



國立台灣大學電機資訊學院電機工程學系

博士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Doctoral Dissertation

以軌跡資料剖析與倉儲進行物體移動行為分析

Trajectory Data Profiling and Warehousing

for Behavior Analysis of Moving Objects

吳蕙如

Huey-Ru Wu

指導教授：陳銘憲 博士

Advisor: Ming-Syan Chen, Ph.D.

共同指導教授：葉彌妍 博士

Co-advisor: Mi-Yen Yeh, Ph.D.

中華民國 102 年 8 月

August, 2013





Trajectory Data Profiling and Warehousing for Behavior Analysis of Moving Objects

Huey-Ru Wu

Advisor: Ming-Syan Chen, Ph.D.

Co-Advisor: Mi-Yen Yeh, Ph.D.

Department of Electrical Engineering

National Taiwan University

Taipei, Taiwan, R.O.C.

August 2013





Acknowledgement

I would like to take this opportunity to acknowledge all those people who have provided me assistance throughout my doctoral work in these years.

First of all, I would like to express my gratitude to my advisor, Dr. Ming-Syan Chen. Though concerned, he has allowed me to explore data mining and database field in my own pace and has given me great freedom in choosing my research topic. Meanwhile, he offers me timely suggestions when I faltered and wondered about my next step, that always helps me find a way to keep on going.

Next, I am particularly grateful for the patient guidance and persistent help given by my co-advisor, Dr. Mi-Yen Yeh. She spends time to discuss with me about my developing ideas, helps me put my pieces of thoughts and parts of designs together, and gives me constructive opinions as I tried to organize my work. Without the support and guidance from my advisors, I could not complete my doctoral work.

In addition, I would like to thank the members of my Ph.D. defense committee: Dr. Meng-Chang Chen, Dr. Chih-Wei Yi, Dr. Shou-De Lin, and Dr. Kun-Ta Chuang. They have taken effort in giving me valuable comments and suggestions on my earlier version of this dissertation.

I also want to thank my labmates in Network Database Laboratory in the Department of Electrical Engineering, National Taiwan University. As a part-time student, I rely a lot on their help and information sharing to keep up with the school matters along these years. Special thanks go to Dr. Chih-Hua Tai, for her frequent kindness experience providing, and Ms. Yu-Fen Chen, for her assistance in many school administrative procedures.

Thing would not have been possible without the support from the managers in my office, the Meteorological Information Center in Central Weather Bureau. I would like to offer my appreciations to Director Shen and Director Cheng, they gave

me permission and let me start my doctoral study while keeping my full time job. And to Chief Huang and Chief Chang, they provide a lot of cares about my progress along all these years.

I would like to extend my sincere thanks to my colleagues at work in Numerical Weather Prediction Section. They help to cover those occasionally happened problems or temporary assignments of our application systems, when I left for school. And they encouraged me to continue my doctoral work when I felt really exhausted on trying to deal with both my office and school works at the same time.

Finally, yet importantly, I would like to express my deepest thanks and appreciations to my beloved family members, especially to my parents. It is their unconditional love and undoubtedly confidence in me that makes me continue on going and complete my doctoral work. I would never have reached this goal without their endless support. Thank you, I will be grateful forever for your love.

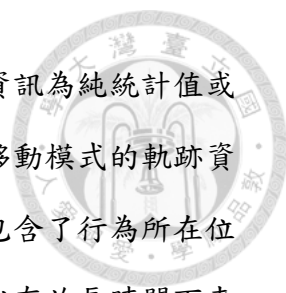


摘要

近年來，發展出許多具有定位功能的設備，可以被運用於追蹤各種物體的移動情況。這類工具蒐集到的大量而連續的位置及時間資料，足以描繪出各物體的移動軌跡。這些物體移動軌跡資料中藏有豐富的移動行為訊息，值得進一步處理並嘗試解析。在這篇論文中，我們鎖定以資料剖析與資料倉儲方式，由物體軌跡資料中分析出他們的移動行為。

在物體移動分布的大範圍中，不同的小區域因其地理特性與時空條件，分別呈現出物體於該區域中的移動行為。並且，在分析各群組物體移動路線的同時，我們也同時關心其中的移動速度差異。於是，我們的目標訂在，由一組分布於大範圍的物體移動軌跡資料中，找出這些物體在各小區域的典型移動行為。所找到的區域典型移動行為，形成了對這組軌跡資料的剖析結果，可反映出該類物體的移動型態與各區域的地理資訊。然而，物體移動當中，時有速度變化也常有方向的改換，以致多個物體的共同行為不易由其移動軌跡中分析出。我們提出的 DivCluST 方法，同時考量了時間與空間的特性後，將軌跡資料進行切割與分群處理，並從中得出資料剖析的結果。不同於只處理軌跡的空間特性，或只分析特定小範圍區域的其他既存方法，DivCluST 可用於分析大區域範圍，並且在其切割的步驟中，限制了各軌跡段落時間上與空間上的變化，而在以 k-means 為基礎的分群方法中，也同步量測兩軌跡段落時間與空間距離。我們以數個真實世界中的軌跡資料集，對所提出的方法進行了一系列的測試，並且設計將結果視覺化的方式，以印證 DivCluST 可以有效的剖析物體移動軌跡。

在大量的物體移動軌跡被連續不斷收集的情況下，我們需要設計合宜的資料結構，來長期存放移動行為相關資訊，以幫助訊息分析，並作進一步的處理與應用。軌跡資料倉儲，可以有效率的濃縮摘要自物體軌跡中萃取出移動模式，並



且可以提供模式查詢，幫助事件分析與進行決策。不同於轉換資訊為純統計值或只專注在特定小區域的其他既存方法，我們設計了儲存各區域移動模式的軌跡資料倉儲。首先是提出適合表現移動行為的資料存放格式，其中包含了行為所在位置、移動方向、前進速度等。我們也規劃了完整的倉儲架構，以存放長時間下來分析出的移動模式與其他相關資訊。我們使用兩階段的演算法，用以自軌跡資料中萃取出移動模式，首先以在線處理方式摘要分布於廣大區域的定位資訊，然後以多維度網格為基礎分析出移動模式。以不同精細度的單位網格為準設計的模式運算工具，可以結合倉儲中的移動模式與其他資訊，回應各種可能的查詢需求與幫助資料的長期維護。同樣的，我們實作了所設計的物體軌跡資料倉儲，並以數個真實世界中的軌跡資料集，驗證了所提方法能有效且快速的進行物體移動行為分析、模式儲存、也回應相關的資訊查詢。

關鍵字：物體軌跡、移動模式、軌跡分割、行為剖析、軌跡倉儲



Abstract

Nowadays, devices attached with position detecting techniques are used on many places to track moving of objects. The collected time and position records, which constructed moving trajectories of objects, are in huge amount. Among the object trajectories, interesting moving behaviors are hidden and worth to be revealed through some processing. In this dissertation, we focus on analyzing object moving behaviors through trajectory data profiling and warehousing.

In an area where a set of objects moving around, there are some typical moving behaviors of objects at different regions in respect to the geographical nature or other spatiotemporal conditions. Not only paths that objects moving along, we also want to know how different groups of objects move with various speeds. Therefore, given a set of collected trajectories spreading in a bounded area, we are interested in discovering typical moving styles in different regions of all monitored moving objects. These regional typical moving styles are regarded as profile of the monitored moving objects, which may help reflect the geographical information of observed area and the moving behaviors of observed moving objects. However, an object can move with various speeds and arbitrarily changing directions. The changes cause difficulty in analyzing behaviors among object trajectories. Thus, we present DivCluST, an approach to finding regional typical moving styles by dividing and clustering trajectories in consideration of both spatial and temporal constraints. Different from existing works that considered only spatial properties or just some interesting regions of trajectories, DivCluST focuses more on finding typical regional spatiotemporal behaviors over a large area. It takes both spatial and temporal information into account when designing the criteria for trajectory dividing and the distance measurement for adaptive k -means clustering. Extensive experiments on three types of real data sets with specially designed visualization are presented to

show the effectiveness of DivCluST.

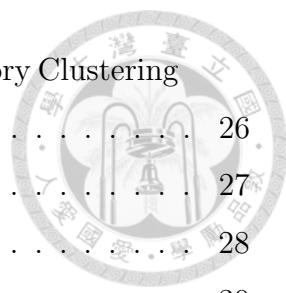
With huge amount of object moving trajectories collected continuously and boundlessly, we need a well designed data structure to analyze trajectory data and keep moving behavior information for further processes and applications. A trajectory data warehouse is an effective way to store organized moving patterns extracted from object trajectories, and can offer efficient information queries for event analysis and decision making. Different from existing works that stored only statistic values of trajectories or focused only on limited number of selected regions, we present a trajectory data warehouse storing moving patterns spreading in all areas. We design a proper data format for moving patterns to represent typical behaviors, containing main properties of trajectories, such as laying positions, moving directions and forward speeds. Also, we propose a corresponding table schema for keeping long-term moving patterns in our trajectory warehouse. A two-stage algorithm is proposed to online process the incoming trajectory data over a large area and extract the moving patterns from them based on multidimensional unit grids. Operations on moving patterns and related tables, such as spatial position relation and aggregation, based on multiple granularity of grids, are provided for flexible query requirements and warehouse maintenance. Experiments on real-world trajectory data sets show that our designs on storage and operations of trajectory patterns make our trajectory data warehousing effective and efficient for moving pattern analysis.

Keywords: object trajectory, moving pattern, trajectory dividing, behavior profiling, trajectory warehousing



Contents

Acknowledgement	iii
Abstract	v
Contents	ix
List of Figures	xi
List of Tables	xiii
List of Algorithms	xiv
1 Introduction	1
1.1 Motivation	1
1.2 Overview of the Dissertation	3
1.3 Organization of the Dissertation	6
2 Profiling Moving Objects by Dividing and Clustering Trajectories Spatiotemporally	7
2.1 Introduction	7
2.2 Related Works	11
2.3 Problem Statement	13
2.4 DivST: Spatiotemporal Trajectory Dividing	15
2.4.1 Spatiotemporal Trajectory Dividing Algorithm	16
2.4.2 Selection of Thresholds in DivST	19
2.5 CluST: Spatiotemporal Replacement Line Clustering	20
2.5.1 Spatiotemporal Line Distance	20
2.5.2 k -means Based Line Clustering Algorithm	24



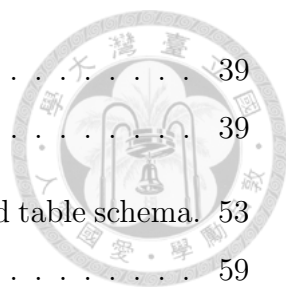
2.5.3	Profiling Moving Objects by Interpreting Trajectory Clustering Results	26
2.6	Experiment Results	27
2.6.1	Three Sets of Real Trajectory	28
2.6.2	Profiling Results of DivCluST	30
2.6.3	Comparisons with Spatial-only and Temporal-only Profiling	32
2.6.4	Analysis of DivST	34
2.6.5	Analysis of CluST	37
2.7	Summary	41
3	Trajectory Warehousing for Multi-Granularity Moving Pattern Analysis	43
3.1	Introduction	43
3.2	Related Works	47
3.3	Preliminaries	50
3.4	Extracting Moving Patterns	54
3.4.1	Trajectory Consistency Dividing	54
3.4.2	Group Pattern Generating	58
3.5	Warehouse Operations and Queries	62
3.5.1	Spatial and Temporal Operations	63
3.5.2	Aggregation and Warehouse Maintenance	65
3.5.3	Distinct Trajectory Estimation	67
3.5.4	Moving Pattern Queries	69
3.6	Experiments and Evaluations	71
3.6.1	Trajectory Sets and Pattern Extracted	71
3.6.2	Space and Time Analysis	74
3.6.3	Operation Analysis	76
3.7	Summary	77
4	Conclusion and Future Work	79
4.1	Conclusion	79
4.2	Future Work	80
	Bibliography	81



List of Figures

1.1	Observations about object moving behaviors and trajectories.	2
2.1	Illustration of DivCluST.	8
2.2	Output compared with main references.	10
2.3	Illustrations of data points and lines.	14
2.4	Illustrations of element lines L_1, L_2 passing the dividing criteria th_{len} and th_{spd} and being replaced by L_R	19
2.5	Illustrations of the two parts of shift distance.	21
2.6	Illustrations of equal shift distance.	21
2.7	Condition changes vary shift distances.	22
2.8	Zero shift distance cases: (a) $d_{dp} = 0$, (b) $d_{al} = 0$	23
2.9	Cases of zero speed distances: $s_{L_a} = s_{L_b}$	23
2.10	Illustration of mean line computation in <i>CluST</i>	25
2.11	Information about real data sets.	28
2.12	Profiling results on typhoon track and corresponding spatial-only and temporal-only comparisons.	29
2.13	Profiling results on bus trajectory and corresponding spatial-only and temporal-only comparisons.	31
2.14	Profiling results on taxi trajectory data	32
2.15	The <i>DivST</i> parameter selection analysis.	34
2.16	Impacts of th_{len} for typhoon tracks profiling.	35
2.17	Impacts of th_{spd} for typhoon tracks profiling.	35
2.18	Comparison of DivST and the approximate MDL.	37
2.19	<i>CluST</i> results with too large k	37
2.20	<i>CluST</i> results with too small k	38
2.21	Impacts of w_{sht} and w_{spd} on bus data profiling.	38

2.22	DBSCAN parameters analysis on typhoon data.	39
2.23	DBSCAN representation lines on typhoon data.	39
3.1	Process of building trajectory data warehouse and related table schema.	53
3.2	Illustration of multidimensional grids and granularity.	59
3.3	Example of information loss in grid grouping.	62
3.4	Illustration of spatial domain operations.	65
3.5	Illustration of significant patterns on selected area.	72
3.6	Time spent in extracting moving patterns.	76
3.7	Results of pattern related operation on selected grid.	77
3.8	Analysis of pattern related operation.	78





List of Tables

2.1	<i>DivCluST</i> : List of notations	16
3.1	Summarizing information amount in pattern extraction processes . . .	75



List of Algorithms

2.1	DivST: Spatiotemporal Trajectory Dividing	17
2.2	CluST: Spatiotemporal Line Clustering	25
3.1	Online trajectory consistency dividing	57
3.2	Multidimensional grid based moving pattern generating	59
3.3	Trajectory flow amount trace	68



Chapter 1

Introduction

1.1 Motivation

Tracking movements of objects, such as human activities, animal migrations or vehicle travelings, and learning about their behaviors interest people a lot. With the development of position tracking techniques and the availability of related devices, position records and their corresponding time instants of objects can be collected easily. The continuously collected records of an object construct the object's moving trajectory. As huge amount of collected object moving records are available, many studies are proposed to discover interesting information from object moving trajectories in recent years. The topics include trajectory data clearing and preprocessing [1] [2], trajectory data indexing and information retrieval [3] [4], similarity measurement between trajectories [5], trajectory positioning quality and uncertainty problem [6] [7], privacy preservation of trajectories [8] [9] [10], distinguishing special activities from trajectories [11], analyzing specific types of trajectories [12] [13], providing method for location-based services [14] [15], linking trajectories with geographical data such as road or topographic maps [16], or visualizing trajectory related information [17] [18].

Among various kinds of research topics related to trajectory data, we are specially interested in analyzing moving behaviors of objects. A trajectory has both spatial and temporal information, which describe moving paths and directions, and spent times and speeds, respectively. We can infer the moving behaviors of objects from these spatial and temporal information. From object moving trajectories, we observe

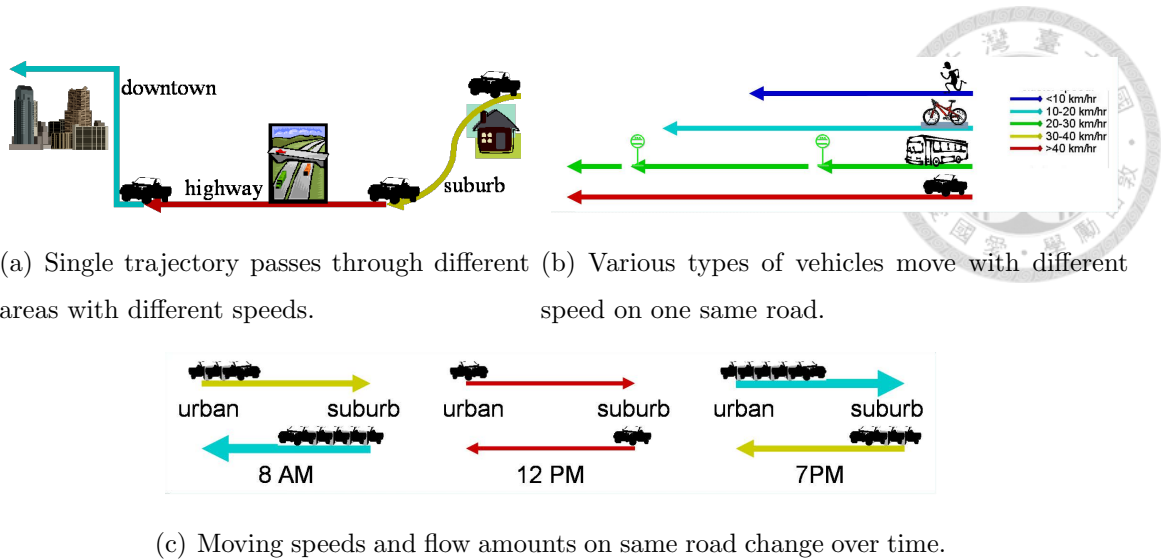
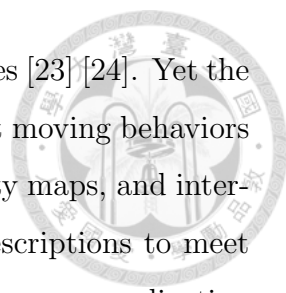


Figure 1.1: Observations about object moving behaviors and trajectories.

that an object often moves in changing directions and speeds along a route. For instance, a car would pass through different areas in a long route with changing directions and different speeds as shown in Fig. 1.1(a). As a result, we want to find out the changes of behaviors among different areas. We also observed that different types of objects move differently on one same route. For example, various types of vehicles may move with different speeds on a section of road as Fig. 1.1(b). So, we should try to distinguish the difference of moving behaviors in an area via analysis. Furthermore, we also noticed that even having same type of objects moving on one same path, the moving speeds would be quite variant in different time periods. Say, the moving speeds and flow amounts on one same road would be really different between rush and off hours, and between workday mornings and evenings, as shown in Fig. 1.1(c). Thus, trajectory analysis should also consider about the dynamic moving behavior changes with time.

To reveal moving behaviors from trajectory data, there are several existing methods. One method grouped similar parts of object moving paths, then swept included members and constructed representation as a single new trajectory [19]. But various regional behaviors vanished during construction of one main trajectory. The other focused on predefined interesting regions, and analyzed only about trajectory behaviors over those limited regions [20] [21] [22]. However, we do not always know enough about trajectory data and corresponding area to assign proper interesting regions. Another type of works transformed trajectories to records in cells



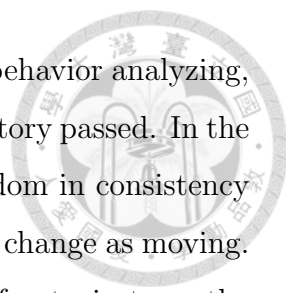
they passed, and summarized behavior properties as statistic values [23] [24]. Yet the statistics, such as summation or average, in cells cannot represent moving behaviors well. Still others mapped trajectories to semantic data such as city maps, and interpreted the movings into real-world locations or else semantic descriptions to meet predefined application requirements [25] [26]. The interpretations were application dependent and limited other possibilities of further usage though.

We would like to reveal typical moving behaviors over all regions of the large area where object trajectories spread, even without previous knowledge of the data and the corresponding area. With a sketch of behaviors, we can understand general moving styles of objects being monitored, such as animal migration in Autumn, or downtown traffic during rush hours. In the mean time, an object almost never moves in one single direction or with a steady speed through a long path, and the corresponding moving trajectory is almost always in an irregular shape. With many irregular shaped trajectories in a data set, though we want to find some common moving behaviors among them, we can imagine the results of sketching all trajectories in a figure would be a chaos and hardly any moving patterns can be revealed. Moreover, as a trajectory is usually composed of many sampling positions and the corresponding time instances, the amount of data to be processed is huge.

In this dissertation, we analyze object moving behaviors through trajectory data profiling and warehousing. A trajectory profiling takes some algorithms or mathematical techniques to discover patterns or correlations from large quantities of data. We would like to use profiling to sketch the moving tendency and to understand the moving behaviors of a set objects. After understanding moving behaviors through profiling, we choose to warehouse the trajectory data for further analysis on moving behaviors. Moving behaviors among trajectories from different time periods would be extracted and kept as patterns in a long term. And regular formatted moving patterns in a trajectory data warehouse can be used in various analytic processes.

1.2 Overview of the Dissertation

In moving behavior analysis, we care about not only the path an object moving through, but also the time spent and the speed of its moving. Temporal information

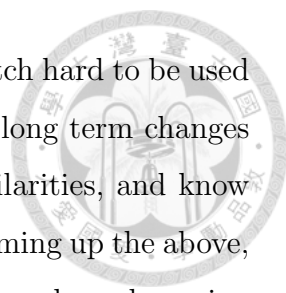


like speed and time should be taken into account during moving behavior analyzing, as well as spatial information such as areas where the object trajectory passed. In the meantime, when an object moving through a long path, it is seldom in consistency behavior from its start to end, both its direction and speed would change as moving. Since moving behaviors of an object varies in different parts of a trajectory, the behaviors should be analyzed separately. We would like to use profiling to sketch and to understand typical moving behaviors among trajectories, while considering in spatiotemporal information and analyzing moving of different trajectory parts.

Given a group of object moving trajectories in an area, and we know little about the characteristics of trajectory data and the geographical conditions of their corresponding spread area. We would like to analyze trajectories spread in large area while revealing typical regional moving behaviors of the objects at the same time. Therefore, we want to find typical moving behaviors through clustering similar trajectories. Before clustering can work, we have to measure the distance between any two trajectories first. Furthermore, in trajectory distance measurement, we have to deal with irregular shapes and changing speeds of trajectories.

Thus, in Chapter 2, we propose DivCluST, an approach to automatically reporting regionally typical moving behaviors over a wide bounded area. Both the spatial and temporal properties of moving behaviors are considered during the process. First, the non-uniform movement of a trajectory is dealt through dividing it into uniformly behavior sub-trajectories, by examining the consistency of properties on both spatial and temporal domains. Next, the typical moving behaviors are extracted through clustering methods, while similarities between sub-trajectories are measured spatiotemporally, and those ones with similar behaviors are clustered together. By visualizing the cluster representations, we effectively present the profile of given moving object trajectories. The moving behavior profile of an object type can help understand moving tendency of this kind of objects. Also, the profile provides a signature as typical moving of this type of object for further applications.

While the DivCluST profiles moving behaviors of object trajectories through spatiotemporally dividing and clustering, it does not consider about changes of moving behaviors in different time periods. Furthermore, while clustering methods are effective in revealing similar behaviors among trajectory divisions, the arbitrarily



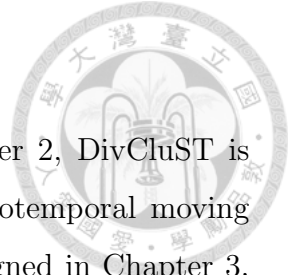
cluster sizes and distributing areas make the moving behavior sketch hard to be used in further processes and comparisons. We would like to analyze long term changes of object moving behaviors, compare their similarities or dissimilarities, and know about representing value ranges of each extracted behaviors. Summing up the above, a trajectory data warehouse is a proper choice for keeping new and aged moving patterns extracted from trajectories in a long term, and for providing operations for behavior analytic processes.

To build a trajectory data warehouse, we first have to decide the format of moving patterns being stored. The format should be able to reflect object moving behaviors, and allow pattern comparisons between different time periods. Next, the data amount kept after extracting moving behaviors from large number of trajectory points must be condensed, while the information loss ought to be constrained as well. Finally, some special operations designed for moving patterns are required, in order to help information retrieval from the warehouse and long term pattern maintenance.

Therefore, in Chapter 3, we store extracted moving patterns into a data warehouse, which can summarize trajectory behavior information, keep new and aged moving patterns, and meet the need of information retrievals, pattern comparisons or further application requirements. The analytic process works on all area where trajectory records spread, and then can provide a snapshot of moving behaviors in a selected region, or can give a long term condition sketch of a spot. With large amount of position and time records arriving and awaiting for moving pattern extraction, an online algorithm is used for processing and condensing the information. Meanwhile, to provide comparisons and further operations for moving behaviors, patterns are generated based on multidimensional unit grids, which regularized the representing value ranges of pattern attributes. Afterward, pattern operations with different granularity grids on spatiotemporal domains can help information queries and application usages, and also help pattern summarizations. With these operations, moving behavior template of an area or frequent moving style trend of a spot can be found. Further more, with link to other spatial, temporal or specific domain information in the data warehouse, various possibilities of analytic processes are then provided.

1.3 Organization of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, DivCluST is proposed to profile trajectory data, and to reveal typical spatiotemporal moving behaviors of objects. Then, a trajectory data warehouse is designed in Chapter 3, to extract moving patterns from trajectories and to provide operations for object behavior analysis. In the final, works in this dissertation are concluded and possible future directions are discussed in Chapter 4.





Chapter 2

Profiling Moving Objects by Dividing and Clustering Trajectories Spatiotemporally

2.1 Introduction

With the common use of tracking and positioning devices and the advance of mobile communication technologies, a huge amount of trajectory data can be collected almost anytime, anywhere. These trajectories are usually recorded with both positions and time-stamps, which denote the moving paths and speeds of objects being tracked. For example, movement of different types of vehicles can be recorded with the equipped GPS devices, travel paths of migrating animals are recorded by transmitters worn on them, and tracks of typhoons are monitored by meteorological satellites.

