personal navigation routes. In Proceedings of the International Conference on Computational Science and ITS Applications, Santander, Spain, 20–23 June 2011; pp. 454–465.
14. Zhou, C.; Frankowski, D.; Ludford, P.; Shekhar, S.; Terveen, L. Discovering personally meaningful places: An interactive clustering approach. *ACM Trans. Inf. Syst.* **2007**, *25*, 12. [CrossRef]
15. Zhou, C.; Frankowski, D.; Ludford, P.; Shekhar, S.; Terveen, L. Discovering personal gazetteers: An interactive clustering approach. In Proceedings of the ACM International Workshop on Geographic Information Systems, Washington, DC, USA, 8–13 November 2004; pp. 266–273.
16. Zhao, X.-L.; Xu, W.-X. A clustering-based approach for discovering interesting places in a single trajectory. In Proceedings of the 2009 IEEE Second International Conference on Intelligent Computation Technology and Automation, ICICTA 2009, Zhangjiajie, China, 10–11 October 2009; pp. 429–432.
17. Rocha, J.A.M.; Times, V.C.; Oliveira, G.; Alvares, L.O.; Bogorny, V. Db-smot: A direction-based spatio-temporal clustering method. In Proceedings of the 2010 5th IEEE International Conference Intelligent Systems, London, UK, 7–9 July 2010; pp. 114–119.
18. Lv, M.; Chen, L.; Xu, Z.; Li, Y.; Chen, G. The discovery of personally semantic places based on trajectory data mining. *Neurocomputing* **2016**, *173*, 1142–1153. [CrossRef]
19. Yan, Z.; Parent, C.; Spaccapietra, S.; Chakraborty, D. A hybrid model and computing platform for spatio-semantic trajectories. In Proceedings of the International Conference on the Semantic Web: Research and Applications, Heraklion, Greece, 30 May–2 June 2010; pp. 60–75.
20. Zheng, Y.; Zhang, L.; Xie, X.; Ma, W.Y. Mining interesting locations and travel sequences from gps trajectories. In Proceedings of the International Conference on World Wide Web, WWW 2009, Madrid, Spain, 20–24 April 2009; pp. 791–800.

21. Fu, Z.; Tian, Z.; Xu, Y.; Qiao, C. A two-step clustering approach to extract locations from individual GPS trajectory data. *ISPRS Int. J. Geo-Inf.* **2016**, *5*, 166. [CrossRef]
22. Yuan, Y.; Raubal, M. Measuring similarity of mobile phone user trajectories—A spatio-temporal edit distance method. *Int. J. Geogr. Inf. Sci.* **2014**, *28*, 496–520. [CrossRef]
23. Chen, S.; Meng, H.; Zhang, C.; Liu, C. A KD curvature based corner detector. *Neurocomputing* **2016**, *173*, 434–441. [CrossRef]
24. Guidotti, R.; Trasarti, R.; Nanni, M. TOSCA: Two-steps clustering algorithm for personal locations detection. In Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 3–6 November 2015.
25. Tran, L.H.; Dang, T.K.; Thoai, N. Hybrid stop discovery in trajectory records. In Proceedings of the 24th International Workshop on Database and Expert Systems Applications (DEXA), Prague, Czech Republic, 26–30 August 2013.