An object can move with various speeds and arbitrarily changing directions. Given a bounded area where a set of objects moving around, there are some typical moving styles of the objects at different local regions due to the geography nature or other spatiotemporal conditions. Not only the paths that the objects move along, we also want to know how different groups of objects move with various speeds. For example, typhoons usually move faster after they develop over the tropical oceans in low altitude regions but slow down when moving poleward or disappear when hitting lands. In the main roads of a big city, a huge volume of vehicles

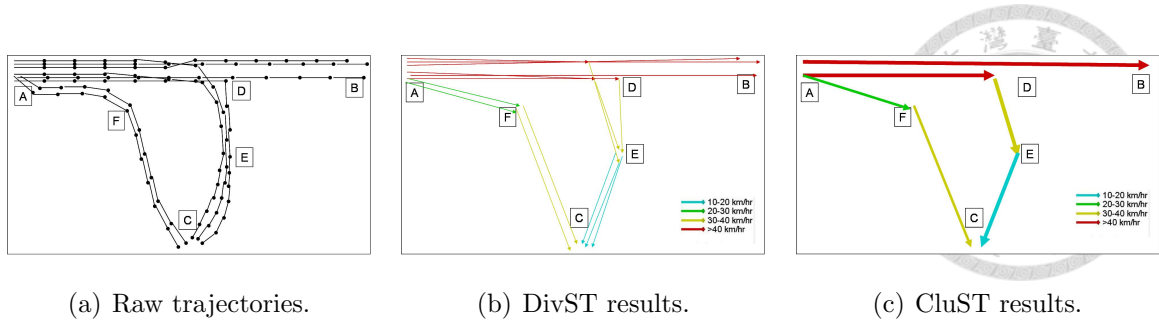
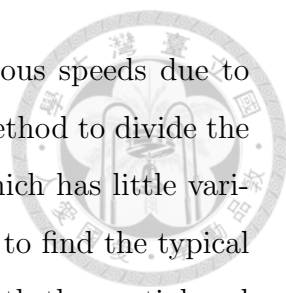


Figure 2.1: Illustration of DivCluST.

pass through them bidirectionally in a low speed during rush hours while only few vehicles will pass through it fast in the midnight. Therefore, given a set of collected trajectories spreading in a bounded area, we are interested in discovering the typical moving styles in different regions of all the monitored moving objects. These regional typical moving styles are regarded as the *profile* of the monitored moving objects, which may help reflect the geographical information of the observed area and the moving behaviors of the observed moving objects. For example, the profile of vehicles moving in the downtown area of a city can be used to detect the road design flaws in urban planning. The profile of animal migration can help us realize not only the main walkable paths but also their habitual behaviors in different geographical conditions. In this chapter, motivated by this need, we want to develop an automatic approach to profiling the given set of moving objects by analyzing their trajectories, namely to discover the profile of the regional typical moving styles of the moving objects.

Wanting to know the intrinsic moving behaviors of the moving objects without any prior knowledge about the geographic information of the area they moved around, we need to deal with two main challenges. First, how should we extract each group of moving objects that having the similar moving styles? This can be solved by applying a clustering technique on the trajectories since it groups similar ones while separating dissimilar ones. By identifying clusters with its representatives, we can discover different moving styles while excluding the redundant information. As the speed change, length of the moving path, and the directions are equally important, we need to design a distance measurement that considers all these factors simultaneously, which is absolutely not a trivial process. Second, an important characteristic of a trajectory is that its movement is not uniform in both the spatial and temporal



domains. For example, one can drive along one path with various speeds due to different road conditions or speed limits. Therefore, we need a method to divide the original trajectories into segments of sub-trajectories, each of which has little variations in both speeds and directions, before we can cluster them to find the typical moving styles. The design of dividing criteria should consider both the spatial and temporal information, while keeping the number of segments low.

To tackle the above problems, we develop *DivCluST*, an approach to finding regional typical moving styles by dividing and clustering the trajectories in consideration of both the spatial and temporal constraints. We use an example in Fig. 2.1 to illustrate our idea, where the set of raw trajectories is given in Fig. 2.1(a). In essence, *DivCluST* is comprised of two phases: the *DivST* phase and the *CluST* phase. In the *DivST* phase, the trajectories are divided into segments according to both the given spatial and temporal criteria by a sliding window based method, such that each divided sub-trajectory has little speed and direction variations and can be represented by a replacement line. As shown in Fig. 2.1(b), different colors represent different average speed of the replacement lines. Meanwhile, we keep the number of segments as small as possible. The online dividing algorithm can deal with a large number of trajectories streaming in with limited memory. In the *CluST* phase, where a k -means based algorithm with a newly designed spatiotemporal distance measurement for directional lines is applied. Through visualizing the cluster representatives, each of which is also a line representing the overall position, speed (color), direction (arrow), and member count of that cluster (line width), we can have a profile that shows the regional moving behaviors of the area spanned by the given trajectory set, as shown in Fig. 2.1(c).

Our profiling idea of finding typical moving styles is novel and different from the main related research works as follows. First, most existing works of sub-trajectory clustering such as *TraClus* [19] consider only the spatial information of trajectories, i.e., the physical position and the shape of trajectories, on both trajectory segmenting and clustering. Also, the goal in [19] is to find one representative trajectory describing the overall movement of each cluster found, where only paths with significantly different directions will be revealed but some regional moving styles were omitted. For example, on the same set of trajectories given in Fig. 2.1(a), we

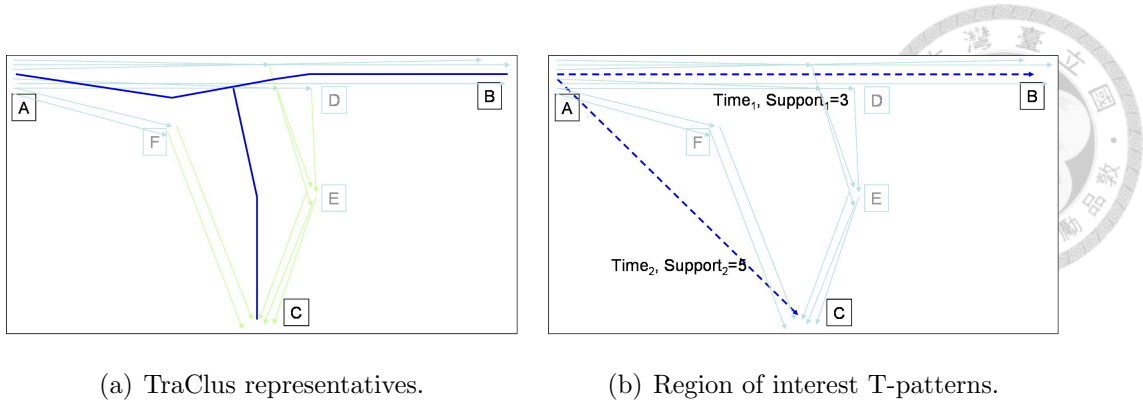
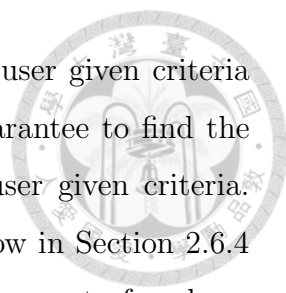


Figure 2.2: Output compared with main references.

simulated the TraClus algorithm and illustrated the results in Fig. 2.2(a). Only two clusters were generated by DBSCAN, of which the two representative trajectories were found as shown in blue lines. We can see that two clusters of objects moving shorter along the $A \rightarrow B$ path, such as $A \rightarrow F$ and $A \rightarrow D$, were missing. Moreover, the speed differences between $D \rightarrow E$ and $E \rightarrow C$ or between $F \rightarrow C$ and $E \rightarrow C$ were ignored. On the other hand, works such as [27] [20] only care the trajectories or paths between the identified interesting locations. As illustrated in Fig. 2.2(b), suppose only A , B and C were recognized as the interesting locations, the main turning points in location D would be omitted. Also, the two paths $A \rightarrow D \rightarrow C$ and $A \rightarrow F \rightarrow C$ would not be distinguished as they spent similar time to pass.

To the best of our knowledge, we are the first to tackle the trajectory profiling problem of a given set of moving objects, considering both the temporal and spatial changes simultaneously. We make several contributions as follows.

- We propose a new way to look the collected trajectories and design DivCluST, an approach to automatically reporting the regional typical moving styles over a wide bounded area. By visualizing the cluster representative associated with the physical location, speed, direction, and cluster size information, we effectively present the profile of the given trajectories of the monitored objects.
- The profiling results on three real data sets can be found in Section 2.6.2. Compared with the conditions that consider only the spatial or temporal information of trajectories, we demonstrate the out-performance of DivCluST in revealing the moving behaviors in Section 2.6.3.
- We design DivST to divide each trajectory into segments, each of which has



variations in both speeds and directions controlled to the user given criteria and is replaced by a replacement line. Meanwhile, we guarantee to find the minimum number of those replacement lines under the user given criteria. Compared with the approximate MDL used in [19], we show in Section 2.6.4 that DivST has better processing efficiency when similar amount of replacements are produced.

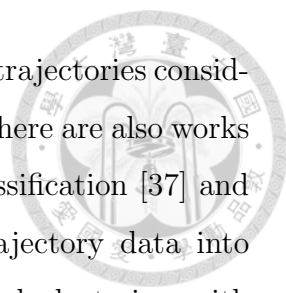
- We design CluST, a k -means based clustering method with an adaptive spatiotemporal distance function for two lines. In addition, we propose how to generate a representative of a cluster, which is a directional mean line of all line segments in that cluster. The analysis of the k selection, impact of different weights on measuring the spatial and temporal distances, and the better effectiveness and efficiency of CluST compared with DBSCAN are given in Section 2.6.5.

The remainder of this chapter is organized as follows. We discuss the related work in Section 2.2. We introduce the problem statements and preliminaries in Section 2.3. Our proposed approaches to dividing and clustering trajectories are in Section 2.4 and 2.5, respectively. Section 2.6 presents the experiment results on real data sets. We conclude our work in Section 2.7.

2.2 Related Works

Many traditional clustering methods are designed for spatially distributed data [28]. However, most of them are hard to apply on trajectory data directly due to the uneven spatiotemporal characteristics along a trajectory. Non-trivial transformations or newly designed methods are often needed to process trajectory data [1]. For example, previous works process the trajectories by the regression and probabilistic methods [29], the hidden Markov model [30], or the polynomial approximation [31]. Instead of building models for trajectories in advance, other works handle trajectory in a piecewise fashion with the hierarchical [32], indexed based [33], or distance based methods [34]. All these works basically focus on clustering the whole trajectories.

In contrast to considering the whole trajectories, more and more works focus on the characteristics of partial trajectories. For example, there are works detecting region-of-interest from paths of moving objects [27] [35], and finding frequent patterns

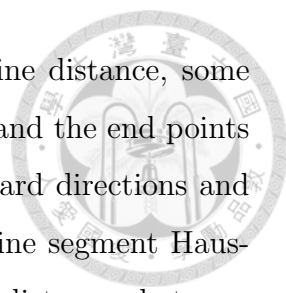


[20] by defining popular points among trajectories. [36] calibrated trajectories considering their spatial relationships with pre-defined anchor points. There are also works considering partial trajectory patterns on traffic networks in classification [37] and pattern matching [38]. These works take only part of the trajectory data into analyzing. In [39] and [40], the authors considered hierarchical clustering with partial trajectory patterns. In [6], the authors mapped trajectories onto grid cells and searched for top- k frequent routes. In [41], different transportation modes of a human trajectory are partitioned by walk segments. In [19], the authors partitioned a trajectory and changed it into line segments, and then clustered the lines instead. These works generated or considered only majority patterns in a region.

While the time or speed information of moving behaviors is as important as the positions, only few of trajectory clustering works take the temporal part into consideration. However, the temporal information as time or speed are used only for sampling fair trajectories [33], finding similar sub-trajectories in specified time window [42], detecting interesting places [27], and distinguishing moving patterns [20], but not on the clustering or the pattern finding process itself.

To extract regional characteristics inside trajectories, an important pre-process is to divide them into desired segments. In order to divide the raw trajectories properly, the work in [43], for example, computes the change of curviness among a trajectory. In [19], a minimum description length method based on [44] was used to divide trajectories by choosing a model that reaches the best tradeoff between the total length of partitions and the sum of the difference from the raw trajectory. However, the MDL method suffers from a high computation cost due to the repeated tradeoff tests, which may not be suitable to deal with large amount of continuous incoming trajectory data. Moreover, all the aforementioned trajectory dividing methods do not take the temporal domain of the trajectory data into consideration.

Last but not least, the distance measurement between trajectories is very important for trajectory clustering. When deciding the distance, many works measure it by computing the point-to-point Euclidean distance [30], or by the Hausdorff distance between two sets of data points [45], which simply computes the distance between some representative points of trajectories. If we regard a trajectory as a set of connected piecewise lines, measuring the distance between two trajectories is



to measure the line distances between them. To measure the line distance, some works simply measure the Euclidean distance between the start and the end points of two lines [38], which may overestimate the differences of forward directions and line lengths. In [19] and [45], the authors proposed to use the line segment Hausdorff distance to measure the parallel, perpendicular and angle distances between lines separately. However, this measurement was originally designed for pattern recognition that aimed to detect patterns even under stretch and rotation. Thus, it ignores the length difference and fails to reflect the direction change larger than 90 degrees. Furthermore, all of the aforementioned distance measurements do not take the temporal domain into consideration either.

To the best of our knowledge, our work is the first one to take both the spatial and temporal criteria into account for trajectory dividing, while clustering them with a distance measurement designed in consideration of the spatial and temporal information at the same time.

2.3 Problem Statement

Given a set of moving objects, we want to profile them by analyzing their accumulated trajectories. To be more specific, we want to group those similar sub-trajectories while separating dissimilar ones apart. The resulting profile should reflect distinct moving behaviors over the regions passed by the objects. Intrinsically, a trajectory, composed of consecutive points with positions and time-stamps, which may imply various speeds and directions in different parts, can be defined as follows.

Definition 2.1 (Trajectory and element line). *A trajectory is composed of a sequence of N data points. Namely,*

$$TR : \{pt_1, pt_2, \dots, pt_N\} = \{(p_1, t_1), (p_2, t_2), \dots, (p_N, t_N)\},$$

where p_i and t_i indicate the position and corresponding recorded time of point pt_i , respectively, and $t_1 < t_2 < \dots < t_N$. Two consecutive points constitute an element line $L : (pt_i, pt_{i+1})$. Therefore, a trajectory can also be viewed as a sequence of element lines. ■

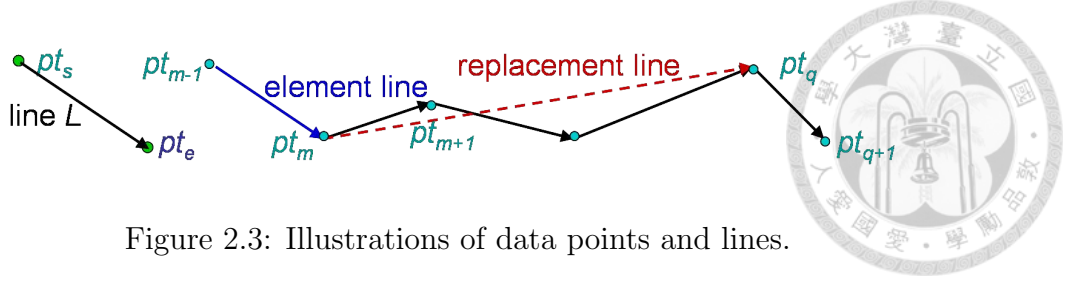


Figure 2.3: Illustrations of data points and lines.

If we treat each complete trajectory as an object and cluster them directly, it is hard to find similar ones as each trajectory has various length and contains different position and speed changes. Instead, it makes more sense to divide trajectories into homogeneous characteristic partitions, while considering both the spatial and temporal conditions, before any further processing. Therefore, we define the following terms.

Definition 2.2 (Sub-trajectory). A sub-trajectory is a subset of a trajectory, a part of consecutive data points in the trajectory. $STR : \{pt_m, pt_{m+1}, \dots, pt_q\}$, where $1 \leq m < q \leq N$. Similarly, a sub-trajectory can be regarded as a set of consecutive element lines. ■

Definition 2.3 (Line, its direction and speed). A directional line L is composed of two end points pt_s and pt_e as $L : (pt_s, pt_e)$, where $pt_s = (p_s, t_s)$ and $pt_e = (p_e, t_e)$. With $t_s < t_e$, we say its direction is from pt_s to pt_e , and its speed s_L is defined as

$$s_L = s(pt_s, pt_e) = \frac{d(p_s, p_e)}{(t_e - t_s)},$$

where $d(p_s, p_e)$ is the non-negative direct distance between pt_s and pt_e . ■

If some consecutive element lines of a sub-trajectory have similar speed and direction characteristics, we can summarize them with a *replacement line*, which is defined as follows.

Definition 2.4 (Replacement line). If a sub-trajectory is viewed as a sequence of points in ascending chronological order, a replacement line is the directional line connecting the point with earliest time-stamp and the one with latest time-stamp. That is, the replacement line of $STR : \{pt_m, pt_{m+1}, \dots, pt_q\}$ is $L : (pt_m, pt_q)$. ■

The illustration of data points and a line with direction along with the relationship between element lines and a corresponding replacement line is shown in Fig. 2.3.

With the above definitions, we can state the two main problems studied in this chapter.

Problem Definition 2.1 (Spatiotemporal Trajectory Dividing). *Given a trajectory TR , a spatial threshold and a temporal one, we would like to divide TR into a set of sub-trajectories $\{STR_1, STR_2, \dots, STR_M\}$ with an efficient algorithm. The divided sub-trajectories must meet the following requirements: 1) the variation in speeds (for temporal domain) and in directions (for spatial domain) of the element lines in each STR_i must be as small as possible, and 2) the number of generated sub-trajectories, i.e., M , must be as small as possible. Each sub-trajectory can then be replaced by a replacement line, and the whole set of replacement lines summarizes the trajectory TR . Meanwhile, the dividing algorithm should be able to handle the endless in-coming time-location data.* ■

Problem Definition 2.2 (Spatiotemporal Replacement Line Clustering). *Given a collection of replacement lines $\{L_1, L_2, \dots, L_O\}$ generated from a given trajectory set $\{TR_1, TR_2, \dots, TR_N\}$, we would like to cluster them into k clusters $\{C_1, C_2, \dots, C_k\}$. We want to adapt the k -means clustering algorithm and make it suitable for line items, and design a spatiotemporal distance function for these line objects. In other words, the line objects in the same cluster share similar speed and distribution characteristics. Each cluster C_j is then represented by a mean line, which shows the average position, length, direction, and speed information of the objects inside C_j . The mean lines of all clusters then gives the summary of distinct moving types of the trajectory set $\{TR_1, TR_2, \dots, TR_N\}$.* ■

To solve the two problems defined above, we proposed *DivCluST*. Through comprising of the two phases, DivST for dividing trajectories and the CluST for clustering replacement lines, the regional typical moving styles are profiled from trajectory data. We summarize the notation used in this chapter as Table 2.1. We shall elaborate on more details of the algorithm DivCluST in the following sections.

2.4 DivST: Spatiotemporal Trajectory Dividing

In this section, we shall introduce the design of spatiotemporal trajectory dividing algorithm, *DivST*, which solves Problem Definition 2.1. To partition a long trajec-

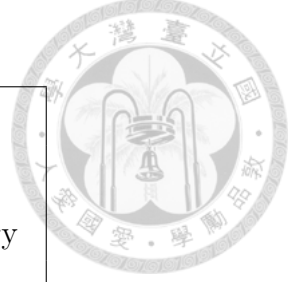


Table 2.1: *DivCluST*: List of notations

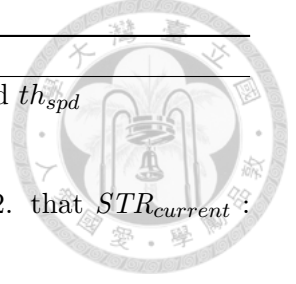
TR	$\{pt_1, pt_2, \dots, pt_N\}$	Trajectory
pt_i	(p_i, t_i)	Data point
STR	$pt_m, pt_{m+1}, \dots, pt_q$	Sub-trajectory
L	$L : (pt_s; pt_e)$, where $t_s < t_e$	Directional line
d_L	$d(p_s, p_e)$	Line length
s_L	$s(pt_s, pt_e) = \frac{d(p_s, p_e)}{(t_e - t_s)}$	Line speed

tory efficiently, we apply a sliding-window based algorithm that can online divide a given trajectory, without suffering from loading in the entire trajectory at one time and searching exhaustively for the global optimum partition of a trajectory. Moreover, we want to provide the flexibility on controlling the number of generated replacement lines. In this way, our dividing method not only can generate quality replacement lines but also can adjust the data amount to meet the resource constraints such as limited memory space. We introduce the algorithm in Section 2.4.1 and give remarks on the threshold selection in Section 2.4.2.

2.4.1 Spatiotemporal Trajectory Dividing Algorithm

Essentially, given a trajectory, we start the test from its initial element line. Then, we sequentially add each of the following element lines to form a new sub-trajectory, and test whether the new sub-trajectory fulfills both the spatial and temporal criteria, which would be explained later. If the inclusion of a certain element line, say L_e , fails to meet any of the dividing criteria, a replacement line is generated from the sub-trajectory without including L_e . Then, the same dividing test restarts from L_e to form next sub-trajectory and determine its replacement line. The spatiotemporal trajectory dividing algorithm is shown in Algorithm 2.1.

How the spatial and temporal criteria work is the key of proper trajectory dividing. In the spatial domain, as indicated in Problem Definition 2.1, the *direction* variation of each element line in a sub-trajectory is supposed to be as small as possible. In other word, we want a replacement line to represent a sub-trajectory containing element lines with similar directions. Our computation checks the length difference between a replacement line and its corresponding sub-trajectory based on



Algorithm 2.1 DivST: Spatiotemporal Trajectory Dividing

Input: a trajectory $TR : \{pt_1, pt_2, \dots, pt_N\}$, dividing criteria th_{len} and th_{spd}

Output: a set of replacement lines: L_1, L_2, \dots, L_M

Set start and end points of current sub-trajectory as $m = 1, q = 2$. that $STR_{current} : \{pt_1, pt_2\}$

while $m < N - 1$ and $q < N$ **do**

Test inclusion of next element line $L_e : (pt_q, pt_{q+1})$ on the corresponding sub-trajectory $STR_{next} : \{pt_m, \dots, pt_q, pt_{q+1}\}$ and the candidate replacement line $L : (pt_m, pt_{q+1})$ by dividing criteria th_{len} and th_{spd}

if criterion (3.1) or (3.2) is not fulfilled **then**

Output $L : (pt_m, pt_q)$ as a replacement line of $STR_{current} : \{pt_m, pt_{m+1}, \dots, pt_q\}$

Set $m = q, q = q + 1$

else

Set $q = q + 1$

end if

end while

Output the last line $L_M : (pt_m, pt_N)$

the property of triangle inequality, the length of one edge of a triangle would be close to the sum of the other two edges when an inner angle is close to 180° and the other two are near 0° . Since each divided sub-trajectory is replaced by a replacement line, we may ensure a small direction variation of element lines inside a sub-trajectory by examining the difference between the length of its replacement line and the length sum of all its corresponding element lines.

Suppose the current sub-trajectory $STR_{current} : \{pt_m, pt_{m+1}, \dots, pt_q\}$ has met the spatial criterion, and the coming element line is $L_e : (pt_q, pt_{q+1})$. We check the length difference between $STR_{next} : \{pt_m, \dots, pt_q, pt_{q+1}\}$ and the candidate replacement line $L : (pt_m, pt_{q+1})$. Given the threshold th_{len} , if the following condition in (3.1) is met, the coming element line $L_e : (pt_q, pt_{q+1})$ would then be included. Then we renew $STR_{current}$ to $\{pt_m, \dots, pt_q, pt_{q+1}\}$, and continue the test on the next element line $L_{e+1} : (pt_{q+1}, pt_{q+2})$. Otherwise, the replacement line $L : (pt_m, pt_q)$ is generated and the dividing test is restarted from L_e , by assigning $STR_{current}$ as $\{pt_q, pt_{q+1}\}$ and

the next element line L_{e+1} as (pt_{q+1}, pt_{q+2}) .

$$\frac{diff_{len}}{\sum_{i=m}^{q-1} d(p_i, p_{i+1})} < th_{len}, \quad (2.1)$$



where $diff_{len} = \sum_{i=m}^{q-1} d(p_i, p_{i+1}) - d(p_m, p_q)$.

In the temporal domain, we consider only the variance of forward speeds. To be more specific, we want to ensure a replacement line can represent a sub-trajectory containing element lines with similar speeds. We sum the speed difference between a candidate replacement line and each element line contained in STR_{next} and normalize the result by the total speed sum of each element line as in (3.2). Note that we use speed, which indicates the forward moving measurement, rather than velocity, which includes both speed and direction, since the influence of direction has been computed in the spatial domain th_{len} .

$$\frac{diff_{spd}}{\sum_{i=m}^{q-1} s(pt_i, pt_{i+1})} < th_{spd}, \quad (2.2)$$

where $diff_{spd} = \sum_{i=m}^{q-1} |s(pt_i, pt_{i+1}) - s(pt_m, pt_q)|$.

To show how the designed spatial and temporal criteria (3.1) and (3.2) work, we list three examples in Fig. 2.4. Suppose L_1 and L_2 are two element lines which span equal time duration. We would like to check if it is acceptable to include them in the same sub-trajectory and replace them by L_R . Fig. 2.4(a) and Fig. 2.4(b) are two conditions that would pass the spatial domain test in (3.1), while Fig. 2.4(a) and Fig. 2.4(c) are two conditions that would fulfill the temporal domain criterion in (3.2). For Fig. 2.4(a), the two element lines L_1 and L_2 have almost the same direction, so the sum of their lengths approximates to the length of L_R . For Fig. 2.4(b), although the direction difference between L_1 and L_2 is rather big, it can satisfy the spatial dividing criterion since their lengths are also very different and the length of L_1 dominates the test. However, it would not satisfy the temporal criterion as the speed variance is large between L_1 and L_2 . In both Fig. 2.4(a) and Fig. 2.4(c), L_1 and L_2 have similar speed value and pass the temporal test (3.2), the dividing is then decided by the spatial criterion (3.1).

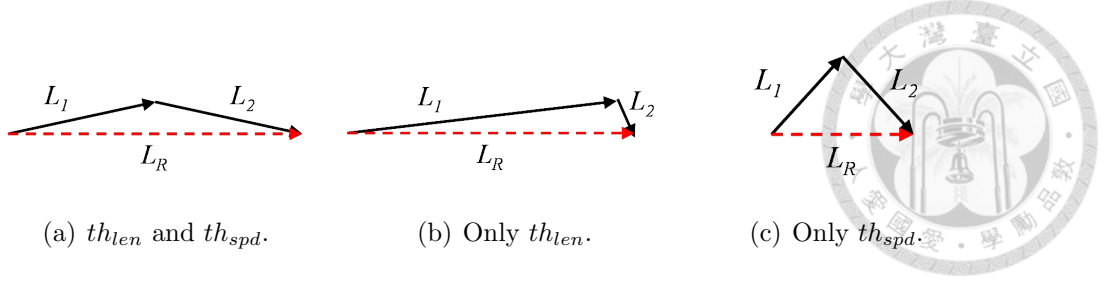


Figure 2.4: Illustrations of element lines L_1 , L_2 passing the dividing criteria th_{len} and th_{spd} and being replaced by L_R .

2.4.2 Selection of Thresholds in DivST

The goal of DivST is to generate a minimum number of replacement lines while ensuring the lines to preserve as many spatiotemporal properties of the original trajectory data as possible, as stated in Problem Definition 2.1. The goal can be reached through a proper selection of thresholds th_{len} and th_{spd} , to find the balance between *concision* (quantity) and *precision* (quality) when dividing a given trajectory data set.

A better precision can be reached by setting lower th_{len} and th_{spd} such that the differences of length and speed between a original sub-trajectory and its replacement line would be smaller. However, by doing so, more replacement lines would be generated, which increases the computation load of later steps. On the other hand, suppressing the number of replacement lines sacrifice precision. Hence, we make a proper selection of th_{len} and th_{spd} through evenly taking both preciseness and conciseness into account when dividing trajectories and generating replacement lines. Regarding this, we propose the two measurements, $R_{precise}$ and $R_{concise}$, defined as follows.

$$\begin{aligned}
 R_{precise} &= (th_{len} + th_{spd})/2, \text{ and} \\
 R_{concise} &= f(th_{len}, th_{spd}) = \frac{\text{total replacement lines}}{\text{total element lines}}.
 \end{aligned}
 \tag{2.3}$$

The $R_{precise}$ would be lower as th_{len} and th_{spd} are smaller, which makes the criteria more strict. On the other hand, $R_{concise}$ would be lower as the number of total replacement lines decreased, which is reached by assigning larger th_{len} and th_{spd} . Thus, the decrease of $R_{precise}$ and $R_{concise}$ contradicts to each other by natural. To make the best trade-off between precision and concision, we choose th_{len} and th_{spd}

that derive the minimum sum of $R_{precise}$ and $R_{concise}$ as follows.

$$(th_{len}, th_{spd}) = \arg \min(R_{precise} + R_{concise}). \quad (2.4)$$

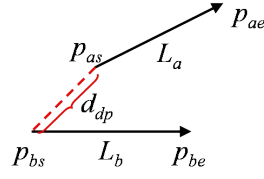
The best-fit criteria th_{len} and th_{spd} can be determined heuristically by observing the historical trajectories. They can also be adjusted dynamically to meet different precise and concise requirements and produce line segments accordingly while introducing proper weights to the criteria in (2.3) and obtaining thresholds according to (2.4). This shows DivST can provide better flexibility when dealing with huge amounts of trajectory data under different available memory space or other computing resources.

2.5 CluST: Spatiotemporal Replacement Line Clustering

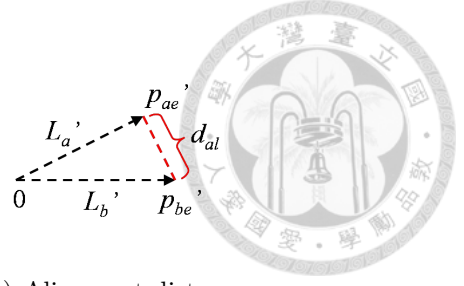
In *CluST*, we deal with Problem Definition 2.2 and cluster the replacement lines of trajectories obtained from the DivST phase. As a replacement line is representing those consecutive element lines with homogeneous speeds and directions inside a trajectory, we do not need a special designed model-based clustering algorithm as what [33] [29] [30] [31] [32] [34] do. Instead, we choose to use a typical clustering method to find convex clusters of replacement lines that are similar in geographic positions, lengths, directions, and speeds, then determine a representation for each cluster. Under this goal, the density based clustering method perhaps is not a good choice as it was in [19]. Although the density reachable idea makes it possible to find a cluster with an arbitrary shape, it may include replacement lines that are in fact quite different apart but connected due to the similar ones in-between. As a result, we adapt the k -means idea, which is the most popular partition-based clustering method in practice [46], to design our CluST method.

2.5.1 Spatiotemporal Line Distance

Measuring the distance between objects is the basis of a clustering method. We propose that a proper line distance measurement should consider both the spatial and temporal characteristics. As a result, the line distance function is constructed by

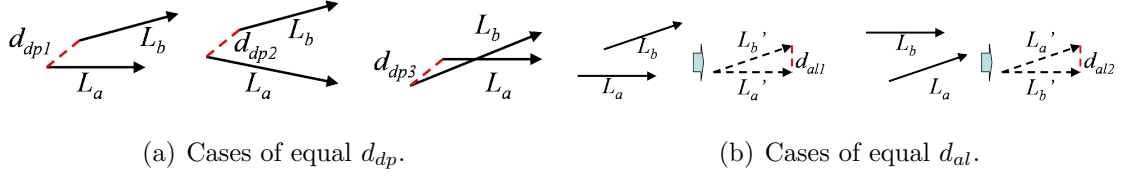


(a) Displacement distance.



(b) Alignment distance.

Figure 2.5: Illustrations of the two parts of shift distance.



(a) Cases of equal d_{dp} .

(b) Cases of equal d_{al} .

Figure 2.6: Illustrations of equal shift distance.

two major parts, the *shift distance* d_{sht} and the *speed distance* d_{spd} . The shift distance measures the spatial difference between lines with its displacement and alignment parts. In the meantime, the speed distance indicates the temporal difference of two lines, which measures the variance between their forward speeds. Thus, the spatiotemporal distance $d(L_a, L_b)$ between a pair of lines L_a and L_b is as below.

$$d(L_a, L_b) = w_{sht} \times d_{sht}(L_a, L_b) + w_{spd} \times d_{spd}(L_a, L_b), \quad (2.5)$$

where w_{sht} and w_{spd} are weights for the shift and speed distances respectively. The weights can be adjusted to strengthen either of spatial or temporal information, or to normalize the value range of different distance measure granularities used for the two parts [47].

Shift Distance

Here we illustrate the spatial distance, measured by d_{sht} , between two lines. When taking one of the lines as reference base, the relative positions between two lines can be described by three parts, the displacement between their start points, the stretching or shortening of line length, and the rotation from the referenced line. We compute the first part using the *displacement distance*, d_{dp} , and the following two together as the *alignment distance*, d_{al} , as illustrated in Fig. 2.5. Note that L'_a and L'_b in Fig. 2.5(b) are the result of shifting L_a and L_b in Fig. 2.5(a), and make

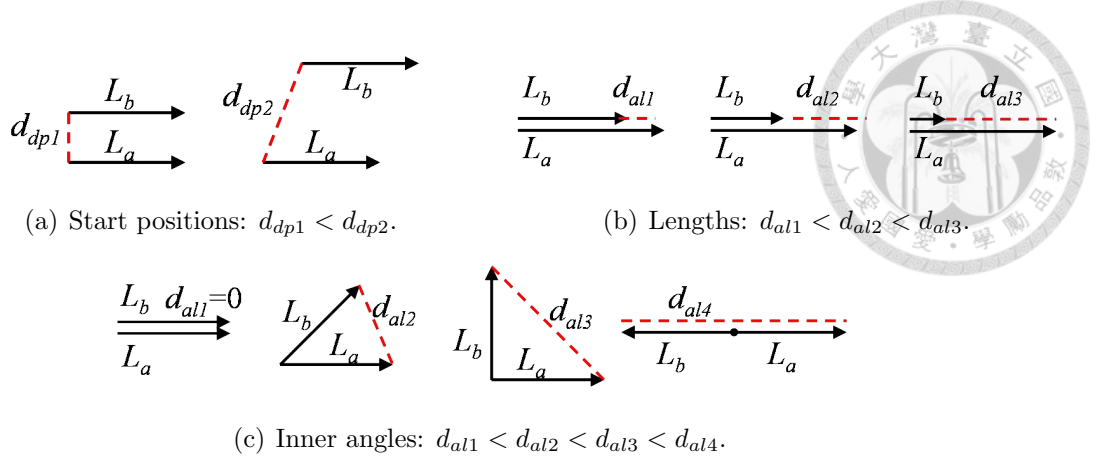


Figure 2.7: Condition changes vary shift distances.

their start points virtually overlap at the same position.

The displacement distance d_{dp} measures the geographic positioning difference [48] between the start points of a pair of lines, defined as below,

$$d_{dp}(L_a, L_b) = d(p_{as}, p_{bs}), \quad (2.6)$$

where p_{as} and p_{bs} are the start points of lines L_a and L_b . There are possibly more than one case having the same d_{dp} , as shown in Fig. 2.6(a). Though their lengths and directions are different, the distances between the starting points of two lines in these cases are the same. In addition, according to (2.6), d_{dp} increases as the longitudinal or latitudinal displacement between the two start points on L_a and L_b increases, as illustrated in Fig. 2.7(a).

The alignment distance d_{al} is designed to estimate only relative length stretching or shortening and the relative rotation between a pair of lines. It computes the relative change between the end points of two lines, to avoid repeatedly estimating the part that is already computed in the displacement distance, as illustrated in Fig. 2.5(b). The alignment distance d_{al} is defined as

$$d_{al}(L_a, L_b) = d(p'_{ae}, p'_{be}) = d(p_{ae} - p_{as}, p_{be} - p_{bs}), \quad (2.7)$$

where p_{ae} and p_{be} are the end points of lines L_a and L_b , and p'_{ae} and p'_{be} are the new end positions of the shifted lines L'_a and L'_b . There may exist cases of line pairs ending in the same virtually connected lines L'_a and L'_b , and getting the same d_{al} as shown in Fig. 2.6(b). The d_{al} increases as the length difference gains or the inner angle of L_a and L_b enlarges, as illustrated in Fig. 2.7(b) and Fig. 2.7(c). In addition, some special cases of zero d_{dp} and d_{al} are illustrated in Fig. 2.8.

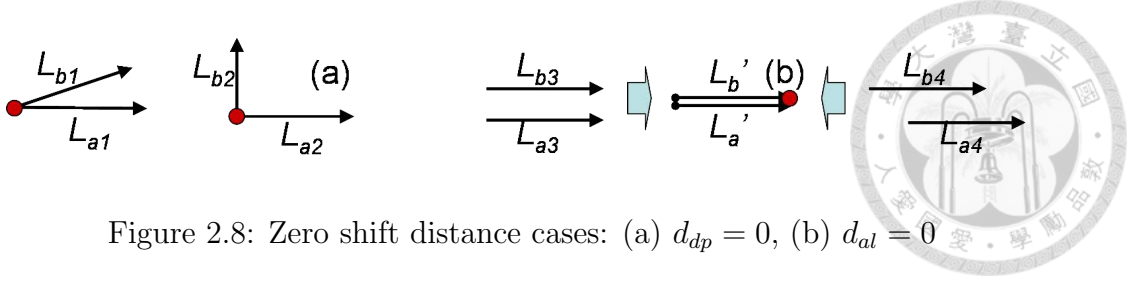


Figure 2.8: Zero shift distance cases: (a) $d_{dp} = 0$, (b) $d_{al} = 0$



(a) $d(p_{as}, p_{ae}) = d(p_{bs}, p_{be})$, $t_{ae} - t_{as} = t_{be} - t_{bs}$. (b) $d(p_{as}, p_{ae}) = 3d(p_{bs}, p_{be})$, $t_{ae} - t_{as} = 3(t_{be} - t_{bs})$.

Figure 2.9: Cases of zero speed distances: $s_{L_a} = s_{L_b}$.

The shift distance d_{sht} , comprised of d_{dp} and d_{al} , is defined as follows.

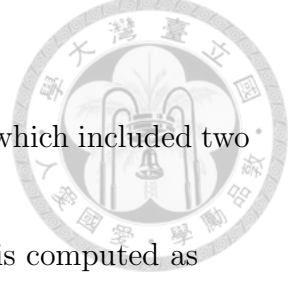
$$d_{sht}(L_a, L_b) = d_{dp}(L_a, L_b) + d_{al}(L_a, L_b). \quad (2.8)$$

Speed Distance

Here we illustrate the speed distance, measured by d_{spd} , between two lines. d_{spd} computes the difference of forward speeds of a pair of lines, without considering about their moving directions. The reason is that only the forward speed belongs to the temporal domain. The direction is in fact a part of spatial domain and its difference has already been considered in d_{al} of the shift distance. The speed distance is defined as below.

$$d_{spd}(L_a, L_b) = |s_{L_a} - s_{L_b}| = \left| \frac{d(p_{as}, p_{ae})}{t_{ae} - t_{as}} - \frac{d(p_{bs}, p_{be})}{t_{be} - t_{bs}} \right|. \quad (2.9)$$

The speed distance is the difference of forward speed values between two replacement lines, which is similar to the design of temporal dividing criterion th_{spd} in Section 2.4.1. Note that d_{spd} defined in (2.9) is independent of the absolute time-stamp values of each replacement line. In this way, we can discover lines with similar speed patterns across different time periods. Two cases of d_{spd} equivalent to zero are illustrated in Fig. 2.9, where the lengths, directions and time spent of the pair of lines may all be different, but the speeds of the lines can be equal and make zero speed distances.



Distance Function Analysis

The spatiotemporal line distance $d(L_a, L_b)$ is measured by (2.5), which included two components, d_{sht} and d_{spd} .

In shift distance d_{sht} , base on (2.6), its displacement part d_{dp} is computed as

$$d_{dp}(L_a, L_b) = d(p_{as}, p_{bs}) = d((x_{as}, y_{as}), (x_{bs}, y_{bs})) = \sqrt{(x_{bs} - x_{as})^2 + (y_{bs} - y_{as})^2}.$$

We can see that as $(a - b)^2 \geq 0$ is always true, $d_{dp}(L_a, L_b) \geq 0$ would also be true. And $d_{dp}(L_a, L_b) = 0$ happens only when $x_{bs} = x_{as}$ and $y_{bs} = y_{as}$, which indicating $L_a = L_b$. Also, as $(b - a)^2$ would always equal to $(a - b)^2$, we have $d_{dp}(L_a, L_b) = d_{dp}(L_b, L_a)$. Finally,

$$d_{dp}(L_a, L_c) = d(p_{as}, p_{cs}) \leq d(p_{as}, p_{bs}) + d(p_{bs}, p_{cs}) = d_{dp}(L_a, L_b) + d_{dp}(L_b, L_c)$$

would be met based on the triangle inequality property among three points, $\{p_{as}, p_{bs}, p_{cs}\}$, in a space. From the above analysis, the displacement part d_{dp} of shift distance is a metric. Next, the alignment part d_{al} as in (2.7) is computed by the distance between relative end points p'_e of line, where $p'_e = p_e - p_s$. Thus, similar to the analysis of d_{dp} , d_{al} also fits non-negativity, identity of indiscernible, symmetry and subadditivity conditions required for a metric. In conclusion, d_{sht} defined in (2.8) is a metric, while both of its components, d_{dp} and d_{al} , are metrics.

As to d_{spd} defined by (2.9), we can see both condition $d_{spd}(L_a, L_b) \geq 0$ and $d_{spd}(L_a, L_b) = d_{spd}(L_b, L_a)$ would always be met with the absolute value assigned in distance. Next, as d_{spd} is an absolute difference of two real numbers, by definition it would meet triangle inequality function as $|a - c| \leq |a - b| + |b - c|$, while equality holds if and only if $a \leq b \leq c$. However, according to our definition of s_L , we would get multiple possibilities of (pt_s, pt_e) pairs having the same s_L , just as Fig. 2.9(b) shown. Thus, d_{spd} does not meet the identity of indiscernible condition of a metric. In conclusion, d_{spd} is not a metric, but is only a pseudo-metric.

2.5.2 k -means Based Line Clustering Algorithm

Here we show how k -means based idea is adapted to our clustering method, CluST. First, CluST starts with k randomly selected replacement lines as initial cluster



Algorithm 2.2 CluST: Spatiotemporal Line Clustering

Input: a set of replacement lines: L_1, L_2, \dots, L_M , number of cluster: k
Output: line clusters: C_1, C_2, \dots, C_k and k cluster means

 Randomly choose k lines as the initial cluster means

repeat

 for $i = 1$ to M **do**

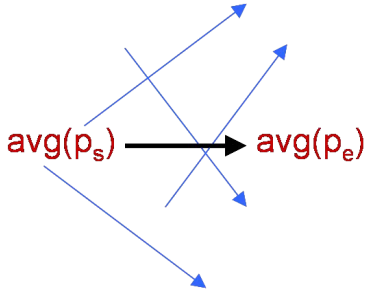
 Assign L_i to the cluster with the closest mean line using the spatiotemporal line distance function

 end for

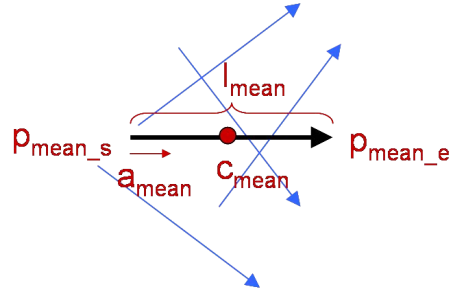
 for $j = 1$ to k **do**

 Update the mean line of cluster C_j

 end for
until no new cluster assignment for any L_i
return k clusters and their corresponding mean lines



(a) Average two end points.



(b) Deal each property separately.

 Figure 2.10: Illustration of mean line computation in *CluST*.

means. Then, it generates new partitions by assigning each of the remaining replacement lines to its nearest mean using the distance measurement defined in (2.5). After computing new mean of each cluster, CluST repeats the previous steps until no more new assignment is needed. The algorithmic form of CluST is shown in Algorithm 2.2.

The next problem we are solving is how to decide the *cluster mean* of a group of replacement lines. Since a cluster generated by CluST is a collection of directional lines, its mean should also be a line with similar format that represents the temporal and spatial attributes of all the cluster members, which are replacement lines. We



denote the *mean* of our line cluster as

$$L_{mean} : (p_{mean_s}, p_{mean_e}, s_{mean}),$$

where p_{mean_s} , p_{mean_e} are two end points describing the position of mean line and s_{mean} is the average speed of cluster members, respectively. The initial idea of obtaining p_{mean_s} and p_{mean_e} is to simply average the start p_s and end p_e positions of every line in the cluster, as illustrated in Fig. 2.10(a), where the thin lines indicate cluster members and the thick line is their mean. However, the length and direction information of lines is blurred during the averaging process. Instead, we propose to compute the mean line that preserve average position, length, direction and speed, respectively, from all the members in a line cluster, as shown in Fig. 2.10(b). First we compute the average center position c_{mean} , average length l_{mean} , and unit vector a_{mean} for the direction of all line members L_n in a cluster C as follows.

$$\begin{aligned} c_{mean} &= \frac{\sum_{L_n \in C} (p_{ns} + p_{ne})/2}{|C|}, \\ l_{mean} &= \frac{\sum_{L_n \in C} d(p_{ns}, p_{ne})}{|C|}, \text{ and} \\ a_{mean} &= \frac{\sum_{L_n \in C} ((p_{ne} - p_{ns})/d(p_{ns}, p_{ne}))}{|C|}, \end{aligned} \quad (2.10)$$

where p_{ns} and p_{ne} are the two end points of L_n , and $|C|$ is the number of lines in cluster C . Then, the mean line L_{mean} is computed using (3.5) based on (3.4).

$$\begin{aligned} p_{mean_s} &= c_{mean} - \frac{l_{mean} \times a_{mean}}{2}, \\ p_{mean_e} &= c_{mean} + \frac{l_{mean} \times a_{mean}}{2}, \text{ and} \\ s_{mean} &= \frac{\sum_{L_n \in C} s_{L_n}}{|C|}. \end{aligned} \quad (2.11)$$

2.5.3 Profiling Moving Objects by Interpreting Trajectory Clustering Results

By properly interpreting the cluster means found out in CluST, we can profile the given set of moving objects well. The output mean lines give an overall picture of the original trajectory set with properties as their laying positions, spanning lengths, stretching directions, moving speeds as well as the distribution densities of the input trajectory data. We carefully design a method to visualize the clustering results and

demonstrate the characteristics of each cluster. Besides drawing the mean lines on their related geographical positions and spans, we also put a mark on the end point of each mean line to show their forward directions, use different colors to represent the speeds and use line widths to represent the cluster sizes. Thus, the graph of cluster means becomes a well profile that summarizes the moving patterns of given trajectories.

To approximate a proper range of k , we can apply the cluster validation methods such as those in [38] [49] [50] with the spatiotemporal distance defined here. We check the range of k based on the inter-cluster and intra-cluster scattering measurements.

$$\begin{aligned} CD_{inter} &= \min_{1 \leq i, j \leq k, i \neq j} d(L_{mean_i}, L_{mean_j}), \text{ and} \\ CD_{intra} &= \frac{\sum_{L_n \in C} d(L_i, L_{mean_n})}{k \times |C|}, \end{aligned} \quad (2.12)$$

where L_{mean_n} is the mean of C_n and $1 \leq n \leq k$. A small enough CD_{intra} and a relatively larger CD_{inter} are required for a proper k selection. And of course, to get a more precise k , the parameter selection requires expertise of domain experts. The profiling results are presented using the cluster mean lines. The results would only reveal those significant and dominated spatiotemporal properties at smaller k 's while some details would be omitted. On the other hand, more information would be uncovered accompanying with more noise at larger k 's.

2.6 Experiment Results

We conducted extensive experiments on three different real trajectory data sets to evaluate DivCluST, and to show their profiling results. In addition, we compared the results of DivCluST with those considered only the spatial or temporal information in the trajectory dividing phase and in the line clustering phase. Moreover, we analyzed the impact of different thresholds in the DivST phase and different k in the CluST phase. Finally, we implemented the MDL segmenting and the DBSCAN method with our spatiotemporal line distance measurements to compare its trajectory dividing and profiling results with ours. All algorithms were implemented in C++ and the experiments were run with 2.5GHz CPU and 2GB RAM.

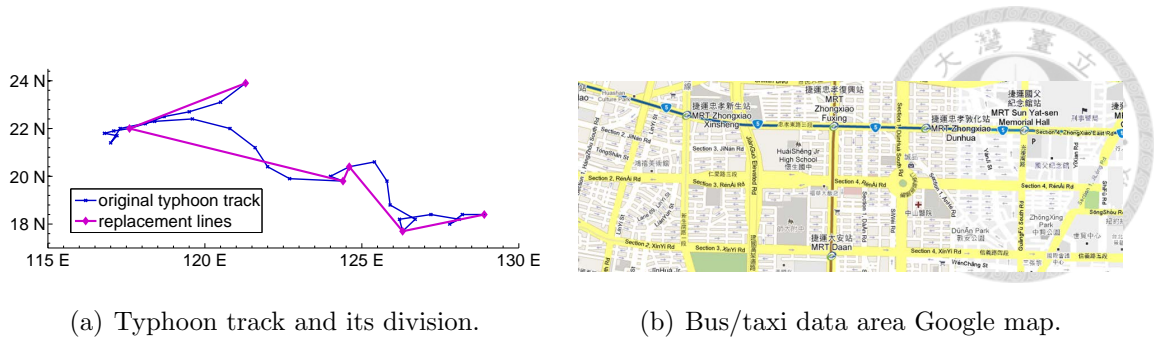


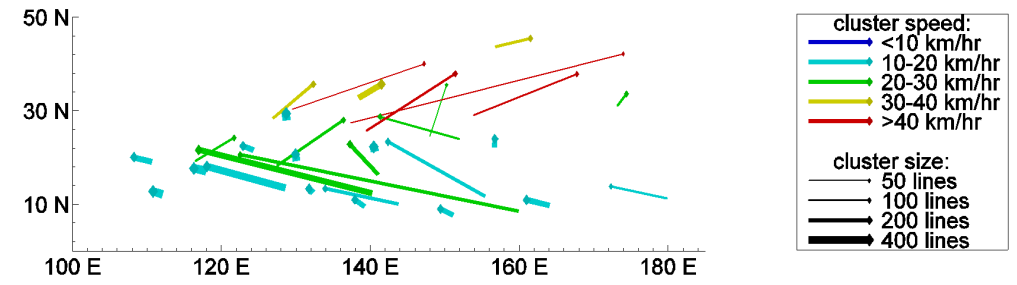
Figure 2.11: Information about real data sets.

2.6.1 Three Sets of Real Trajectory

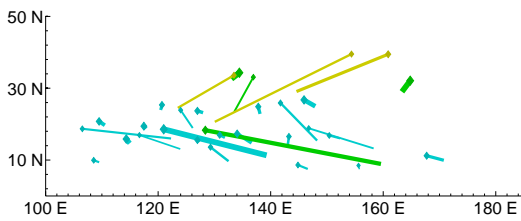
The three real trajectory data sets we used for experiments were typhoon tracks, bus and taxi trajectories. The first data set we chose to use was typhoon tracks over the Western North Pacific, which is the area having the most typhoon activities. The typhoon best tracks used for the experiments were data from year 1945 to 2007 offered by Joint Typhoon Warning Center [51]. The locations of each typhoon were recorded every six hours from the time it formed till it disappeared. Therefore, we gathered tracks of 1,830 typhoons with totally 53,075 records.

The second data set we used was the trajectories of city buses in Taipei. We gathered two weeks of data during mid-March, 2010 from the Taipei city e-bus system [52]. This system collected the GPS position data of all city buses on duty around every 100 seconds. The bus trajectory data within several blocks in the downtown area, as shown in Fig. 2.11(b), was selected for the experiments. We have totally 3,296 buses, including 433 routes and 191,932 runs within 2,802,586 GPS records for profiling.

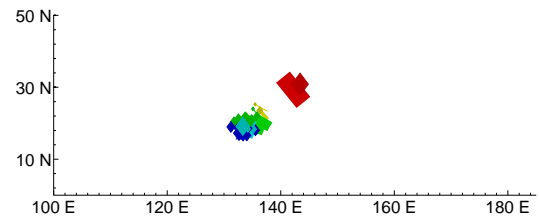
The third data set was the GPS trajectories of taxis in Taipei. We used the data recorded between 7-8 PM of the whole November, 2010 with the sampling rate be one data point per second. We chose only data points that are located within the same area as that chosen for the bus data. Totally 1,466,471 records were used for experiments. We aimed to compare the profiling results of the taxi data, which may have more erratic movements, to those of the bus data.



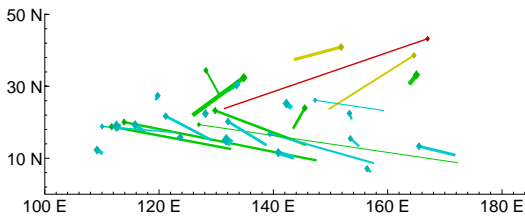
(a) Typhoon track *DivCluST* ($k = 30, th_{len} = 0.15, th_{spd} = 0.25$)



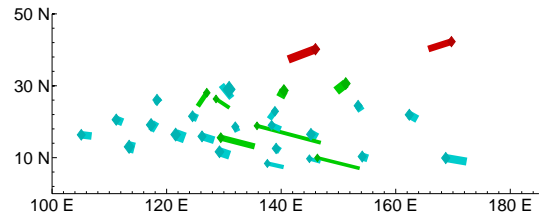
(b) *S-only* dividing and clustering.



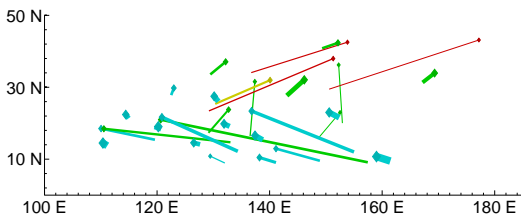
(c) *T-only* dividing and clustering.



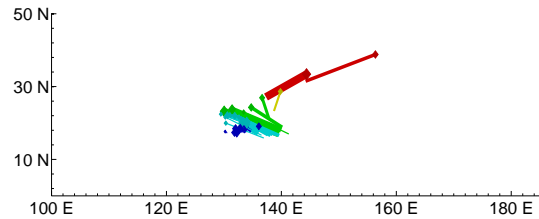
(d) *S-only* dividing and *CluST*.



(e) *T-only* dividing and *CluST*.

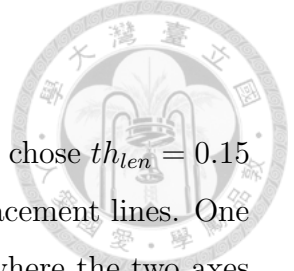


(f) *DivST* and *S-only* clustering.



(g) *DivST* and *T-only* clustering.

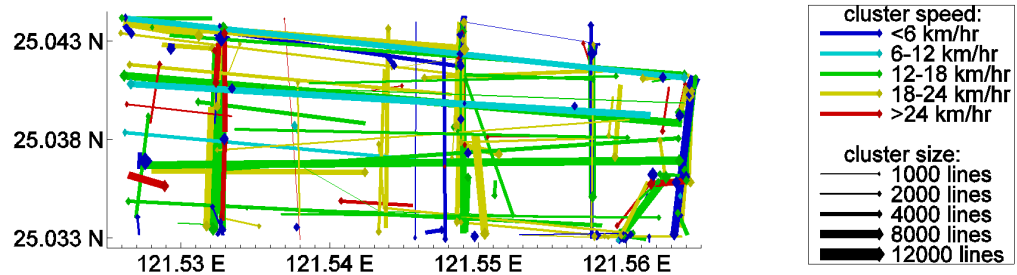
Figure 2.12: Profiling results on typhoon track and corresponding spatial-only and temporal-only comparisons.



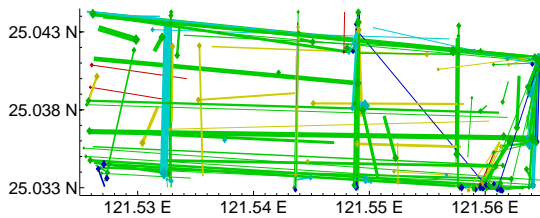
2.6.2 Profiling Results of DivCluST

We first show the profiling results on the typhoon track data. We chose $th_{len} = 0.15$ and $th_{spd} = 0.25$ according to (2.4) and thus obtained 7,269 replacement lines. One sample of the divided typhoon tracks is shown in Fig. 2.11(a), where the two axes in this and the following profiling result figures are their longitude and latitude. More details about the threshold analysis of th_{len} and th_{spd} will be given in Section 2.6.4. Fig. 2.12(a) shows the mean lines generated by DivCluST when $k = 30$. Each mean line represents one cluster. In this and several following plots, the speed, cluster size and forward direction of each mean line are represented with a corresponding color, a proportional width, and a diamond mark respectively. To be more specific, the colors varied from red to blue as the speeds of objects were from fast to slow, while the lines were drawn wider as more replacement lines are contained in one cluster. In Fig. 2.12(a), the DivCluST approach successfully revealed the main moving behaviors of typhoons [53]. Typhoons over the Western North Pacific typically move westward after developing over tropical oceans, which are shown by the green lines in the bottom half of Fig. 2.12(a). Then roughly half of the typhoons swing poleward around subtropical high, while most of the others move over to lands. The speeds of typhoons decrease significantly when they are hitting lands or recurving, which are indeed shown by the cyan short lines heading west or north in the left part. When typhoons move far enough north, they are caught in the westerly flow and curve to northeast. In middle latitudes, the forward speeds of typhoons normally increase, which are clearly shown by the yellow and red lines on the upper part of Fig. 2.12(a).

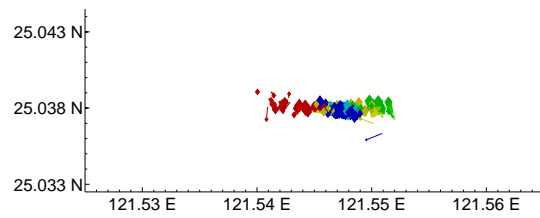
Next, we show the profiling results of the bus trajectory data. We chose $th_{len} = 0.1$ and $th_{spd} = 0.9$ according to (2.4), and the bus trajectories were replaced by 535,932 lines. Similarly, we will discuss the choice of thresholds later in Section 2.6.4. We plotted the cluster means of $k = 160$ in Fig. 2.13(a). The clustering results illustrated many representative characteristics of bus moving behaviors and verified the effectiveness of DivCluST. By plotting the cluster means, we can see a clear outline of the road network, of which the real road map is shown in Fig. 2.11(b). We easily identified the main bus routes with the wider lines that obviously fall on those main streets in the chosen area. The average speed of each cluster never



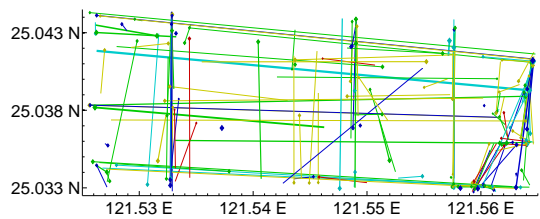
(a) Bus trajectory *DivCluST* ($k = 160, th_{len} = 0.1, th_{spd} = 0.9$)



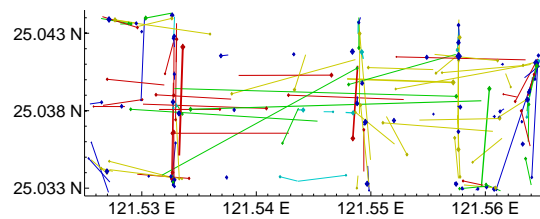
(b) *S-only* dividing and clustering.



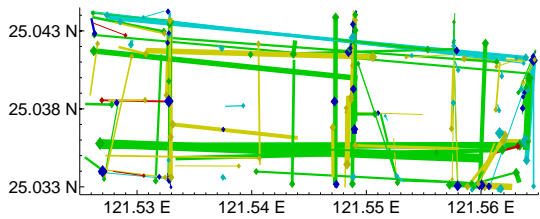
(c) *T-only* dividing and clustering.



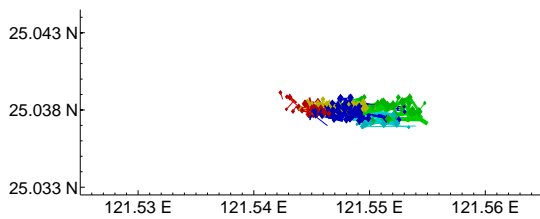
(d) *S-only* dividing and *CluST*.



(e) *T-only* dividing and *CluST*.



(f) *DivST* and *S-only* clustering.



(g) *DivST* and *T-only* clustering.

Figure 2.13: Profiling results on bus trajectory and corresponding spatial-only and temporal-only comparisons.

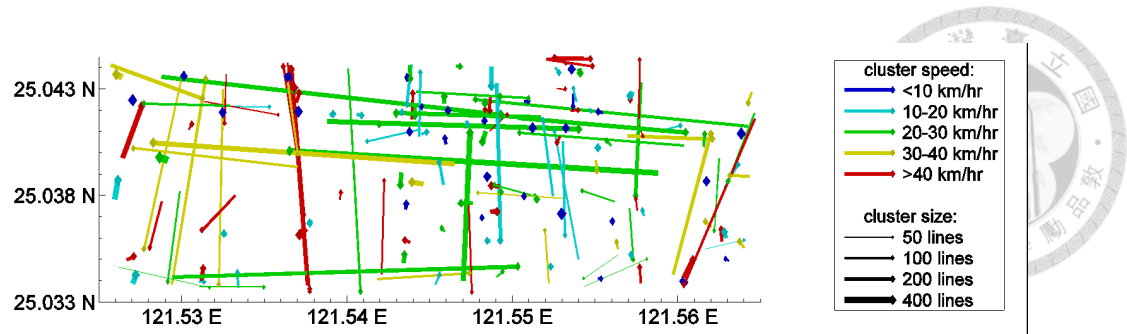


Figure 2.14: Profiling results on taxi trajectory data

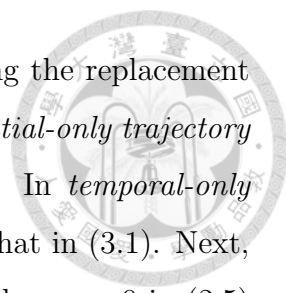
exceeded 40 km/hr, which is the limit of the city. In addition, there are close parallel lines in opposite directions, showing those bi-directional bus routes. There are also overlapped lines with different colors from red, yellow, green to cyan, which implied the different speeds, from fast to slow, of bus movement under different traffic conditions. We also notice some big blue spots that clearly showing the zero speed of buses at the main bus stops or crossroads on their routes.

Third, we show the profiling results of the taxi trajectory data. We used the same parameters as bus data in DivST, i.e., $th_{len} = 0.1$ and $th_{spd} = 0.9$, and got 31,144 replacement lines after dividing. The results, with $k = 160$, were shown in Fig. 2.14. Compared to Fig. 2.13(a), we observed the same main road networks while many new lines appeared. These new lines indicated those road segments that were never passed by buses. Meanwhile, since the movement of taxis were faster and more erratic compared to buses, we observed more red color lines and various short lines indicating some fast and irregular moving behaviors in Fig. 2.14.

The experiment results on the three real data sets, which locate in very different geographic scopes and have distinct moving styles, well sketch their different regional moving behaviors and successfully shown the effectiveness of DivCluST.

2.6.3 Comparisons with Spatial-only and Temporal-only Profiling

Here we show the importance of considering both of spatial and temporal properties when dividing and clustering trajectories. Focusing only on the spatial or temporal domain cannot capture the whole picture of trajectory moving behaviors. To compare with the results of DivCluST algorithm, we profiled the trajectory set by using only the spatial or temporal criterion when dividing trajectories and measuring



only the spatial or temporal part of line distance when clustering the replacement lines. In other words, we have the following settings. First, in *spatial-only trajectory dividing*, we applied only the criterion (3.1) and ignored (3.2). In *temporal-only trajectory dividing*, we used only the criterion in (3.2) and skip that in (3.1). Next, we set $w_{spd} = 0$ in (2.5) for *spatial-only trajectory clustering*, and $w_{sht} = 0$ in (2.5) for *temporal-only trajectory clustering*. Also, to be fair, we set the parameters th_{len} , th_{spd} , w_{sht} , w_{spd} and k the same as those in Section 2.6.2, if not specifically specified.

The results of dividing and clustering two real trajectory sets with only spatial information were shown in Fig. 2.12(b) and 2.13(b). The cluster mean lines were with much less speed variety (in much similar colors) comparing to those obtained by DivCluST in Fig. 2.12(a) and 2.13(a). It is because here a long trajectory was segmented only at significant direction change, while the speed changes along the same direction were omitted. The condition was even more obvious on the bus trajectory data. We observed many straight and long cluster mean lines, since the replacement lines tended to be straight and long along roads, and were clustered into one group even if their speeds were significantly different. Moreover, the stops of bus trajectory shown as blue spots in Fig. 2.13(a) almost totally vanished in Fig. 2.13(b).

The results of spatial-only trajectory dividing with regular CluST on two real data sets were shown in Fig. 2.12(d) and 2.13(d). We observed similar phenomenons as in Fig. 2.12(b) and 2.13(b). Only more speed variety was revealed as the clustering phase considered the speed distance. We then show what if we divided trajectories with DivST but clustering the replacement lines with only the spatial distance. The results were in Fig. 2.12(f) and 2.13(f). Although the outcome had a layout similar to Fig. 2.12(a) and 2.13(a), we still lost some speed information as the clustering phase grouped those replacement lines of different speeds but in close geographic locations.

Next, we show the results when the spatial information was not considered in either the dividing phase or the clustering phase. Fig. 2.12(c) and 2.13(c) show the results of temporal-only dividing and temporal-only clustering. All the spatial properties vanished so that we could not distinguish the spreads, lengths and directions of lines. Most of the mean lines was short because the clusters contained

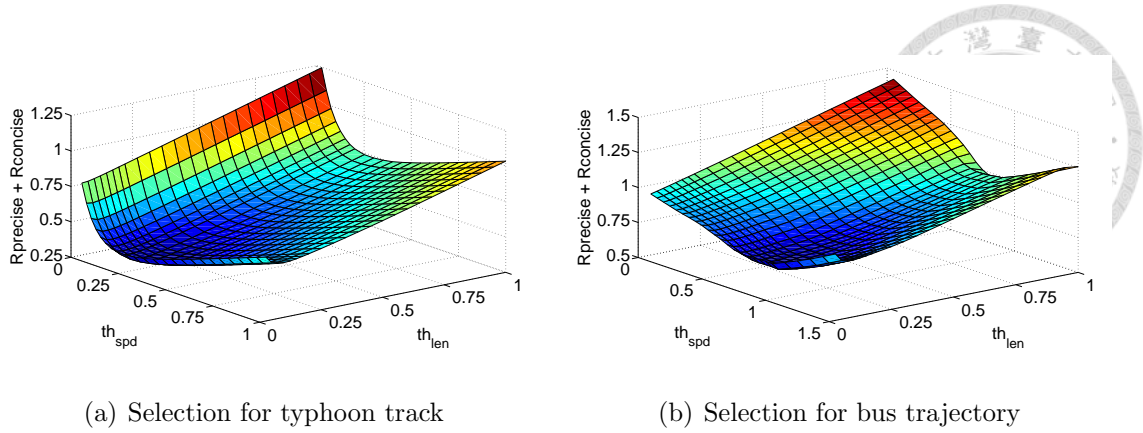


Figure 2.15: The *DivST* parameter selection analysis.

replacement lines of various positions and directions. Also, due to the lack of counting in the spatial information, all the mean lines gathered together in the middle of the observing geographic area after the average position computation in (3.4). Although obtaining the speed variances between clusters, we lost all of the important geographic properties.

Fig. 2.12(e) and 2.13(e) shows the results when we did temporal-only trajectory dividing and CluST. The results revealed many small fragments with different speeds distributing among different geographic regions. However, as the dividing phase only had response to the speed change over trajectory, we lost most of the spatial moving trend, which should be revealed by the length of the cluster mean lines. Also, most of the road distribution information vanished in Fig. 2.13(e). In Fig. 2.12(g) and 2.13(g), we show the results when dividing trajectories with DivST but clustering the replacement lines with only the temporal distance. As we can see, most of the spatial information disappeared due to the clustering phase grouped replacement lines with similar speeds no matter how far their real spatial distances were.

With the above results, we can see it is important to consider both of the spatial and temporal information when analyzing the given trajectories. As a result, we show that DivCluST can produce better profiling results.

2.6.4 Analysis of DivST

In this section, we first show the impact of different th_{len} and th_{spd} in DivST followed by the comparison with the approximate MDL method used in [19].

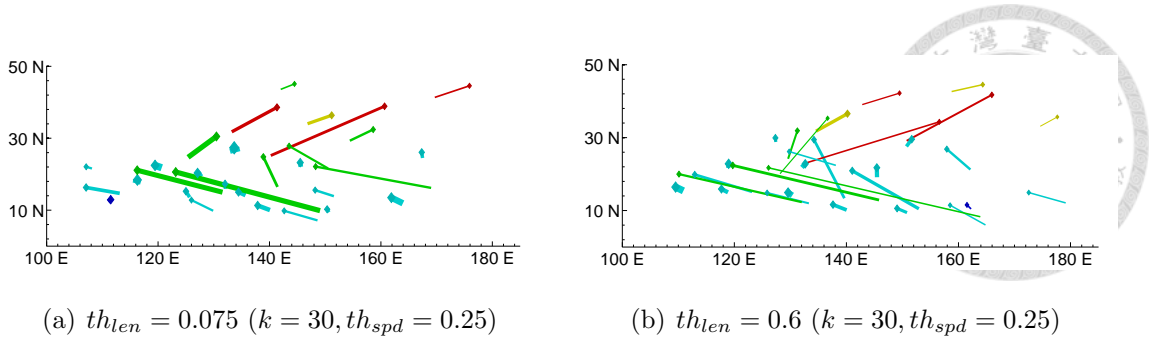


Figure 2.16: Impacts of th_{len} for typhoon tracks profiling.

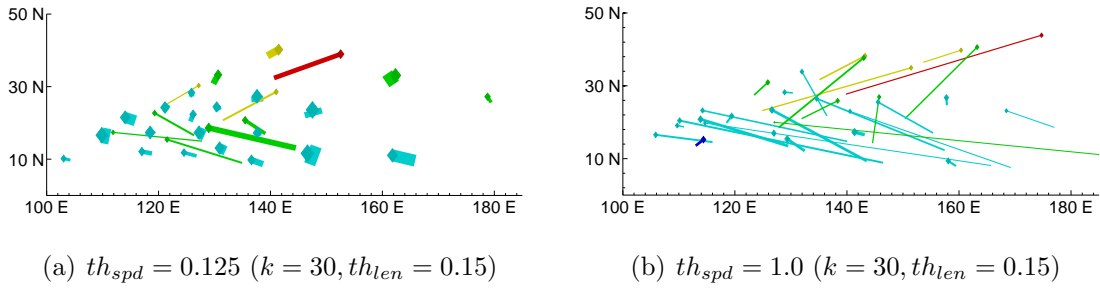
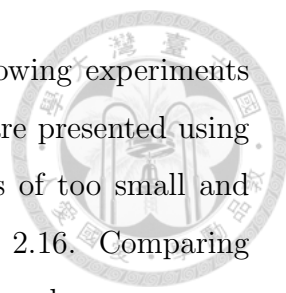


Figure 2.17: Impacts of th_{spd} for typhoon tracks profiling.

As we stated in Section 2.4.2, the threshold selection is a tradeoff between preciseness and conciseness according to (2.4). Therefore, we show the results of $R_{precise} + R_{concise}$ over different threshold values on the two real data sets in Fig. 2.15. Both w_{sht} and w_{spd} were set to one in this and all of the following experiments. The minimum values were reached at quite different threshold settings in the two types of trajectories. For the typhoon track data, as shown in Fig. 2.15(a), we found that $R_{precise} + R_{concise}$ reached its minimum when both th_{len} and th_{spd} were around 0.2 ($th_{len} = 0.15$ and $th_{spd} = 0.25$). Both of the two thresholds were small, which hints that the trajectory data under study changed steadily in both speed and direction. In fact, a typhoon indeed rarely changes its speed or direction abruptly. In the meantime, the minimum value of $R_{precise} + R_{concise}$ for the bus trajectory data was attained when $th_{len} = 0.1$ and $th_{spd} = 0.9$ as shown in Fig. 2.15(b). This hints moving paths of buses were smooth along the same roads (only with significant changes when turning at crossroads), while the speeds could vary extremely between moves and stops. This is why the selected th_{spd} was large.

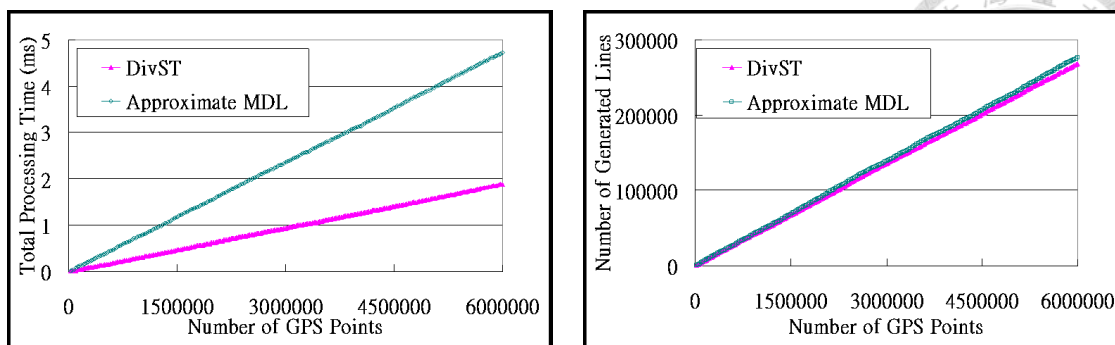
Here we show what would happen if we do not choose the thresholds as suggested by conducting some experiments on the typhoon track data. Without specifying



specific values, we by default set all the parameters in the following experiments the same as those used in Section 2.6.2. The clustering results are presented using the same legends as those in Fig. 2.12(a). The profiling results of too small and too large th_{len} (0.075 and 0.6, respectively) were shown in Fig. 2.16. Comparing to the results in Fig. 2.12(a), with a too small th_{len} , the dividing phase was more sensitive to minor direction changes and thus we lost the information of longer sub-trajectories with similar forward speed and directions. For example, those cyan lines are relatively short in the bottom of Fig. 2.16(a). With a too large th_{len} , on the other hand, several green clusters at the low latitude area are longer in length and have larger longitude spread than those of Fig. 2.16(b). The profiling results of too small and too large th_{spd} (0.125 and 1.0, respectively) were shown in Fig. 2.17. As th_{spd} decreased, the trajectory dividing phase would be more sensitive to the speed variance. In Fig. 2.17(a), we see clusters of different speeds but all the mean lines were short. It is because many short replacement lines were generated in the dividing phase due to the small th_{spd} . In contrast, when th_{spd} was large as shown in Fig. 2.17(b), most trajectories were not easily divided and we see many long cluster mean lines. At the bottom of Fig. 2.17(b), there are many cluster lines with similar speeds, comparing to Fig. 2.12(a). The above results verified what we discussed in Section 2.4.2. We have to properly select th_{len} and th_{spd} for effectively dividing trajectories.

Next, we show how DivST can work better in dividing the trajectories when compared with the approximate MDL method described in [19]. We selected trajectories including up to 6,000,000 GPS points from the taxi data sets. We implemented the approximate MDL method by adopting our spatial distance functions and adding in the temporal measurements. The $L(H)$, which described the compressed trajectory, is defined as the summation of the lengths of all replacement lines, plus the sum of d_{spd} between each element line and its corresponding replacement line. As to $L(D|H)$, which described the difference between the compressed and the raw trajectories, we use summation of d_{al} between each element line and its relative portion of the candidate replacement line, plus the sum of d_{spd} between the two.

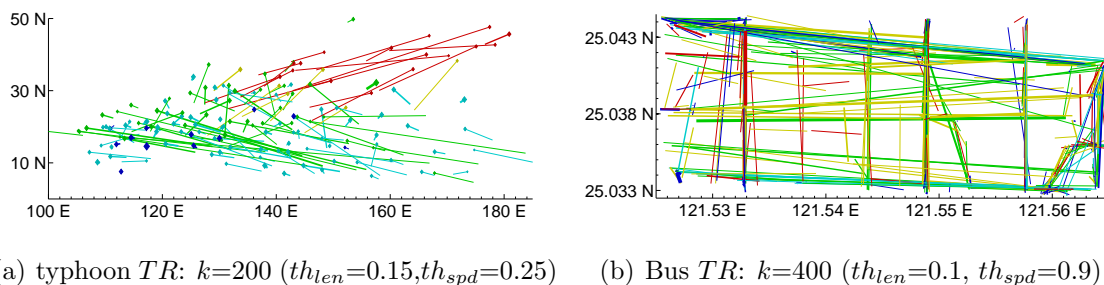
We show the accumulated processing time to divide whole trajectory set and the generated line numbers from its first record to the last in Fig. 2.18(a) and



(a) Time consuming compare.

(b) Generated line compare.

Figure 2.18: Comparison of DivST and the approximate MDL.



(a) typhoon TR : $k=200$ ($th_{len}=0.15, th_{spd}=0.25$)

(b) Bus TR : $k=400$ ($th_{len}=0.1, th_{spd}=0.9$)

Figure 2.19: *CluST* results with too large k .

Fig. 2.18(b), respectively. Under the proper threshold selection of DivST, where $th_{len} = 0.05$ and $th_{spd} = 0.3$, we can see the approximate MDL method was much slower when similar numbers of replacement lines were generated. This showed that DivST was able to get the same compression quality with a smaller time cost. Moreover, as we mentioned earlier in Sec. 2.4, DivST had better flexibility to adapt different compression ratios in consideration of the memory space limit.

2.6.5 Analysis of CluST

One main challenge of the k-means based clustering is the decision of k . Here we discuss the impact of k on the profiling and provide some guidelines for choosing k .

Based on (2.12), k between 30 and 55 was recommended for the typhoon track data, and k from 60 to 160 was suggested for the bus trajectory data. After we can decide a range of k based on some clustering validation measurements as in [50], the exact k selection should base on the domain expertise of the data. In our experiments, for typhoon tracks, clusters should tell different parts of the moving

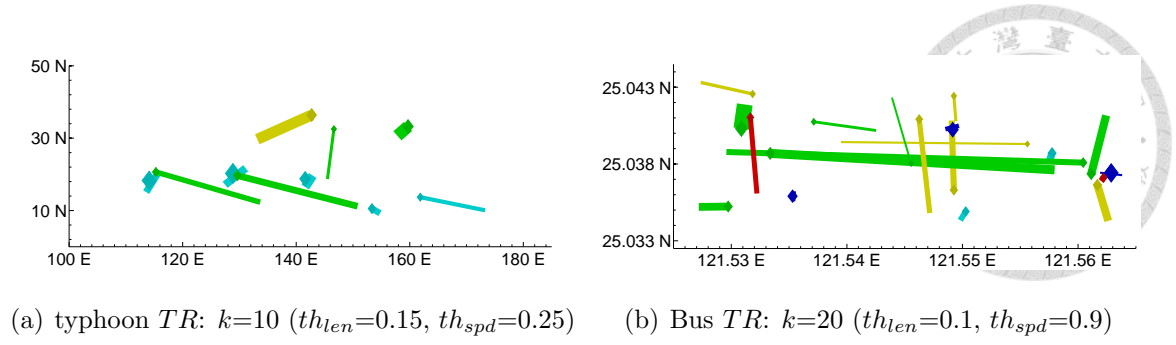


Figure 2.20: *CluST* results with too small k .

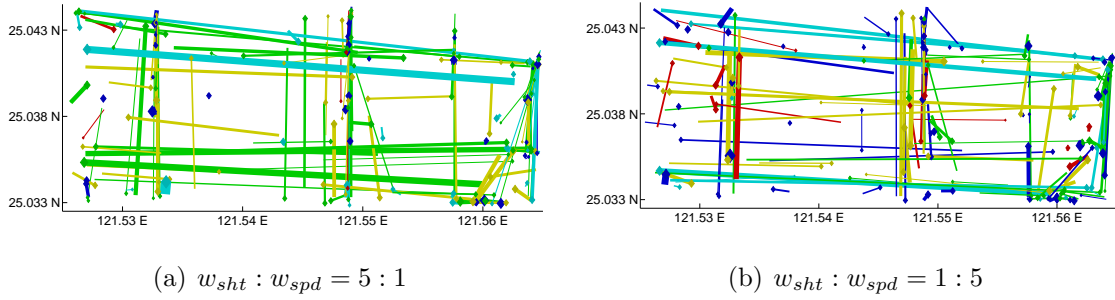


Figure 2.21: Impacts of w_{sht} and w_{spd} on bus data profiling.

characteristics along their paths of typhoons from low latitudes to high latitudes and distinguish different patterns. For bus trajectories, the number of clusters must be large enough to separate traffic on different segments of roads and distinguish those with different speeds on one road.

Finally, we show the clustering results if too small or too large k was used in Fig. 2.19 and Fig. 2.20 on both the typhoon tracks and the bus trajectories. When k was too large as shown in Fig. 2.19, too many similar and close-by clusters were generated, which gave us too many redundant information. On the other hand, when k was too small, many basic typhoon track moving styles vanished as shown in Fig. 2.20(a) while the bus routes that were originally on different roads were grouped into one cluster as shown in Fig. 2.20(b).

The weights in distance function (2.5) can be adjusted to strengthen either of spatial or temporal information. The impact of w_{sht} can be shown by comparing Fig. 2.13(a), Fig. 2.21(a) and Fig. 2.13(f), where the $w_{sht} : w_{spd} = 1 : 1$, $5 : 1$ and $1 : 0$, respectively. The speed diversity decreased as ratio of w_{sht} increased, while the generated clusters fit better into the real road network. In the mean time, the impact of w_{spd} is shown through comparing Fig. 2.13(a), Fig. 2.21(b) and Fig. 2.13(g), where

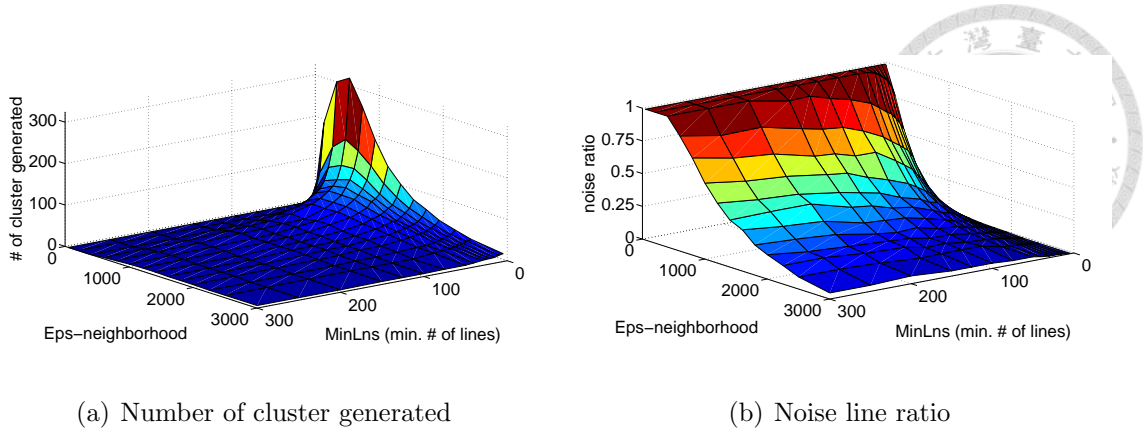


Figure 2.22: DBSCAN parameters analysis on typhoon data.

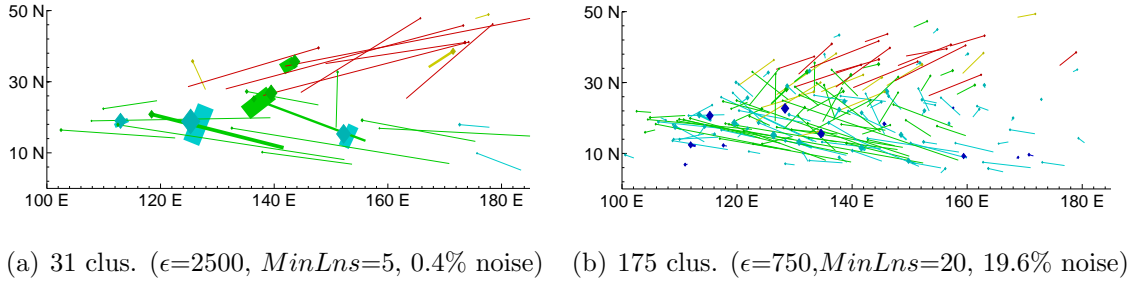
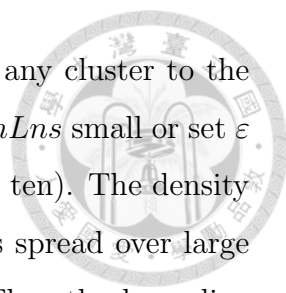


Figure 2.23: DBSCAN representation lines on typhoon data.

the $w_{sht} : w_{spd} = 1 : 1, 1 : 5$ and $0 : 1$, respectively. The speed diversity increased as ratio of w_{spd} got larger, while the generated clusters distributed more randomly and the original road network was totally vanished when $w_{sht} = 0$. With the above observations, the w_{sht} and w_{spd} should be adjusted according to the importance of the spatial and speed information.

Based on the same replacement lines generated by DivST, here we show the impact of different clustering methods on the profiling results. We compare our k -means based clustering method, CluST, with the adapted DBSCAN algorithm used in [19] on the typhoon track data set. As discussed in Section 2.5, the model-based clustering methods require a data-dependent design and thus we did not compare those algorithms with ours.

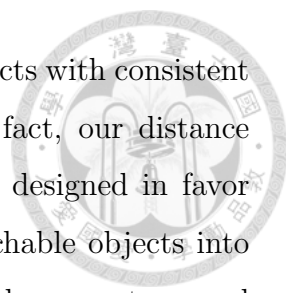
For the adapted DBSCAN method, we tested a series of ϵ -neighborhood and $MinLns$ (originates from $MinPts$ in the DBSCAN method [54], which is the minimum number of lines in our case) values, while using the same line distance defined in (2.5). As the parameters ϵ and $MinLns$ varied, we plotted the number of clusters



and the noise ratio (the ratio of lines that were not included in any cluster to the total number of replacement lines) in Fig. 2.22. When we set $MinLns$ small or set ϵ large, the number of generated clusters was very small (less than ten). The density reachable characteristic of DBSCAN, which tends to group lines spread over large area, is inflicted on generating few but large in size clusters. Thus the large line cluster cannot be interpreted simply by computing its mean. To solve this problem, an additional afterward cluster sweep was used in [19], which took a lot of time to compute the average coordinate whenever the number of line segments encountered was more than $MinLns$. On the other hand, although the generated cluster number increased rapidly as ϵ -neighborhood was smaller, the noise line ratio also increased fast as illustrated in Fig. 2.22(b), which would cause serious information loss. Furthermore, k -means has $O(n+k)$ memory cost and $O(nkt)$ time complexity, where n is the number of objects to be clustered, t is the iteration number. Usually $k, t \ll n$, so the k -means-based method is more scalable and efficient in processing huge amounts of trajectory data compared to DBSCAN, which has $O(n^2)$ complexity in both time and space.

To be clearer about the clustering results of DBSCAN, we provided sample results under two sets of parameter settings on the typhoon data in Fig. 2.23. First we set $\epsilon = 2500$ and $MinLns = 5$ so that the number of generated cluster is 31, which is close to the cluster number, $k=30$, used in our CluST method in previous sections. Under similar number of clusters, the profiling results of DBSCAN, as shown in Fig. 2.23(a), lost some important regional characteristics compared to those in Fig. 2.12(a). For example, the delicate turning points of trajectories vanished from several large clusters generated by the density reachable property of the clustering method (represented by those very wide short lines). To capture more details, we used another settings: $\epsilon = 750$ and $MinLns = 20$. The chosen ϵ was the average intra-cluster distances in Fig. 2.12(a), while this $MinLns$ caused a tolerable noise ratio, 19.6%. The resulting cluster number was 175 and the corresponding cluster mean lines were shown in Fig. 2.23(b). We can see that too many clusters were generated, which missed our goal of finding distinct moving patterns with a clear view.

The above results showed that the DBSCAN-based method was not suitable for



our goal to cluster sub-trajectories: to find convex clusters of objects with consistent spatiotemporal properties, as we discussed in Section 2.5. In fact, our distance function and the way of computing cluster mean lines were all designed in favor of convex clusters. The DBSCAN method connects density-reachable objects into one group and may included members spreading over large spatial area or temporal differences, which can result in less consistent spatiotemporal properties. Thus, the DBSCAN method was unfavorable to reveal the regional moving behaviors of trajectories in our design. This explains why we chose the k-means based clustering method in DivCluST.

2.7 Summary

In this work, we presented *DivCluST*, an approach to profiling a set of moving objects by dividing and clustering their accumulated trajectories spatiotemporally. First, DivST divided the trajectories and generated a proper number of replacement lines while preserving their original spatiotemporal properties well, through the designed spatial and temporal dividing criteria and the threshold parameter selections. Then, CluST clustered the replacement lines based on the proposed spatiotemporal line distance measurement in consideration of the differences between positions, lengths, directions and speeds. With the specially designed mean line representation, the clustering results can reveal the regional main spatiotemporal moving behaviors of the given trajectory data set. Finally, we conducted extensive experiments on three different real world trajectory data sets. The results showed that DivCluST can effectively produce the profiles of moving objects and identify regional typical moving styles from the trajectory data.





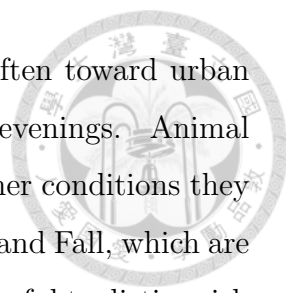
Chapter 3

Trajectory Warehousing for Multi-Granularity Moving Pattern Analysis

3.1 Introduction

Nowadays, various kinds of devices, equipping positioning techniques such as GPS, are available for tracking objects at any time from anywhere. The tracked position and time records of objects construct their moving trajectories. Huge amount of object trajectories, such as vehicle movements or animal migrations, are collected. Moving behaviors of objects are hidden in their trajectories. The behaviors worth to be extracted from trajectory data for further analysis. Learning typical vehicle moving speeds and traffic flow amounts in various sections can help analyzing and improving road design or traffic control. Examining about animal moving patterns can help checking species annually migration or environment changing conditions. In order to find out worthwhile knowledge, some earlier works clustered or classified trajectory data and found some meaningful information, such as main moving paths [19], overall behavior sketches [55], or distinguishing object groups [40]. However, these works only provided analysis mainly on static trajectory data and cannot reflect the evolving change of moving behaviors.

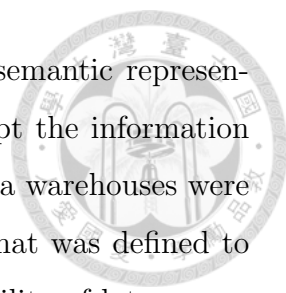
As we know, object moving behaviors vary in different periods of time. For example, moving speeds of vehicles in rush hours can be much slower than those



during midnight in downtown area. Traffic flow directions are often toward urban area in workday mornings and back to suburb of city in the evenings. Animal migration paths and spent times are influenced by different weather conditions they encountered every year. Typhoon moving styles differ in Summer and Fall, which are effected by the global climates. It would be interesting and meaningful to distinguish and compare patterns happened in different periods of time. Also, more knowledge can be extracted through analyzing moving pattern evolvement in a long term basis, to know about the behavior changes and possible influences.

To organize trajectory information for long, storing the extracted moving patterns into a data warehouse would be a good choice. Through systematically summarizing trajectory data, a warehouse can provide long term storage for moving patterns, easy information access through queries, and possible other applications that need online analytic of object behaviors. A moving pattern should include static attributes, such as representing amount, as well as spatiotemporal properties, like lay position and valid time, for various possible applications. The application possibilities include to build some moving behavior templates and use them for abnormal detection or special event analysis, or to search for pattern changing trends and find temporal relationships or predict possible next moving. The analytic ought to be provided on any large or small area that trajectories spreading, or to process a snapshot in single time period as well as analyze long term behavior condition of a location. Moreover, using some further processes, we can find relations between moving patterns and other spatial, temporal, or specific domain information.

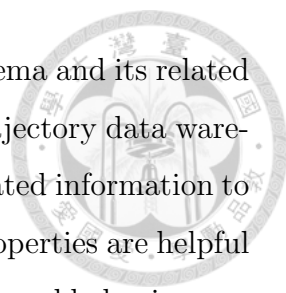
Some existing trajectory data warehousing works focused on statistics over regular-sized spatial cells [23] [24], amount of trajectories passing each cell and average values of other properties like speed were recorded in the tables. In these cell-based work, the moving paths, length spans, and various forward directions of trajectories could not be shown by the kept information in corresponding data warehouse. The second group of works reconstructed trajectories by simplifying them into a data point on each visited region of interest (RoI) [21] [22]. Only information on the previously defined regions were analyzed, and frequent patterns were stored in the data warehouse. In these RoI-based works, trajectory behaviors outside previously selected regions were ignored directly, which caused obvious information loss. Still another



group of works transformed trajectories into different kinds of semantic representations, such as streets or landmarks with descriptions, then kept the information in a data warehouse [25] [26]. The semantic-based trajectory data warehouses were mostly designed for specific applications. Each information format was defined to fit the previously determined application, which limit the possibility of later usage changing requirements.

Since a data warehouse system aims to support data analyzing and decision making, the information stored in should be processed and transformed into a general format. For a trajectory data warehouse, the first challenge is what characteristics from raw trajectories should be kept in a generated moving pattern. Information of moving such as length, direction and speed are hidden behind the recorded positions and times. Also, spatiotemporal characteristics of trajectory data are closely related to each other. For example, moving lengths are decided by positions and forward speeds goes with directions. Thus, these properties should be kept as a set rather than be separately dealt with. We settle the moving pattern format based on a directional line, which keeps position, direction and length of spatial domain, along with speed and time information. Meanwhile, for easier pattern processing, the representative value in each dimension of moving patterns should be defined as a regular range.

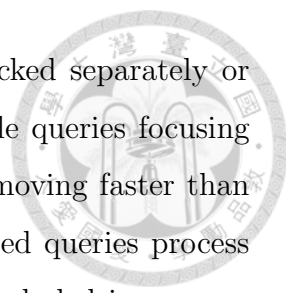
The second challenge is how to extract moving patterns from raw trajectories that are incoming boundlessly with a huge amount and spreading in a wide area. To deal with continuous incoming trajectories, an online algorithm is needed to process arriving data points and extract typical moving behaviors among them. The behavior extraction helps eliminating redundant portions existed in huge amount of original data. Moving patterns generated by format transformation are then stored into the trajectory data warehouse. We would like to reveal moving patterns from each period of time and include all areas where the trajectories spread. Unlike some earlier works focused only on some predefined regions, we believe behaviors from all trajectory spreading areas worth being revealed. Interesting information may not always happen on known regions, while we do not always know enough to decide where are important in overall area. Furthermore, the process of pattern extraction should meet the requirements of their format and predefined attribute ranges.



The third challenge is design of trajectory data warehouse schema and its related operations for queries or other processes. The table schema in trajectory data warehouse should fit the format of moving patterns and allow their related information to be stored. Query operations designed for spatial and temporal properties are helpful to deal with the moving pattern retrievals. Also, as both new and aged behaviors are kept in the warehouse, moving pattern updating process for long term maintenance is required. Furthermore, since different part of a trajectory may be included in multiple patterns appearing in a large area, we need a method to estimate distinct objects that contained in patterns relating to an area. With pattern operations and other related information integrated in the data warehouse, we can analyze and use the moving patterns in various applications.

In this work, we choose to build our trajectory data warehouse based on extracting typical moving patterns of overall object trajectories. The original trajectories are online processed and examined about their consistency in directions and speeds with spatiotemporal criteria, which work with the thresholds on changes between each pair of consecutive position and time records. After dividing trajectories according to the criteria, direction and speed consistent segments can be transformed into replacement lines. The lines remain to contain direction and speed properties of original trajectory divisions and constraint the space requirement in the meantime. After a period of time, the collected replacement lines are processed, and typical moving behaviors among them are summarized into group representation lines. The behavior extraction process works on regular-sized multidimensional grids, and group representation lines are transformed into moving patterns if the corresponding grids contain enough amount of replacement lines. Using regular-sized grids helps constraint the property differences in each group and limit the information loss of generated patterns, and make the design of pattern operations of data warehouse more straight forward. After all the above steps, moving patterns and related information are kept in the trajectory data warehouse.

Since moving patterns storing in the data warehouse still keep their spatial and temporal properties, the available query styles are no longer limited on checking only average statuses of trajectories in a cell, or only focusing on limited regions as earlier works. The moving patterns starting from, ending in, or passing through different



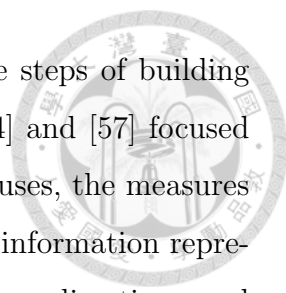
areas, and happen during different time periods can all be checked separately or mixed in respond to query requirements. Which includes, simple queries focusing on one or multiple attribute values, like searching for patterns moving faster than a speed or having members above certain amount. And advanced queries process on spatial and temporal domains, through analyzing patterns included in an area and a period of time. Furthermore, series changes of moving behaviors or frequent patterns over a long period can also be queried. Through aggregation operation on space or time dimensions with different grid granularity, pattern summarization and aged pattern maintenance can be done in an efficient way. With the records of trajectory flows, distinct trajectory amount including in queried patterns can also be properly estimated. The data warehouse can further include other spatial, temporal or specific domain databases, and provide variety of analytic processes related to moving patterns.

The remainder of this Chapter is organized as follows. We discuss the related works in Section 3.2. Then we introduce preliminaries of work and several definitions in Section 3.3. Our proposed approach to extract moving patterns from trajectories is described in Section 3.4. The warehouse queries and operations are illustrated in Section 3.5. Section 3.6 presents the experiment results and analyzes the algorithms and operations on real data sets. We conclude our work in Section 3.7.

3.2 Related Works

Trajectories need to be processed and transformed, before the information from raw data set is ready to be stored into a data warehouse. Among the existing trajectory data warehousing works, there are three major groups of design ideas. One is cell-based, the other is RoI-based, and still another is semantic-based trajectory data warehouse.

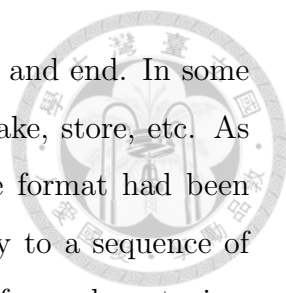
A *cell-based* trajectory data warehouse mainly focused on checking the amount and averaged properties of trajectories passing over each cell [23] [24]. These works first reconstructed trajectories according to the uniform-sized cells they passed over. Then numeric measures on the reconstructed trajectories are processed per cells and stored in the warehouse. The aggregation operations worked as value summation or



averaging among spatial and temporal dimensions. [23] gave the steps of building a cell-based trajectory data warehouse and [56] visualized it. [24] and [57] focused more on the aggregation processes. In this group of data warehouses, the measures worked only in each cell as sum amount or averaged values. The information representing is limited by the size and boundary of cell. The various directions and speeds of trajectories passing the same cells cannot be presented separately, thus would mislead the overall picture in an area where several object moving behaviors intersects. Furthermore, the length spans or moving ranges of trajectories cannot be shown by the measures in cells, which is quite a loss of original information. Our work can keep more trajectory properties by allowing both speed and direction variances of an intersection area to be shown. Moreover, a line format moving pattern can show more clear spatial moving conditions with its start, end positions and length spans.

A *RoI-based* trajectory data warehouse cared about "interested" regions much more than the other areas [21] [22]. Before processing trajectory data, the interested regions ought to be defined first. Then, each raw trajectory was reconstructed into a sequence of RoIs, where the interested regions were passed by the trajectory. After reconstructing trajectories, the data warehouse then recorded those found frequent visited RoI sets. Related query operations of RoI-based warehouse mainly focused on frequent visited region groups in [21]. Data reduction by selecting regions and provide efficient querying via encoding regions in [58]. In a RoI-based trajectory data warehouses, data points and moving conditions not happened in any interested region were ignored directly. Though the space requirement was low as trajectories being simplified to limited points, the information loss would be quite high. The need of defining interested regions in advance would also require sufficient previous knowledge over trajectory passing areas. Whether the interested locations are properly selected would influence the information quality of a RoI-based trajectory data warehouse. Our design analyzes all of the trajectory spreading area, while patterns would automatically be generated from frequent visited area. This can decrease information loss and avoid improper selection of interested area.

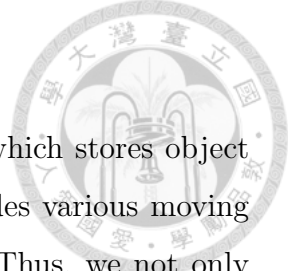
A *semantic-based* trajectory data warehouse transformed trajectories into conceptual information according to specific application requirements. [25] [26] Trajectories



were recorded with moving characteristics like stop, move, begin and end. In some works, records also mapped to geographical locations like city, lake, store, etc. As trajectory paths were turned into semantic representations, the format had been designed to fit a specific application. [59] transformed trajectory to a sequence of spatiotemporal distances between consecutive spots. And [60] focused on trajectory location management and transformed routes according to map. [61] focused on combining domain and spatial ontology represented object activities and spatial relationship of semantic trajectories. The information stored in a semantic-based trajectory data warehouse is pretty condensed. However, before building a semantic-based warehouse, the application scenario must be clearly sketched first. On the other hand, our design provides a general solution which does not require scenario sketch in advance, and is open for various kinds of application to be developed later.

There were also some other trajectory database works focused on several specific targets. [62] generated spatiotemporal pattern for moving objects based on frequent item set among groups. [63] aimed only on group-by operations of trajectories, including overlap, intersection, or both. [64] partitioned the original trajectories with regions and identified preferring directions in each. Then it encoded the moving paths into tree structures, and provided for amount counting and querying. [65] queried trajectories through filtering region search for approximated trajectory segments using octagon bounding, then refined evaluation and additional conditions trace. [66] sampled and modeled uncertainty via beads of trajectory, and provided model based queries. [67] designed a database keeping past, current, future location information of moving objects using grid based indexing, and supported point, range and kNN queries. While this quite early work provided a full structure, it could hardly deal with today's huge amount of trajectory data.

Our trajectory data warehouse design stores patterns in format with attributes properly representing moving behaviors. It covers all areas in which trajectories spread, while it can also reveal the variance in a small region. Moreover, it can shown moving between locations without defining them in advance. Last but not least, it allows flexible future possibilities of application development.



3.3 Preliminaries

Our goal of this work is to build a trajectory data warehouse, which stores object moving patterns extracted from raw trajectory data, and provides various moving behavior sketches in response to different query requirements. Thus, we not only have to design the structure of data warehouse and its related operations for pattern query and maintenance, but also have to propose algorithms to extract moving behaviors from raw trajectory data and to transform the information into patterns which can represent the typical parts of the originals.

The process of moving pattern extraction for data warehouse starts from analyzing raw trajectories of objects.

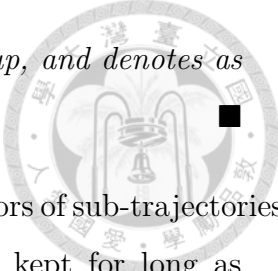
Definition 3.1 (trajectory). *An object trajectory TR is composed of a sequence of moving records, which are data points including positions and corresponding times, $pt : (p, t)$.* ■

Since each trajectory has various length and contains different direction and speed changes, it is hard to define and measure similarity between whole trajectories. Meanwhile, as positions and times are recorded with frequent interval, it cost much to store all of the recorded data points and analyze them repeatedly in each query. Instead, it makes more sense to summarize trajectories, and then try to find similar moving behaviors from the condensed data, especially when trajectories are divided into homogeneous direction and speed segments.

Definition 3.2 (Replacement line). *When a segment of trajectory with consistence speed and direction characteristics is divided, a replacement line is generated as a directional line connecting from the point with its earliest time-stamp of the segment to the one with latest time-stamp, and denotes as $L_R : (pt_s, pt_e)$.* ■

The L_{RS} can now properly replace corresponding sub-trajectories and condense the kept information. Through grouping similar replacement lines considering their positions, lengths, directions and speeds, common moving behaviors among trajectory divisions can be found.

Definition 3.3 (Group representation line). *When certain amount of replacement lines have similar moving behaviors, they become a group G . A group representation*

line is used to represent all replacement lines included in the group, and denotes as $L_G : (p_s, p_e, s)$. 

The extracted L_G s, which representing common moving behaviors of sub-trajectories in a group, need to be transformed into a regular format and kept for long as patterns.

Definition 3.4 (Moving pattern). *Each L_G is transformed into a moving pattern, which is described by properties of L_G , including its start, p_s , and end, p_e , positions, forward direction, \vec{a} , span length, d_L , moving speed, s_L and the amount of group, $|G|$, along with its valid time period.* ■

The moving patterns are stored along with other related information, such as grouping parameters and time periods, in a data warehouse.

Definition 3.5 (Trajectory data warehouse). *A trajectory data warehouse stores current as well as historical moving patterns extracted from trajectories. The repository integrates information related to object moving patterns through extract, transform and load, and can be used for data analysis and decision support.* ■

With the above definitions, we can now state the main steps, extract, transform and load, of building a trajectory data warehouse.

In the *extraction* step, we convert incoming trajectory data into corresponding line segments, which represent their moving behaviors, on-the-fly. For each given TR , we would like to divide it into a set of sub-trajectories with spatial and temporal dividing criteria. The criteria set thresholds on the variation of directions and speeds, and are used to check a raw trajectory point by point. The dividing algorithm processes the boundless in-coming position and time records online, and generate L_{RS} without the need of storing any raw trajectory point information for long or parsing it twice.

In the *transform* step, extracted replacement lines are summarized and turned into moving patterns. The L_{RS} are analyzed per time period, and corresponding L_G s of each period are generated based on regular-sized multidimensional grids, and transformed into moving patterns. We propose how to find common behaviors from L_{RS} , and how to generate L_G s. Meanwhile, the information included in a

pattern is constrained by a multidimensional grid, thus has limited uncertainty and information loss.

In the *load* step, moving patterns extracted and transformed from trajectories are stored in the trajectory data warehouse. The table schema is designed to fit the pattern generating steps, and the attributes are for outputs of each part, as shown in Fig. 3.1. Before moving patterns are generated, table `divide check` keeps the latest trajectory check conditions and table `replacement line` keeps the extracted behaviors. These two tables only keep information temporarily during processes of pattern extraction.

Next, characteristics of a moving pattern, including position, length, speed, direction, time and amount of representation lines, are stored in table `moving pattern` along with its time period identity after transformation. Then, table `time grid` registers the start and end time of each period and its corresponding granularity level of a multidimensional grid. Group extraction in each time period uses regular parameter ranges defined previously for different grid granularity. The related parameter ranges of different granularity level for each grid dimension are saved in table `grid unit`, which keep basic unit records and decides element grids. These parameters can be used for historical moving pattern maintenance as well as query operations. In table `flow trace`, the flow amounts from group to group are recorded through tracing the belonging groups of originally continuous replacement lines. The flow information is used to help distinct trajectory estimation when querying patterns from a larger area.

The data warehouse table schema is designed to deal with online analytic processing, as well as to keep long term information. So the schema need to be designed as easy use for both continuous record insert and historical maintenance update. Beyond simple value queries, some operations which specially focusing on the spatial and temporal properties are needed for well analyzing moving patterns, since a moving always happens during some time and in some area. The aggregation operation worked on one or several dimensions also need to be designed, in order to help summarize queried moving patterns and maintain historical patterns. Still another computation need to be tackled on is distinct trajectory flow amount estimation. Which is to measure the amount of replacement lines in different patterns of an area

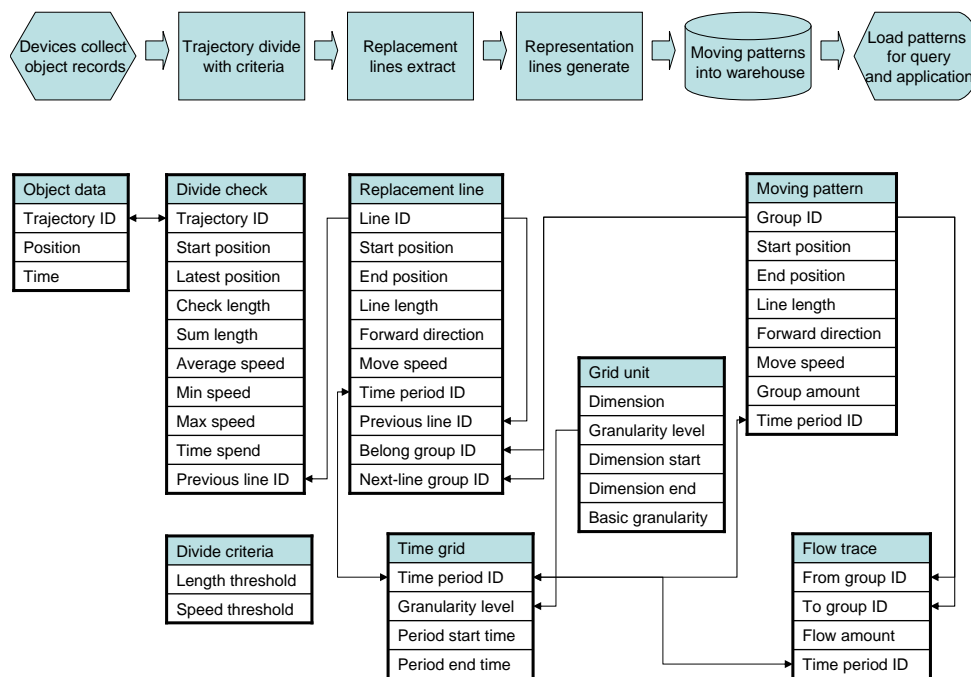
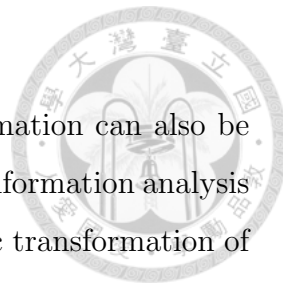


Figure 3.1: Process of building trajectory data warehouse and related table schema.

originated from same trajectories.

Other related spatial, temporal or else domain specific information can also be stored into the trajectory data warehouse, and used in advanced information analysis and further applications. Such as geographical maps for semantic transformation of patterns, weather reports for moving behavior change under different condition, or road incident lists for afterward influences of car accidents or traffic controls.



3.4 Extracting Moving Patterns

We want to build a trajectory data warehouse which stores moving patterns extracted from objects' continuous position and time records. Objects move from locations to locations with various paths, directions and different speeds. Thus, it is hard to measure similarity between irregular-shaped whole object trajectories. So, we decide to divide the original trajectories into segments of direction and speed consistency sub-trajectories. Then, search for similar moving behaviors among the divisions. Afterward, we transform the common moving behaviors among sub-trajectories into moving patterns.

To extract moving patterns, we have to deal with huge amount of continuously and boundlessly incoming data points, and condense the size of stored information. In the mean time, the extracted moving patterns should preserve major moving characteristics of original trajectory data. We propose a two-stage process to transform the raw data into condensed and meaningful moving patterns for the trajectory data warehouse. In stage one, we deal with arriving data points of trajectories simultaneously and on-the-fly, through evaluating the consistency of incoming points and generating replacement lines from divided sub-trajectories. In stage two, we deal with the replacement lines, by processing to reveal similar ones and generating moving patterns. These steps both condense the information amount to be stored into the trajectory data warehouse.

3.4.1 Trajectory Consistency Dividing

In the first stage of moving pattern extraction, we divide each trajectory into direction and speed consistency sub-trajectories. Then the divisions of trajectories are

transformed into replacement lines, which can efficiently limit the storage space requirement and provide regular format data for further processing. We adapt the trajectory dividing method used in Section 2.4, with some modifications in the criteria setting to make it parse each data point record only once. We also update Algorithm 2.1 to process data points from multiple trajectories at the same time. The incoming data points are dealt as soon as they arrived and simultaneously. The corresponding trajectory records in temporary table `divide check` are updated, and then the continuously arriving data points can be dropped without being saved. Once a newly incoming data point fails to meet the consistency check, the trajectory is divided and a replacement line is generated, then recorded into temporary table `replacement line` for next stage of process.

We propose a modified version of spatiotemporal criteria to check the consistency among continuous data points of a trajectory. The distance dividing criterion, th_{len} , as (3.1) is used to measure the length difference ratio between element length sum of current trajectory, d_{sum} , and related candidate replacement line, d_{chk} .

$$\frac{d_{sum} - d_{chk}}{d_{sum}} < th_{len}, \quad (3.1)$$

where

$$d_{sum} = \sum_{i=s}^{e-1} d(p_i, p_{i+1}), \text{ and}$$

$$d_{chk} = d(p_s, p_e).$$

$d(p_i, p_{i+1})$ in (3.1) is the geographical distance between p_i and p_{i+1} .

On the other hand, the speed dividing criterion, th_{spd} checks the largest variation of speed among current trajectory through tracing the maximum speed, s_{max} , and minimum one, s_{min} , between each pairs of continuous data points, and work as (3.2).

$$\frac{s_{max} - s_{min}}{s_{avg}} < th_{spd}, \quad (3.2)$$

where

$$s_{avg} = \frac{\sum_{i=s}^{e-1} s(pt_i, pt_{i+1})}{e - s},$$

$$s_{max} = \max_{i=s}^{e-1} [s(pt_i, pt_{i+1})], \text{ and}$$

$$s_{min} = \min_{i=s}^{e-1} [s(pt_i, pt_{i+1})].$$

$s(pt_i, pt_{i+1})$ in (3.2) is moving speed between two data points, computing as $d(p_i, p_{i+1})/(t_{i+1} - t_i)$.

In order to deal with large amount of incoming trajectory data points, the dividing algorithm is designed to work online with minimum information kept. When first point pt_{i1} of trajectory TR_i arriving, a corresponding record is set up. The attributes of record TR_i are initialized in table `divide check` as follow. Both of the start pt_{is} and end pt_{ie} points of candidate L_R are assigned as pt_{i1} . Then record attributes $d_{sum} = d_{chk} = 0$ and $\Delta t = 0$, while $s_{min} = s_{max} = s_{avg} = NULL$. As pt_{i2} arriving, pt_e is assigned as pt_{i2} , and update $d_{sum} = d_{chk} = d(p_{i1}, p_{i2})$, $s_{min} = s_{max} = s_{avg} = s(pt_{i1}, pt_{i2})$, and $\Delta t = t_{i2} - t_{i1}$.

As more data points of TR_i arriving, the spatial and temporal criteria are tested, and corresponding actions are taken according to the results. The newest element line $L_E : (pt_{ie}, pt_{i(e+1)})$ is used to test criteria (3.1) and (3.2) with reference of attributes in record TR_i , where $d_{sum} = d_{sum} + d(p_{ie}, p_{i(e+1)})$, $d_{chk} = d(p_{is}, p_{i(e+1)})$, $s_{avg} = (s_{avg} * (e - s) + s(pt_{ie}, pt_{i(e+1)})) / ((e + 1) - s)$, $s_{max} = \max[s_{max}, s(pt_{ie}, pt_{i(e+1)})]$ and $s_{min} = \min[s_{min}, s(pt_{ie}, pt_{i(e+1)})]$. When both criteria are met, attributes computed above are updated into record TR_i . Then, $pt_{i(e+1)}$ replaces pt_{ie} , and renews $\Delta t = t_{i(e+1)} - t_{is}$. If any of the criteria is not fulfilled, a new replacement line $L_R : (pt_{is}, pt_{ie})$ is generated. Record TR_i is reset as $pt_{is} = pt_{ie}$, $pt_{ie} = pt_{i(e+1)}$, $d_{sum} = d_{chk} = d(p_{ie}, p_{i(e+1)})$, $s_{min} = s_{max} = s_{chk} = s(pt_{ie}, pt_{i(e+1)})$, $\Delta t = t_{i(e+1)} - t_{ie}$, then wait for the next data point incoming.

The full steps of dividing raw trajectory into spatiotemporal characteristic consistency segments is shown in Algorithm 3.1. This online algorithm takes $O(1)$ computation as each data point arriving. The space requirement for table `divide check` is $O(n)$, where n is the total number of trajectories having data incoming in the time period. The extracted L_R s are stored into table `replacement line` for next stage process.

In the dividing stage, some detailed moving information are eliminated while a trajectory is substituted with several replacement lines. The maximum information loss is controlled in a fixed ratio by the given th_{len} and th_{spd} as dividing criteria. When dealing with a more sensitive trajectory data, smaller th_{len} and th_{spd} can be used to constrain the information loss. Meanwhile, given larger th_{len} and th_{spd} can



Algorithm 3.1 Online trajectory consistency dividing

Input: incoming data points of trajectories, ex. pt_{in} of TR_i , dividing criteria th_{len} and th_{spd} , defined time period T_{ID}

Output: a set of L_R s for table replacement line

```
while new  $pt_{in}$  arriving during time period  $T_{ID}$  do
  if record  $TR_i$  not exist then
    Add record  $TR_i$ 
    Initialize attributes of record  $TR_i$ , with  $pt_{i1}$  and
    Set start and end points of candidate  $L_R : (pt_{is}, pt_{ie})$  as  $s = e = 1$ 
  else
    Test new element line added  $L_E : (pt_{ie}, pt_{i(e+1)})$ 
    if criterion (3.1) or (3.2) is not fulfilled then
      Output  $L_R : (pt_{is}, pt_{ie})$ 
      Set  $s = e$ ,  $e = e + 1$  for candidate  $L_R$ 
      Reset attributes of record  $TR_i$ 
    else
      Set  $e = e + 1$  for candidate  $L_R$ 
      Update attributes of record  $TR_i$ 
    end if
  end if
end while
After time period  $T_{ID}$  ended
while exist record  $TR_i$  with  $s \neq e$  do
  Output the last line  $L_R : (pt_{is}, pt_{ie})$ 
  Delete record  $TR_i$ 
end while
```

decrease the amount of extracted replacement lines. There is a trade off between information loss and data amount to handle in the following processes. As discussed in Section 2.4.2, th_{len} and th_{spd} should be decided through testing sample trajectory data and consider about available storage space.



3.4.2 Group Pattern Generating

In this stage, we process for revealing similar moving behaviors from replacement lines prepared in previous stage. The L_{RS} are distributed in large area with various lengths, directions, speeds, and are summarized per period of time. Clustering lines with similar properties, like in [19] or [55], is the first idea for revealing moving patterns. However, with typical clustering methods, the areas covered by clusters are often with quite different sizes. The large variances inside a cluster, in positions, speeds or others, are not bounded by a fixed value. In the mean time, without regular sizes, the distribution of clusters would be varied in different periods of time, and would cause limits on pattern comparison and processing. Furthermore, while operations as aggregation are needed in a data warehouse for query and maintenance, the moving patterns generated should be formatted in a regular base. Thus we decided to adapt the uniform cell idea from [24], and extend it into multidimensional *unit grids*.

Definition 3.6 (Unit grid). *A unit grid, gd_u , is decided with predefined basic granularity in each dimension. The value range of each dimension is sliced into equal parts. Then the units of dimensions format corresponding regular size grids.* ■

A unit grid constrains the ranges of attribute values inside a group. The possible attribute variance ranges between members in a group is limited by its basic granularity. We illustrate the dimensions of grids in Fig. 3.2, which shows the basic granularity in spatial domain (x, y) , temporal domain t , length d_L , direction \vec{a} , and speed s . The unit grids provide more straight forward way for aggregation and other moving pattern operations, without a need to deal with changing property ranges in any dimensions between patterns. Furthermore, the information loss during generating moving patterns is constrained by the assigned grid granularity, and can be clearly measured. Through periodically processing information in table `replacement line`, the group pattern extraction goes as Algorithm 3.2.

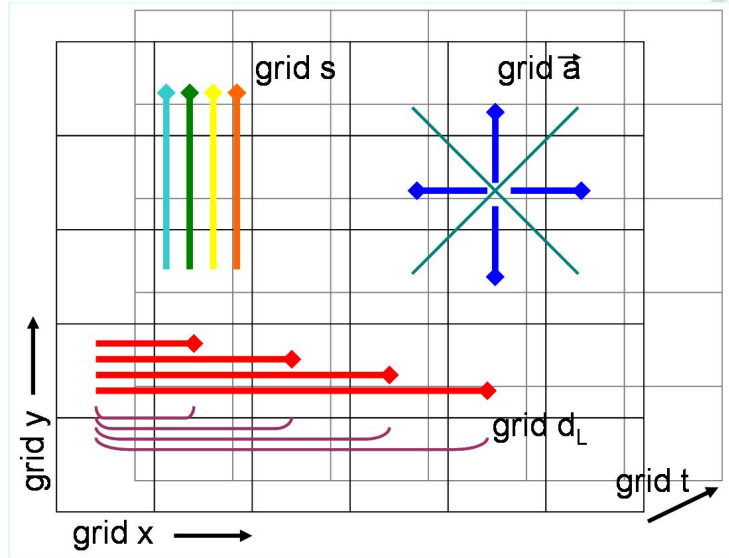


Figure 3.2: Illustration of multidimensional grids and granularity.

Algorithm 3.2 Multidimensional grid based moving pattern generating

Input: a set of L_{RS} from table replacement line, a set of unit grid gd_u in the space

Output: a set of records for table moving pattern

```

for each  $gd_u$  do
    Count  $L_{Rm}$  lay in the grid
    if criterion (3.3) is met then
        Compute  $L_G$ 
        Transform  $L_G$  to moving pattern
        Add pattern record into data warehouse
    end if
end for

```

The replacement lines extracted in stage one are input of stage two along with the unit grids for moving pattern extraction. In this stage, similar L_R s are grouped together when they lay in the same gd_u . An minimum line amount is used to decide whether a group should be generated in this gd_u . Then the group representation line L_G is computed from L_R s in the grid. The L_G s generated are transformed into regular moving pattern format and stored in the trajectory data warehouse. We choose to keep the amount of objects of a group, $|G|$, instead of any member lists, in a pattern. For we want to focus on the general moving behavior of objects, not to specifically deal with any single object trajectory.

The minimum line amount is designed to select moving patterns. Infrequent moving behaviors should not be recorded as patterns. We set the criterion as

$$\frac{|G|}{|L_R|} > th_{amt}, \text{ where } L_R \text{ happen during } T, \quad (3.3)$$

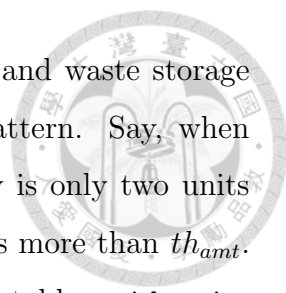
where T is the time period under processing, $|L_R|$ is the total amount of L_R s generated in this period. As (3.3) is met, the group representation line L_G are computed basing on L_R s following (3.4). The L_G computation find the mean length, direction and center of L_R s, along with (3.5), which decide the start, end points and speed of L_G .

$$\begin{aligned} c_{L_G} &= \frac{\sum_{L_R \in G} (p_{L_R,s} + p_{L_R,e})/2}{|G|}, \\ l_{L_G} &= \frac{\sum_{L_R \in G} d(p_{L_R,s}, p_{L_R,e})}{|G|}, \text{ and} \\ a_{L_G} &= \frac{\sum_{L_R \in G} ((p_{L_R,e} - p_{L_R,s})/d(p_{L_R,s}, p_{L_R,e}))}{|G|}, \end{aligned} \quad (3.4)$$

where $p_{L_R,s}$ and $p_{L_R,e}$ are the two end points of L_R , and $|G|$ is the number of lines in group G .

$$\begin{aligned} p_{G,s} &= c_{L_G} - \frac{l_{L_G} \times a_{L_G}}{2}, \\ p_{G,e} &= c_{L_G} + \frac{l_{L_G} \times a_{L_G}}{2}, \text{ and} \\ s_G &= \frac{\sum_{L_R \in G} s_R}{|G|}. \end{aligned} \quad (3.5)$$

The information of L_G and its group amount, are transformed and stored into table `moving pattern` for later usages, while recording the time period and grid granularity information into table `time grid`. We choose to record information



of each L_G instead of each gd_u to avoid keeping a huge matrix and waste storage space, for not every multidimensional grids would include a pattern. Say, when the direction dimension is divided into four quarters, possibility is only two units exist on a road. Also, not every unit area are traveled by objects more than th_{amt} . Meanwhile, the granularity of different dimensions are kept in table `grid unit`, and used in processing each period of L_{RS} . Through indexing attributes of grid dimensions, Algorithm 3.2 can be done in $O(m)$, where m is the amount of L_{RS} .

There are two sources of information loss as generating moving patterns. One is caused by the computation of L_G , and the other is by th_{amt} used in generating a group. Since the goal is to extract common moving patterns from a trajectory set, th_{amt} is used to eliminate those tracks not similar to other ones. The elimination is controlled by a ratio over $|L_R|$, where th_{amt} set the upper bound of L_{RS} being dropped. The pattern computation can not avoid some differences between original replacement lines L_{RS} and the corresponding group representation, L_G . Since composition of a group is limited by each unit grid, the maximum difference between L_G and L_{RS} is also limited by the granularity of gd_u . The largest difference between a L_R and corresponding L_G is the size of grid, which indicates they separately lay on the two ends of attribute ranges in all dimensions. Meanwhile, even in the worst case when all L_{RS} are spread among the edges of unit grid, the average difference between all L_{RS} to the corresponding L_G would not be larger than half of the granularity in each dimension.

In Fig. 3.3, we illustrated a case suffering both information loss during grouping L_{RS} . The original L_{RS} distributed in two grids as Fig. 3.3(a). Ideally, two L_G s, each has two and three L_{RS} as members, should be grouped as Fig. 3.3(b), when $th_{amt} = 2$. However, the real case is Fig. 3.3(c), four members are grouped with larger distance from origins in the lower grid, and the L_R of upper grid is dropped for not reaching th_{amt} . The gd_u can be adjusted to fit the data distribution and decrease the information loss caused by line grouping when dealing with object moves that changing more delicately. And th_{amt} should be assigned properly, so that the information eliminated by discarding those lines not belong to a pattern would not harm the result patterns found.

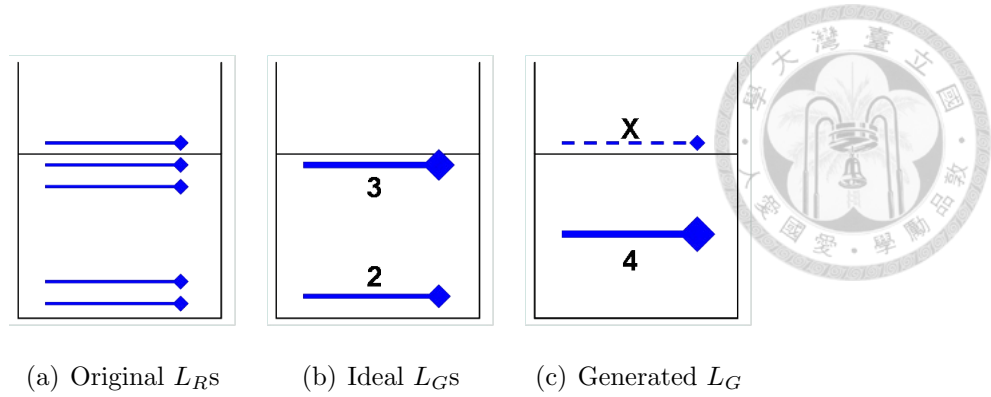
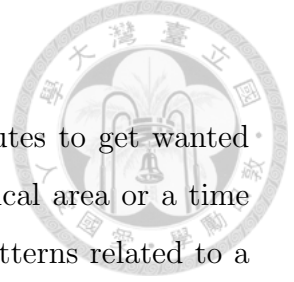


Figure 3.3: Example of information loss in grid grouping.

3.5 Warehouse Operations and Queries

After two stages of processing, the extracted moving patterns are stored into table `moving patten` of the trajectory data warehouse. The granularity of grids used in generating patterns are recorded in table `grid unit`, and the information of time periods in table `time grid`. These relational attributes would be retrieved together in information queries and moving pattern operations.

Moving patterns happen with spatiotemporal properties. Focusing pattern analysis on spatial or temporal domains would help information presenting. So, we first discuss about the spatial and temporal operations designed for retrieving related moving patterns in our trajectory data warehouse. Next, queries often request summarizations on one or multiple dimensions over value ranges larger than basic units. And maintenance is worked through further condensing aged moving patterns with larger granularity grids. In response to the need, aggregation of patterns is designed. Afterward, we propose a process to deal with the distinct trajectory estimation problem caused by dividing stage during moving pattern extraction. The distinct estimation is done by recording some additional attributes in pattern extraction processes. In table `flow trace`, records keep track of the duplicately counted trajectory amount between patterns. Finally, we discuss about several types of moving pattern queries, which work with both ordinary and special designed operations our trajectory data warehouse can provide.



3.5.1 Spatial and Temporal Operations

In a query, we usually limit value ranges on some of the attributes to get wanted information. In moving pattern query and analysis, a geographical area or a time period is often a focused target. We are interested in finding patterns related to a regional area or a time period. Thus we provide several pattern query operations on spatial and temporal domains to meet the need.

Here we define the operations in spatial domain, focusing on the laying positions and passing areas of patterns. First, we deal with the operations related to start and end points of moving patterns.

Operation 3.1 (Start from, End in). *A moving pattern PTN starts from an area A if its start point is inside A. Similarly, PTN ends in A if its end point is inside A.* ■

The query target area A is decided by four points $\{(x_s, y_s), (x_e, y_s), (x_s, y_e), (x_e, y_e)\}$, where $x_s < x_e$ and $y_s < y_e$, as area A in Fig. 3.4. And assume PTN has start point (x_a, y_a) and end point (x_b, y_b) . Then, PTN starts from A if

$$x_s \leq x_a < x_e \text{ and } y_s \leq y_a < y_e,$$

and PTN ends in A when

$$x_s \leq x_b < x_e \text{ and } y_s \leq y_b < y_e.$$

With start from and end in operations, we can find patterns initialized and finalized in a region, and the changing and connection among patterns in an area.

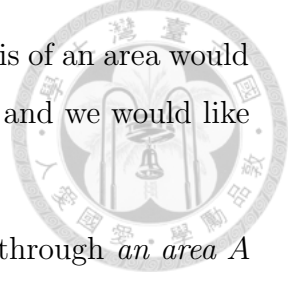
Next, a PTN might all happen inside an area.

Operation 3.2 (Lay on). *A moving pattern PTN lays on an area A if it both starts from and ends in A. Meanwhile, a point P lays on A if it is inside A.* ■

Thus, we can redefine PTN starts from A as its start point lays on A , and PTN ends in A when its end point lays on A . Analyzing patterns lay on a larger area helps emphasizing moving behaviors inside a region, distinguishing them with those cross region boundaries.

Moreover, we are interested in moving patterns related to an area, even their attributes show no direct connection to the area. A moving pattern might neither

start from nor end in an area, but pass through it. Pattern analysis of an area would not be completed without finding out the passing through ones, and we would like to reveal them via an operation.



Operation 3.3 (Pass through). *A moving pattern PTN passes through an area A if the PTN intersects with exactly two boundaries of A .* ■

The pass through definition excluded the *start from* or *end in* cases which intersect only one or less boundary. While the idea of a PTN intersects with exactly two boundaries of A is simple, the computations cost is high in checking intersection with each area boundary in turn. We analyzed the properties and figured out its would be more efficient to deal the computation with diagonal lines of area and their slopes. Thus, we redefine PTN passes through A as it intersects with one of the two diagonal lines in A .

When a pattern is neither *start from* nor *end in* A , we start to check for its intersection with A . First we define the diagonal lines L_+ and L_- of A as

$$L_+ : ((x_s, y_s), (x_e, y_e)), \text{ with positive slope, and}$$

$$L_- : ((x_e, y_s), (x_s, y_e)), \text{ with negative slope.}$$

Next, we calculate the slope of PTN line L_{PTN} as

$$Slope_{PTN} = \Delta y / \Delta x = (y_b - y_a) / (x_b - x_a).$$

When $Slope_{PTN} > 0$, we find for intersecting point of L_{PTN} and L_- , else we search for intersection of L_{PTN} and L_+ . Then, PTN passes through A , if the following check is met with intersecting point (x_c, y_c) .

$$x_s < x_c < x_e \text{ and } y_s < y_c < y_e \text{ and}$$

$$x_a < x_c < x_b \text{ and } y_a < y_c < y_b.$$

The above mentioned spatial operations of moving patterns are all illustrated in Fig. 3.4, the moving pattern is starts from area B , ends in area D , lays on larger area E and passes through area C .

Besides spatial domain, we are also interested in querying for patterns in a time period. In temporal domain, while we store moving patterns period by period, patterns are mapping to time grids they belong with their recorded T_{IDs} .

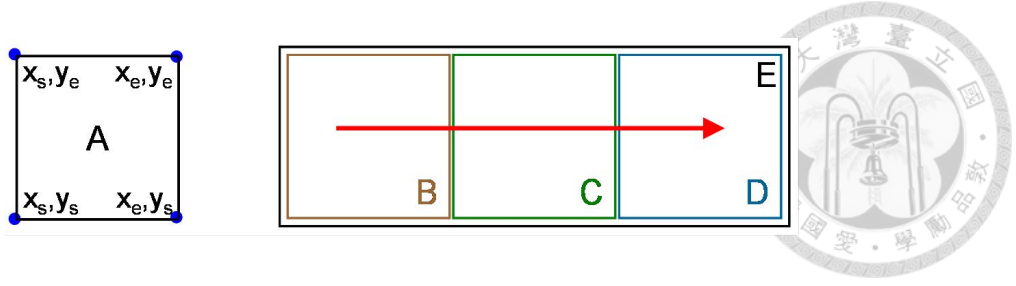


Figure 3.4: Illustration of spatial domain operations.

Operation 3.4 (Happen during). *A moving pattern PTN happens during a time period T , if its corresponding time grid T_{ID} starts and ends in T .* ■

We define target period T start from t_i and end at t_j , and the valid period of PTN queried with T_{ID} as (t_s, t_e) . PTN happens during T when

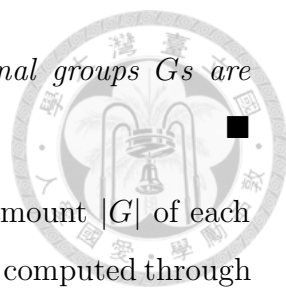
$$t_i \leq t_s < t_e < t_j.$$

The T which patterns happen during may also be a periodical ranges of time, say every 7-9 AM or every Sunday. The time period selecting operation would help pattern queries.

3.5.2 Aggregation and Warehouse Maintenance

The moving patterns storing in the trajectory data warehouse are extracted with regular-sized grids. However, when users query for information, often they may want to know about attributes with ranges larger than the basic granularity in one or more dimensions. The query results are done by *roll-up* operation, which selecting the related patterns from multiple unit grids according to the value requested on each of the dimension. The *slice* and *dice* operations select patterns while giving limit on only one and two or more dimensions, respectively. These operations mainly operate as grouping patterns from multiple grids which meet the required ranges of all dimensions. Since a pattern is extracted based on a unit grid, a gd_u is also the minimum granularity of these operations. When users required not only list of matching patterns but also summarized information of them, aggregation over patterns need to be computed.

Operation 3.5 (Aggregation). *An aggregation of moving patterns finds the new representation line for selected patterns. The start and end positions, time span,*

speed, direction and group amount should be renewed as original groups G s are combined and a new group G' is generated. 

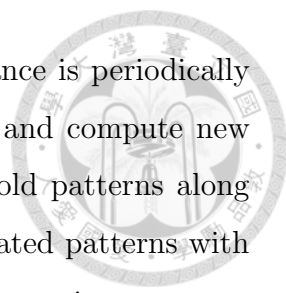
The new pattern is mainly decided by weighting old group amount $|G|$ of each pattern. The new moving patterns $L_{G'}$ generated in aggregation is computed through using each original group amount as weighting parameter to find the new group representations following (3.6) and (3.7). And the new group amount $|G'| = \sum |G|$.

$$\begin{aligned} c_{L_{G'}} &= \frac{\sum_{L_G \in G'} ((p_{L_G,s} + p_{L_G,e})/2) * |G|}{\sum_{L_G \in G'} |G|}, \\ l_{L_{G'}} &= \frac{\sum_{L_G \in G'} d(p_{L_G,s}, p_{L_G,e}) * |G|}{\sum_{L_G \in G'} |G|}, \text{ and} \\ a_{L_{G'}} &= \frac{\sum_{L_G \in G'} ((p_{L_G,e} - p_{L_G,s})/d(p_{L_G,s}, p_{L_G,e})) * |G|}{\sum_{L_G \in G'} |G|}, \end{aligned} \quad (3.6)$$

where $p_{L_G,s}$ and $p_{L_G,e}$ are the two end points of an original pattern L_G , and $|G|$ is the number of lines in a original group G .

$$\begin{aligned} p_{L_{G'}-s} &= c_{L_{G'}} - \frac{l_{L_{G'}} \times a_{L_{G'}}}{2}, \\ p_{L_{G'}-e} &= c_{L_{G'}} + \frac{l_{L_{G'}} \times a_{L_{G'}}}{2}, \text{ and} \\ s_{L_{G'}} &= \frac{\sum_{L_G \in G'} s_G * |G|}{\sum_{L_G \in G'} |G|}. \end{aligned} \quad (3.7)$$

For the trajectory data warehouse to run in a long term, we need a historical pattern maintenance method. The historical maintenance should efficiently condense required storage space, while still keeping main information of aged patterns. First, design of hierarchical granularity levels can help for warehouse maintenance. Several hierarchy of granularity are defined for different aged patterns, usually focusing on spatial area and time range, while sometimes on distinguishing of direction, length or speed. For example, the vehicle moving patterns may first be analyzed in $0.5 * 0.5 km^2$ per hour. After a month, the stored pattern may be aggregated into $1 * 1 km^2$ per three hour grids. The coarser granularity should always be defined as integer times of its finest level, so the aggregation operation can be processed. While applying the spatial and temporal operations defined earlier in this section, the unit of coarser granularity spatial area should always be continuous, and the granularity of time period might become a periodic time set, such as every 9-11 AM of day in a month. The grid granularity used for data warehouse maintenance of different



aged data are recorded in table `grid unit`. Then the maintenance is periodically triggered to aggregate the aged patterns according to new gd_u and compute new summarized representation lines. After the reanalysis is done, old patterns along with the corresponding time period are removed, and new generated patterns with a combined time period of higher level are stored, in both table `moving pattern` and table `time grid`.

When the data warehouse is maintained with aggregation operation, the minimum group size th_{amt} may be increased, as gd_u for generating patterns are enlarged. Thus, both information loss caused by pattern generation would be increased in these renewed aging patterns. This is a trade off between the amount of data to be stored and the conciseness of information to be kept in a long run.

3.5.3 Distinct Trajectory Estimation

Moving patterns kept in the data warehouse are originated from trajectory data sets. For times, we would like to know about the amount of distinct trajectories passing an area. However, different moving patterns happen in an area may include replacement lines originally continuous in one trajectory. Thus, to count distinct trajectories related to an area, additional process is required besides simply check the member amount summation of all patterns in an area. To deal with the distinct trajectory estimation problem without keeping identity list of members in each pattern, we introduced table `flow trace` to record flow information between groups.

As discussed in earlier section, when dividing trajectories, only information of latest checked data points of each trajectory and extracted replacement lines of current period are kept in the temporary tables. Based on the limited information, an additional attribute, previous line ID, is introduced to table `divide check`. This attribute register the replacement line ID as a L_R generated from a trajectory. When the record TR_i is initialized in Algorithm 3.1, $L_{pID} = NULL$ is set. As a L_R is decided to be output, the existing L_{pID} is also transferred into table `replacement line`. Then the ID of just generated L_R is recorded back in L_{pID} of table `divide check` as the attributes in record TR_i are reset.

In table `replacement line`, two additional attributes, belong group ID, G_{ID} , and next-line group ID, G_{nID} , are added and initialized as $NULL$. As a L_R is



Algorithm 3.3 Trajectory flow amount trace

Input: a set of attribute pairs $\{G_{ID}, G_{nID}\}$ from table **replacement line**
Output: a set of flow amount records for table **flow trace**

```

for each records in table replacement line do
  if record  $G_{ID} = NULL$  or  $G_{nID} = NULL$  then
    continue to check next record
  else if record  $\{G_{ID}, G_{nID}\}$  pair does not exist then
    Add record  $\{G_{ID}, G_{nID}\}$  pair
    Initialize flow amount  $|F| = 1$ 
  else
    Update record  $\{G_{ID}, G_{nID}\}$  pair,  $|F| = |F| + 1$ 
  end if
end for

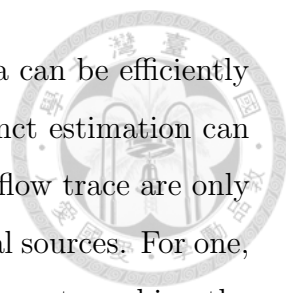
```

included in a group G and the corresponding pattern is generated, the G_{ID} is updated in record L_R . And according to the L_{pID} registered with L_R , the G_{nID} of its previous line is updated as well. After finishing pattern extraction through Algorithm 3.2, the G_{ID} and G_{nID} pairs in table **replacement line** are summarized with Algorithm 3.3. Those pairs without $NULL$ values are processed and the flow amounts of each different pairs are calculated. The distinct G_{ID} and G_{nID} pairs are record into table **flow trace** as from and to group IDs, along with their amounts, $|F|$, for later distinct estimation usage.

Operation 3.6 (Distinct flow amount). *The distinct flow amount AMT in an area A indicates the number of distinct trajectories moving start from, end in, or pass through A . Which can be seen as the total group amount, $|G|$, minus the amount of items that have their next lines also moving in the area, which caused same trajectories being counted among multiple groups.* ■

The AMT in area A can be estimated with the summation of group amount, $|G|$, from table **moving pattern**, minus the summation of flow amount, $|F|$, from table **flow trace**, which both of its to and from group are included in the query results. This can be describe with an equation as follow,

$$AMT = \sum_{G \in A} |G| - \sum_{(G_{from} \in A \text{ and } G_{to} \in A)} |F|. \quad (3.8)$$



Using 3.3 and (3.8), the distinct trajectory amount in an area can be efficiently estimated. The process of parsing additional attributes for distinct estimation can be done in $O(m)$, where m is the amount of L_{RS} . The results in flow trace are only approximate estimation, though. The errors can come from several sources. For one, some replacement lines would not be included into patterns when not reaching the th_{amt} required for a pattern. This might cause underestimate of original trajectory amount, but only to the ratio of information eliminated in pattern generation. In the mean time, an object might start from an area, then move out, and travel back into the area again later. This would cause some overestimate of distinct flow amount. Moreover, when only limited space is available for group flow trace, we might have to keep only the larger amount flow records between groups and eliminate relatively minor ones. This information loss would also cause some underestimate of original trajectory amount.

3.5.4 Moving Pattern Queries

With ordinary condition assignments plus the operations described earlier in this section, moving pattern queries with different complexities can be answered. The basic queries assign some specific properties and get matched moving patterns from the trajectory data warehouse as direct answers. In advanced level, queries include some extra criteria and processes for extracting representations of patterns or their summarized properties. Application level of queries try to give out overall pictures of moving patterns in the trajectory data warehouse, and might further include in other spatial, temporal or related domain information for analytic processes.

A query on specific pattern properties may focus on the value range of one or more attributes, such as finding patterns move in a direction around certain speed. The queries may also work on an area with spatial operations, like finding patterns **start from** a spatial grid or **pass through** an area. Or may apply limits on temporal domain to look for patterns **happen during** some hours of a day. These kinds of property queries work like filters, and are useful in finding out moving patterns matching some assigned conditions.

Query 3.1. *Search for moving patterns **related to** (including **start from**, **end in**, **lay on** and **pass through**) crossroad area A , with moving speed larger than 30*

km/hr, happen during 9-10 AM, 2010-Nov-24.

Next, adding in the **aggregation** and **distinct flow amount** operation, patterns are processed to further reveal information. Such as identify the average moving speeds in different directions of an area, through aggregating patterns in each direction. Or analysis the distribution of length spans in certain direction on a road, to find the effectiveness of traffic light chain setting. Or summarize the **end in amount** of a large area grid by grid, to find out possible locations with special events happening as people gathering. Or process patterns of a spatial grid along each time unit and find out the major directions or the change of average speeds during a day. These queries with summarizing processes can be used to analyze characteristics of attributes under different conditions.

Query 3.2. *For moving patterns related to crossroad area A, which happen during 8-10 AM, 2010-Nov-24, summarize the moving condition in each of their four forward direction units through finding average moving speed and accounting flow amount.* ■

Query 3.3. *Find the behavior changes over a day through calculating the average speeds per hour, in direction north to south, among moving patterns related to crossroad area A, happen during 2010-Nov-24.* ■

Further more, with moving patterns collected in the trajectory data warehouse, general moving conditions can be built as pattern templates for applications. A template of work day morning traffic condition in downtown area can be extracted through finding common patterns from long term collection. Then the pattern template can be used for abnormal behavior detection or regional condition analysis. Or through tracing condition of grids in a long term, the repeated pattern change in speeds, amounts or directions can be revealed and logged as regular trend. Then the trend can be used for event detection or change prediction. These processes can add value to moving patterns stored in the trajectory data warehouse and provide various kinds of useful applications.

Query 3.4. *Find common moving patterns related to crossroad area A, happen during 8-10 AM, work day of Nov, 2010, which is the aggregation results of*

moving patterns occurred on same unit grids during more than 70% of the assigned periods. Then build it as a pattern template of work day 8-10 AM. ■

Query 3.5. *Search for the series trend of speeds among moving patterns related to crossroad area A, through analyzing average speeds in each directions per hour on patterns **happen during** Nov, 2010. Then find out the repeated patterns and save them as regular trends.* ■

3.6 Experiments and Evaluations

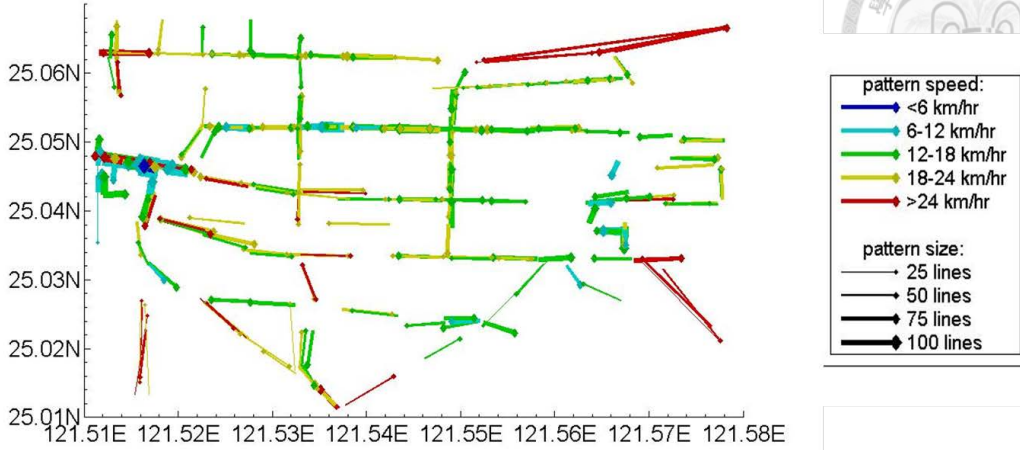
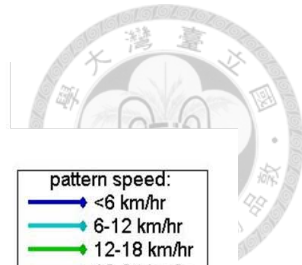
We conducted extensive experiments on two different real trajectory data sets to evaluate the pattern extraction processes and operations we proposed for our trajectory data warehouse. We analyzed the condense of information amount and the time spent of pattern extraction in each steps. Also, the scalability of operations were measured. Along with the analysis, samples of area moving patterns and operation results are also illustrated. All algorithms were implemented in C++ and the experiments were run with 2.9GHz CPU and 4GB RAM.

3.6.1 Trajectory Sets and Pattern Extracted

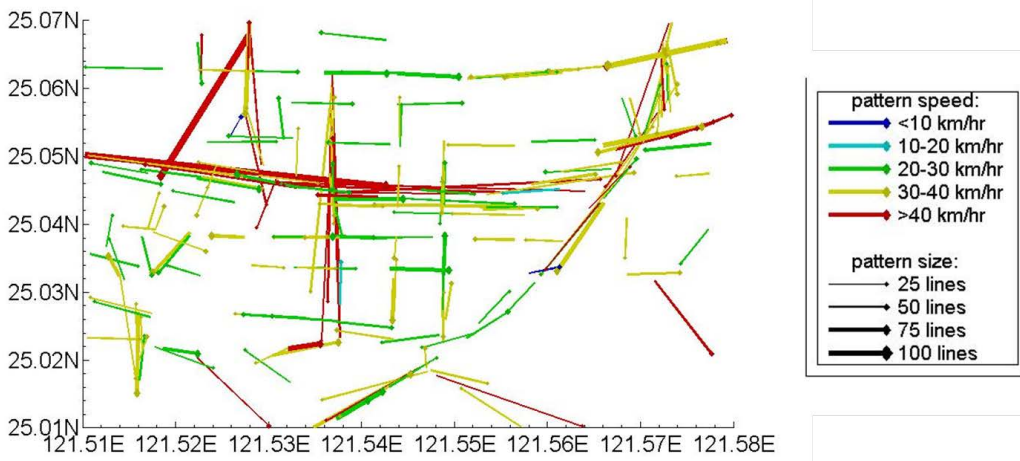
We used trajectory data sets of city buses and taxis for experiments. These two kinds of vehicles both move on city roads but in different moving styles. Thus we used both of them for some comparisons.

The bus trajectories we used are from the Taipei city e-bus system [52], during mid-November, 2012. This system collected GPS positioning data of city buses on duty around every 30 seconds. We used totally 52,892,847 records for pattern extraction. Meanwhile, the taxi trajectories are also GPS position and time data recorded in Taipei city. We used the data from October to November, 2010, with the position sampling in every three seconds. Totally 670,964,334 records were used for experiments.

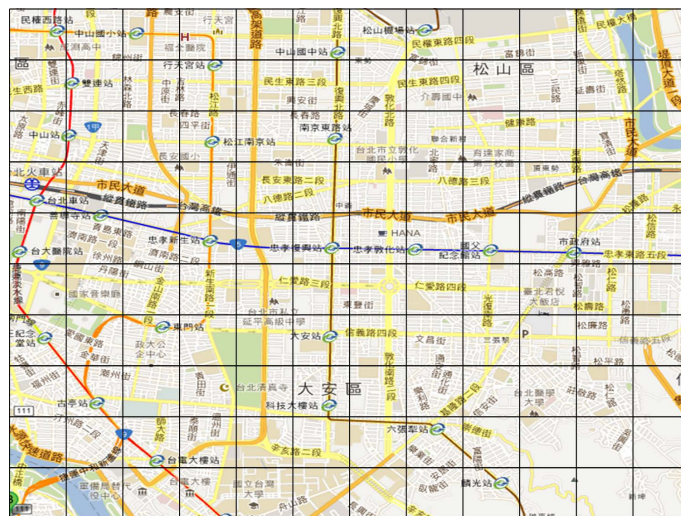
In the experiments, the basic granularity of dimensions in gd_u were set according to data characteristics. The spatial grid was chosen as $0.005^\circ \times 0.005^\circ$ in latitude and longitude, which is similar to the average size of street blocks in Taipei city. The pattern length stretch granularity was set as every 0.5 km, about the distance



(a) Patterns of bus 2012-11-09 17-18 PM



(b) Patterns of taxi 2010-12-02 9-10 AM



(c) Area map: 25.015-25.065N, 121.515-121.575E

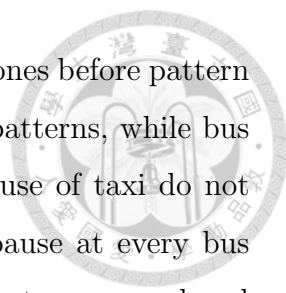
Figure 3.5: Illustration of significant patterns on selected area.

between boundaries of a spatial grid. The speed unit range was 10 km/hr, and the direction granularity was per 90° in 360° angle. The unit grid of time for pattern generating process was per hour.

Based on the experiences learned in vehicle trajectory data profiling, we selected $th_{len} = 0.1$, $th_{spd} = 0.75$ for the dividing criteria. The th_{len} was strict because city roads are mostly straight, and direction changes of vehicles would be rather small besides making turns at crossroads. On the other hand, th_{spd} was set much looser because even in smooth traffic, some speed changes would be often caused by reacting to moving conditions of vehicles nearby. We would like to detect the major speed changes, like stops and moves, but not minor speed adjustments on the road. Meanwhile, we set $th_{amt} = 0.00015$. th_{amt} was set to avoid infrequent patterns to be generated, while the value is also selected as a lower bound to constrain the maximum ratio of information loss.

We show examples of extracted patterns in Fig. 3.5, and get a closer look. A busy area, 25.015-25.065N, 121.515-121.575E, in Taipei city was chosen. The corresponding road map is as Fig. 3.5(c). With the parameters assigned above, 204,570 bus trajectory records in the selected area were divided into 68,437 replacement lines, the L_{RS} separated in 3,959 unit grids. Then 1,763 moving patterns were generated, with group size larger than 10. In Fig. 3.5(a), we illustrated 200 significant patterns. Meanwhile, 314,608 taxi records in the same area were replaced by 36,862 L_{RS} . Candidate L_{GS} separated in 4,571 unit grids, and with minimum group size assigned as 6, 1,435 moving patterns were generated later. In Fig. 3.5(b), 200 major patterns are shown.

From the sample results in Fig. 3.5, we could see the pattern extraction based on dividing criteria and predefined gd_u s can efficiently summarize the moving behaviors. The patterns clearly illustrated various moving speeds, span lengths, flow amounts on different sections and directions of roads. In the meantime, spread of patterns well aligned the distribution of city roads automatically. The results also showed some speedy movings with long length spans, and many other slower ones stopped their length extensions around some crossroads. Furthermore, the sketches revealed distinguishable differences among the two data sets we used. We can see the patterns of taxi movings are more flexible than the bus behaviors. Meanwhile, the generated



replacement lines of taxi data spread among more grids than bus ones before pattern generation. Also, taxi movings included more long and speedy patterns, while bus movings were mostly in quite regular lengths. These were because of taxi do not have previously assigned routes as buses, and do not have to pause at every bus stops on the roads. Furthermore, since the two trajectory data sets were analyzed with same parameters, the generated patterns can be directly compared with each other based on gd_u s, and can used for more detailed analysis.

3.6.2 Space and Time Analysis

The information amount storing in tables shrank obviously after trajectory dividing and pattern generating. We analyzed the amount ratio relations among original records $|pt|$, divided segments $|L_R|$, and generated patterns $|L_G|$ of both data sets with different parameters in the two stages of pattern extraction, and summarized in Table 3.1. Testing series of different th_{spd} from 0.25 to 2.5, we can see the data compression ratio, $|L_R|/|pt|$ is larger with looser dividing criterion, which indicating fewer L_{RS} are generated as more incoming data points pass the dividing check with higher th_{spd} setting. With more $|L_R|$ generated among the fixed number of predefined unit grids, better information condensing ratio of $|L_G|/|L_R|$ are reached in pattern generation. While the final compression ratios $|L_G|/|pt|$ do not vary that much after all processes.

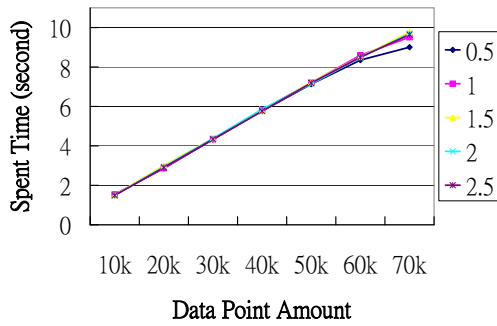
Further information condensing in group generation is decided by the size of unit grids. $|L_G|/|pt|$ are much smaller with latitude and longitude granularity as 0.01° than as 0.005° ones. With larger gd_u , fewer grids would exist in an area. As a result, with the same L_{RS} spreading, fewer patterns would be generated. Moreover, the initial sampling rates of position and time records also influence obviously on the amount of L_{RS} generated during trajectory dividing. Taxi data, which was sampled more frequently, get better condensing ratios. Changes between more frequent sampling records tend to be smaller and are more likely to pass the dividing check, thus fewer divisions were divided. Also, the data moving styles cause differences in compression ratios and information losses. More moving patterns were generated from divided lines of taxi data, according to statistic in earlier section. The more flexible moving of taxi causes flatter distribution among different grids.

Table 3.1: Summarizing information amount in pattern extraction processes

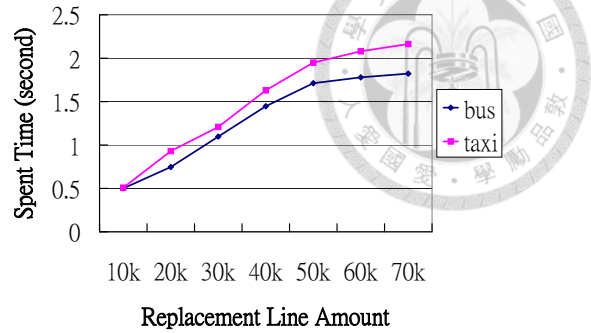
Data set	Divide ($th_{len} = 0.1$)		Group (lat., lon.: 0.005°)			Group (lat., lon.: 0.01°)		
	th_{spd}	$\frac{ L_R }{ pt }$	$\frac{ L_G }{ L_R }$	$\frac{L_R \in L_G}{ L_G }$	$\frac{ L_G }{ pt }$	$\frac{ L_G }{ L_R }$	$\frac{L_R \in L_G}{ L_G }$	$\frac{ L_G }{ pt }$
bus	0.25	45.97%	8.18%	90.20%	1.99%	3.70%	96.73%	1.07%
	0.5	42.67%	8.87%	89.75%	1.96%	4.06%	96.46%	1.07%
	0.75	39.44%	9.64%	89.28%	1.93%	4.47%	96.22%	1.08%
	1.0	35.96%	10.57%	88.71%	1.90%	4.97%	95.96%	1.09%
	1.25	33.02%	11.57%	88.06%	1.88%	5.52%	95.74%	1.12%
	1.5	29.19%	13.13%	87.33%	1.87%	6.37%	95.56%	1.16%
	2.0	21.74%	16.84%	86.17%	1.75%	8.72%	95.13%	1.22%
	2.5	16.80%	19.51%	86.75%	1.60%	10.95%	94.92%	1.20%
taxi	0.25	22.74%	14.79%	84.05%	1.60%	6.33%	96.01%	0.96%
	0.5	19.15%	16.91%	84.01%	1.51%	7.60%	95.36%	0.93%
	0.75	16.37%	18.65%	84.22%	1.39%	8.88%	94.34%	0.86%
	1.0	14.22%	20.13%	84.24%	1.26%	10.09%	93.65%	0.80%
	1.25	12.48%	22.21%	83.39%	1.19%	11.60%	92.94%	0.79%
	1.5	11.05%	23.94%	83.03%	1.14%	12.86%	92.61%	0.77%
	2.0	9.00%	26.90%	82.55%	1.05%	14.99%	92.30%	0.75%
	2.5	7.55%	30.20%	92.30%	1.00%	17.15%	91.74%	0.72%

Meanwhile, with more member amounts lesser than th_{amt} in grids, a little lower ratio of L_R s are included in the L_G s for taxi data.

We analyze about the total information amount condensed in pattern extraction process. However, during trajectory dividing process, the temporary table `divide check` in fact only keeps a record of latest update for each trajectories, rather than all of its raw data points. And the divided L_R s in another temporary table `replacement line` only keeps records in current time period, which are then processed by pattern generating stage and directly dropped afterward. The storage spaces for temporary tables can be reused after patterns of a time period are generated. That is to say, although we have to deal with huge amount of data, the storage usage is quite efficient during processing, and the process condensed information amount to less than two percent of original in the experiments as shown in Table 3.1. In the meantime, the compression analysis of different spatial grids provide outlook of



(a) Trajectory dividing.



(b) Pattern generating.

Figure 3.6: Time spent in extracting moving patterns.

historical maintenance. With times four sized spatial grids, the patterns are lesser to around half amount, which are still reflect the pattern distributing condition over new grids.

The algorithms we used for trajectory dividing check and pattern generating are quite efficient and with good scalability. The time spent with various amount of data is shown in Fig. 3.6. We tested up to 70000 position records and replacement lines in corresponding processes. The results shows the algorithms cost only linear time and are quite scalable. Still have to remind that the trajectory dividing algorithm is designed to process the data on-the-fly, not really in batch mode.

3.6.3 Operation Analysis

The pattern operations introduced in Section 3.5 were implemented. Sample results of operations on an area including a crossroad, in 25.05-25.055N and 121.53-121.535E, of bus data during 2010-03-09 17-18 PM are shown in Fig. 3.7. The result of `start from` included 29 patterns in Fig. 3.7(a), `end in` as Fig. 3.7(b) included 31, `lay on` in Fig. 3.7(c) are for `lay on`, and in Fig. 3.7(d) 8 for `pass through` operations. Additionally, the four new groups of directional aggregation on `start from` results in all direction units are illustrated in Fig. 3.7(e). The figure showed the pattern distributing in each direction of crossroad. While the forward speeds and length spans various with larger range in `start from` and `end in` patterns, they are rather slow in `lay on` and medium fast in `pass through` ones. As to directional aggregation which summarized all the other attributes, we can see both the flow amounts

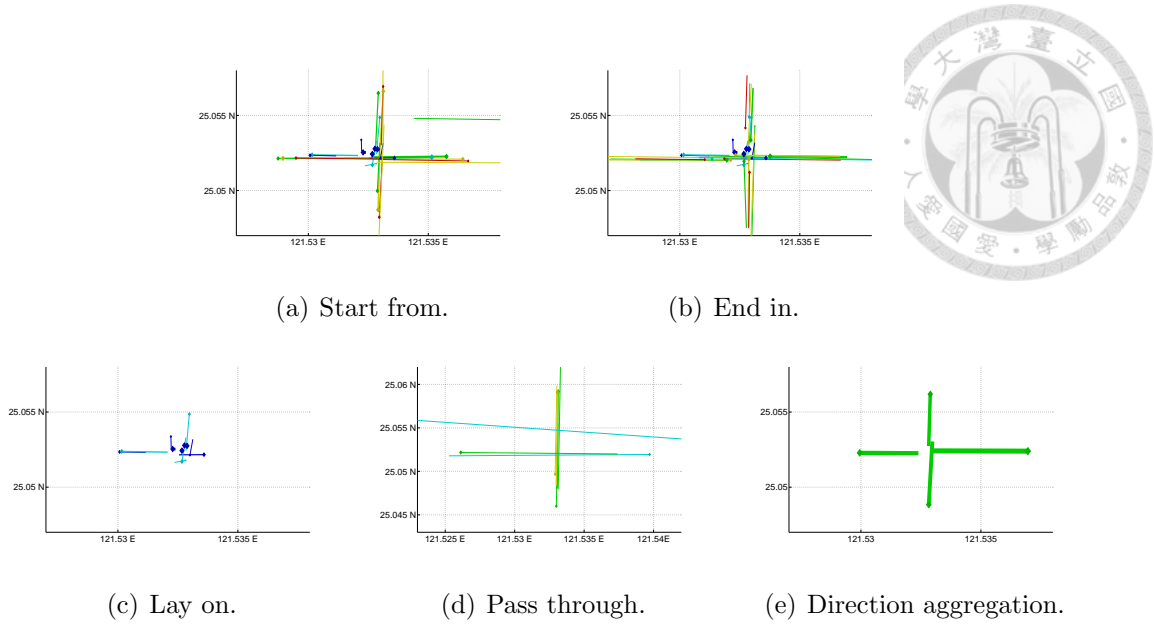


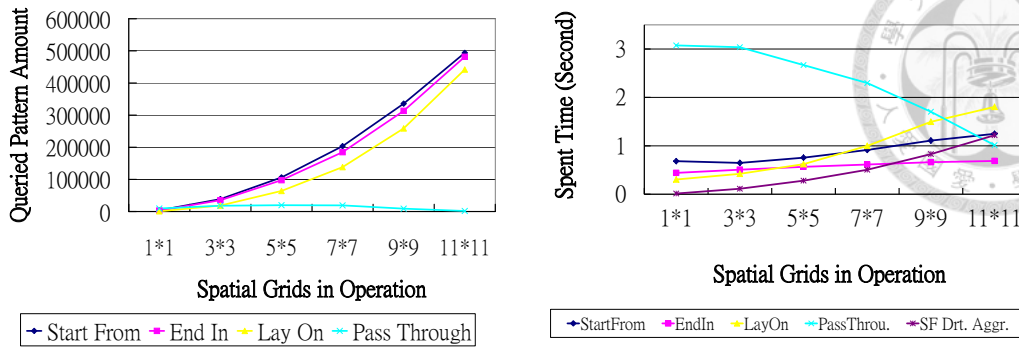
Figure 3.7: Results of pattern related operation on selected grid.

and averaged speeds and lengths are quite even on this crossroad.

Some tests on efficiency and scalability of the spatial and aggregation operations were made. Operation tests were done on different sized spatial areas, which included $1*1$, $3*3$, $5*5$, $7*7$, $9*9$ and $11*11$ unit grids, with 70,000 collected patterns distributed in a $15*15$ unit grids area. The results of queried amount and spent time of operations are shown in Fig. 3.8. The amounts related to the start and end positions increased rapidly as the area getting larger. More times were taken as more patterns would pass the first condition test and goes further, though still increased much slower than the result amounts. As to the time spent on aggregation, it was proportional to the amount of queried start from patterns. In the meantime, the criteria for pass through operation got harder to meet as area size increasing, for longer patterns were required. And as checks would often fail in first one or two conditions, the cost time decreased obviously. Generally speaking, time cost on these operations increased much slower than the problem size. Thus, these warehouse pattern operations are quite efficient and scalable.

3.7 Summary

In this work, we proposed a trajectory data warehouse from its processing, table schema design to moving pattern operations. Moving patterns were extracted from



(a) Result pattern amount.

(b) Operation spent time.

Figure 3.8: Analysis of pattern related operation.

trajectories with a two-stage algorithm and loaded into the data warehouse. First, behaviors of trajectories were analyzed, divided into spatiotemporal consistency segments using an online algorithm and summarized as replacement lines. Then, similar replacement lines were grouped and the representation lines were transformed into regular formatted moving patterns with a multidimensional-grid based algorithm. Finally, patterns were loaded into the data warehouse along with their time and grid unit information.

Different types of pattern queries and data warehouse maintenance could be worked with the spatial and temporal operations and the aggregation of moving patterns based on different granularity in attribute dimensions. We also introduce a process for estimating distinct trajectory among patterns relating to an area. Moreover, we conducted extensive experiments on real trajectory data sets. The results showed efficiency of moving patterns extraction and warehouse operations. The data warehouse not only summarized raw trajectories and transformed information into regular format, but also provided a structure that allows flexible application possibilities.



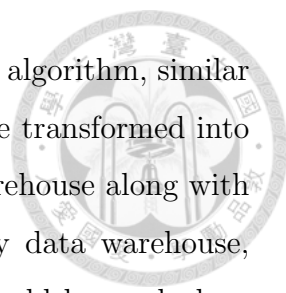
Chapter 4

Conclusion and Future Work

4.1 Conclusion

This dissertation mainly dealt with analyzing object moving behaviors, which were extracted from huge amount of trajectory data. First, we presented DivCluST, an approach to profiling a set of moving objects by dividing and clustering their accumulated trajectories spatiotemporally. We used DivST to divide the trajectories and to generate corresponding replacement lines. Through the design of spatial and temporal dividing criteria accompanied with the threshold parameter selections, a proper number of replacement lines were produced while preserving spatiotemporal properties of their original trajectories well. Then, CluST clustered the replacement lines based on the proposed spatiotemporal line distance function in consideration of the position, length, direction and speed differences. With the specially designed mean line representation, the clustering results revealed regional main spatiotemporal moving behaviors of any given trajectory data set.

Through profiling, we knew better on characteristics of moving behaviors hidden among the trajectories. Since we wanted to use the moving behaviors in further applications, we turned to work on a trajectory moving pattern data warehouse. We proposed a trajectory data warehouse, from its data processing, table schema to pattern operations, focusing on object moving behaviors. Moving patterns were extracted from raw trajectories and then loaded into the trajectory data warehouse. We first analyzed behaviors of trajectories and divided them into spatiotemporal consistency segments with an online algorithm. The divided sub-trajectories were



then replaced by lines. Next, with a multidimensional-grid based algorithm, similar replacement lines were grouped and the representation lines were transformed into moving patterns. Finally, patterns were loaded into the data warehouse along with their time period and grid unit information. In the trajectory data warehouse, different types of pattern queries and warehouse maintenance could be worked on multiple levels of grid granularity, with the aggregation, spatial and temporal operations and distinct flow estimation over moving patterns. The data warehouse not only summarized the raw trajectories and transformed information into a regular format, but also had a structure that allows flexible application possibilities.

We conducted extensive experiments on real trajectory data sets for both trajectory data profiling and warehousing designs. The results illustrated from profiling and warehousing showed ability of these methods. They can successfully reveal the moving object behaviors among different regions and times. The analysis also showed the efficiency in DivST and CluST of trajectory profiling, and moving pattern extraction and operations of trajectory warehousing. With these moving behavior analyzed, further applications related to object trajectories can be provided.


4.2 Future Work

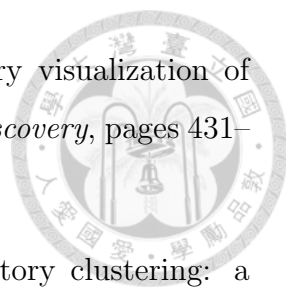
For future research, we may use a profile as the signature of corresponding type of moving objects for other mining tasks. With profiles of different trajectory data sets, various typical moving behaviors can be used to identify the object type. And we may use the information collected in trajectory data warehouse to build typical sketches of object moving for other pattern based data mining tasks. The template patterns can be used for event analysis and real-time abnormal condition detection. And the routinely series trends can be used in moving style change analysis and future moving behavior predictions. Moreover, we may link the moving patterns with many other spatial, temporal, or event databases, and extend the applications of data warehouse. For instance, we can link the patterns to real world road status or weather conditions, and find the related influences in moving behaviors. Analysis may also be further focused on area with interesting behaviors revealed by the moving patterns, and provide location based services or other applications.




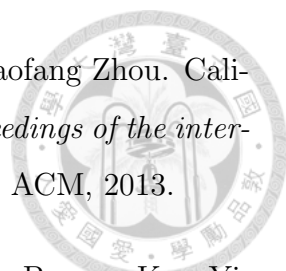
Bibliography


- [1] Wang-Chien Lee and John Krumm. Trajectory preprocessing. In *Computing with Spatial Trajectories*, pages 3–33. Springer, 2011.
- [2] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, pages 99–108. ACM, 2010.
- [3] Ke Deng, Kexin Xie, Kevin Zheng, and Xiaofang Zhou. Trajectory indexing and retrieval. In *Computing with Spatial Trajectories*, pages 35–60. Springer, 2011.
- [4] Jae-Woo Chang, Jung-Ho Um, and Wang-Chien LeeP. A new trajectory indexing scheme for moving objects on road networks. In *Flexible and Efficient Information Handling*, pages 291–294. Springer, 2006.
- [5] Dimitrios Gunopulos and Goce Trajcevski. Similarity in (spatial, temporal and) spatio-temporal datasets. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 554–557. ACM, 2012.
- [6] Hechen Liu, Ling-Yin Wei, Yu Zheng, Markus Schneider, and Wen-Chih Peng. Route discovery from mining uncertain trajectories. In *Data Mining Workshops, IEEE 11th International Conference on*, pages 1239–1242, 2011.
- [7] Goce Trajcevski. Uncertainty in spatial trajectories. In *Computing with Spatial Trajectories*, pages 63–107. Springer, 2011.
- [8] Gianni Giannotti, Fosca Giannotti, and Dino Pedreschi. *Mobility, data mining and privacy: Geographic knowledge discovery*. Springer, 2008.


- 
- [9] Manolis Terrovitis and Nikos Mamoulis. Privacy preservation in the publication of trajectories. In *Mobile Data Management, the 9th International Conference on*, pages 65–72. IEEE, 2008.
- [10] Gyözö Gidófalvi, Xuegang Huang, and Torben Bach Pedersen. Privacy-preserving data mining on moving object trajectories. In *Mobile Data Management, International Conference on*, pages 60–68. IEEE, 2007.
- [11] Nam Thanh Nguyen, Dinh Q Phung, Svetha Venkatesh, and Hung Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov model. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 2, pages 955–960, 2005.
- [12] John Krumm. Trajectory analysis for driving. In *Computing with Spatial Trajectories*, pages 213–241. Springer, 2011.
- [13] Chan-Hyun Kang, Jung-Rae Hwang, and Ki-Joune Li. Trajectory analysis for soccer players. In *Data Mining Workshops, 6th IEEE International Conference on*, pages 377–381, 2006.
- [14] Hassan A Karimi and Xiong Liu. A predictive location model for location-based services. In *Proceedings of the 11th ACM international symposium on Advances in geographic information systems*, pages 126–133, 2003.
- [15] Yukun Chen, Kai Jiang, Yu Zheng, Chunping Li, and Nenghai Yu. Trajectory simplification method for location-based social networking services. In *Proceedings of International Workshop on Location Based Social Networks*, pages 33–40. ACM, 2009.
- [16] Harvey J Miller and Jiawei Han. *Geographic data mining and knowledge discovery*. CRC Press, 2009.
- [17] Gennady Andrienko, Natalia Andrienko, and Stefan Wrobel. Visual analytics tools for analysis of movement data. *ACM SIGKDD Explorations Newsletter*, 9(2):38–46, 2007.


- 
- [18] Menno-Jan Kraak and Otto Huisman. Beyond exploratory visualization of space–time paths. *Geographic data mining and knowledge discovery*, pages 431–443, 2009.
- [19] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 593–604, 2007.
- [20] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 330–339, 2007.
- [21] Luca Leonardi, Salvatore Orlando, Alessandra Raffaetà, Alessandro Roncato, and Claudio Silvestri. Frequent spatio-temporal patterns in trajectory data warehouses. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1433–1440, 2009.
- [22] Marcin Gorawski and Pawel Jureczek. Regions of interest in trajectory data warehouse. In *Intelligent Information and Database Systems*, pages 74–81. Springer, 2010.
- [23] Gerasimos Marketos, Elias Frenzos, Irene Ntoutsis, Nikos Pelekis, Alessandra Raffaetà, and Yannis Theodoridis. Building real-world trajectory warehouses. In *Proceedings of the seventh ACM international workshop on data engineering for wireless and mobile access*, pages 8–15, 2008.
- [24] F Braz, Salvatore Orlando, Renzo Orsini, A Raffaella, Alessandro Roncato, and Claudio Silvestri. Approximate aggregations in trajectory data warehouses. In *Data Engineering Workshop, IEEE 23rd International Conference on*, pages 536–545, 2007.
- [25] Stefano Spaccapietra, Christine Parent, Maria Luisa Damiani, Jose Antonio de Macedo, Fabio Porto, and Christelle Vangenot. A conceptual view on trajectories. *Data & knowledge engineering*, 65(1):126–146, 2008.
- [26] Zhixian Yan, Jose Macedo, Christine Parent, and Stefano Spaccapietra. Trajectory ontologies and queries. *Transactions in GIS*, 12(s1):75–91, 2008.

- 
- [27] Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otavio Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the ACM symposium on Applied computing*, pages 863–868, 2008.
- [28] Jiawei Han, Jae-Gil Lee, and Micheline Kamber. An overview of clustering methods in geographic data analysis. In *Geographic Data Mining and Knowledge Discovery*. CRC Press, 2009.
- [29] Scott J Gaffney, Andrew W Robertson, Padhraic Smyth, Suzana J Camargo, and Michael Ghil. Probabilistic clustering of extratropical cyclones using regression mixture models. *Climate dynamics*, 29(4):423–440, 2007.
- [30] Fatih Porikli. Trajectory distance metric using hidden markov model based representation. In *IEEE European Conference on Computer Vision, PETS Workshop*, volume 3, 2004.
- [31] Jinfeng Ni and Chinya V Ravishankar. Indexing spatio-temporal trajectories with efficient polynomial approximations. *Knowledge and Data Engineering, IEEE Transactions on*, 19(5):663–678, 2007.
- [32] Zhouyu Fu, Weiming Hu, and Tieniu Tan. Similarity based vehicle trajectory clustering and anomaly detection. In *Image Processing, IEEE International Conference on*, volume 2, pages II–602, 2005.
- [33] Elias Frenzos, Kostas Gratsias, and Yannis Theodoridis. Index-based most similar trajectory search. In *Data Engineering, IEEE 23rd International Conference on*, pages 816–825, 2007.
- [34] Aris Anagnostopoulos, Michail Vlachos, Marios Hadjieleftheriou, Eamonn Keogh, and Philip S Yu. Global distance-based segmentation of trajectories. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 34–43, 2006.
- [35] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, pages 791–800. ACM, 2009.

- 
- [36] Han Su, Kai Zheng, Haozhou Wang, Jiamin Huang, and Xiaofang Zhou. Calibrating trajectory data for similarity-based analysis. In *Proceedings of the international conference on Management of data*, pages 833–844. ACM, 2013.
- [37] Gook-Pil Roh, Jong-Won Roh, Seung-Won Hwang, and Byoung-Kee Yi. Supporting pattern-matching queries over trajectories on road networks. *Knowledge and Data Engineering, IEEE Transactions on*, 23(11):1753–1758, 2011.
- [38] J-G Lee, Jiawei Han, Xiaolei Li, and Hong Cheng. Mining discriminative patterns for classifying trajectories on road networks. *Knowledge and Data Engineering, IEEE Transactions on*, 23(5):713–726, 2011.
- [39] Zhouyu Fu, Weiming Hu, and Tieniu Tan. Similarity based vehicle trajectory clustering and anomaly detection. In *Image Processing, 2005. IEEE International Conference on*, volume 2, pages II–602, 2005.
- [40] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094, 2008.
- [41] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 312–321. ACM, 2008.
- [42] Kevin Buchin, Maike Buchin, Marc Van Kreveld, and Jun Luo. Finding long and similar parts of trajectories. *Computational Geometry*, 44(9):465–476, 2011.
- [43] Maike Buchin, Anne Driemel, Marc van Kreveld, and Vera Sacristán. An algorithmic framework for segmenting trajectories based on spatio-temporal criteria. In *Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 202–211, 2010.
- [44] Peter Grünwald. Advances in minimum description length: Theory and applications. chapter Introducing the Minimum Description Length Principle. MIT press, 2005.

- 
- [45] Jingying Chen, Maylor K Leung, and Yongsheng Gao. Noisy logo recognition using line segment hausdorff distance. *Pattern recognition*, 36(4):943–955, 2003.
- [46] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.
- [47] Claudio Bettini, Sushil Jajodia, and Sean Wang. *Time granularities in databases, data mining, and temporal reasoning*, chapter Granularity Systems. Springer, 2000.
- [48] Ed Williams. Aviation formulary v 1.46, 2011. <http://williams.best.vwh.net/avform.htm>.
- [49] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.
- [50] Oded Z Maimon and Lior Rokach. *Data mining and knowledge discovery handbook*, chapter Clustering Method. Springer, 2005.
- [51] Joint typhoon warning center. <http://www.usno.navy.mil/JTWC>.
- [52] Taipei e-bus system. <http://www.e-bus.taipei.gov.tw/>.
- [53] Kerry Emanuel. *Divine Wind-The History and Science of Hurricanes*. Oxford University Press, 2005.
- [54] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. 96:226–231, 1996.
- [55] Huey ru Wu, Mi-Yen Yeh, and Ming-Syan Chen. Profiling moving objects by dividing and clustering trajectories spatiotemporally. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints), 2012.
- [56] Luca Leonardi, Gerasimos Marketos, Elias Frenzos, Nikos Giatrakos, Salvatore Orlando, Nikos Pelekis, Alessandra Raffaetà, Alessandro Roncato, Claudio Silvestri, and Yannis Theodoridis. T-warehouse: Visual olap analysis on trajectory data. pages 1141–1144, 2010.

- 
- [57] Salvatore Orlando, Renzo Orsini, Alessandra Raffaetà, Alessandro Roncato, and Claudio Silvestri. Spatio-temporal aggregations in trajectory data warehouses. In *Data Warehousing and Knowledge Discovery*, pages 66–77. Springer, 2007.
- [58] Elio Masciari. Warehousing and querying trajectory data streams with error estimation. In *Proceedings of the 15th international workshop on Data warehousing and OLAP*, pages 113–120. ACM, 2012.
- [59] Gerasimos Marketos and Yannis Theodoridis. Ad-hoc olap on trajectory data. In *Mobile Data Management, 11th International Conference on*, pages 189–198, 2010.
- [60] Ouri Wolfson. Moving objects information management: The database challenge. In *Next Generation Information Technologies and Systems*, pages 75–89. Springer, 2002.
- [61] Rouaa Wannous, Jamal Malki, Alain Bouju, and Cécile Vincent. Modelling mobile object activities based on trajectory ontology rules considering spatial relationship rules. In *Modeling Approaches and Algorithms for Advanced Computer Applications*, pages 249–258. Springer, 2013.
- [62] Phan Nhat Hai, Pascal Poncelet, and Maguelonne Teisseire. Get_move: an efficient and unifying spatio-temporal pattern mining algorithm for moving objects. In *Advances in Intelligent Data Analysis XI*, pages 276–288. Springer, 2012.
- [63] Oliver Baltzer, Frank Dehne, Susanne Hambrusch, and Andrew Rau-Chaplin. Olap for trajectories. In *Database and Expert Systems Applications*, pages 340–347. 2008.
- [64] Elio Masciari. Warehousing and querying trajectory data streams with error estimation. In *Proceedings of the fifteenth international workshop on Data warehousing and OLAP*, pages 113–120. ACM, 2012.

- 
- [65] Hongjun Zhu, Jianwen Su, and Oscar H Ibarra. Trajectory queries and octagons in moving object databases. In *Proceedings of the 11th international conference on Information and knowledge management*, pages 413–421. ACM, 2002.
- [66] Bart Kuijpers and Walied Othman. Trajectory databases: Data models, uncertainty and complete query languages. In *Database Theory–ICDT 2007*, pages 224–238. Springer, 2006.
- [67] Xiaofeng Meng and Zhiming Ding. Dsttmod: A future trajectory based moving objects database. In *Database and Expert Systems Applications*, pages 444–453. 2003.