OLAP FOR TRAJECTORIES

by
Hou Xiang Wang

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

MASTER OF COMPUTER SCIENCE

School of Computer Science

at

CARLETON UNIVERSITY

Ottawa, Ontario
January, 2010

Your file Votre référence
ISBN: 978-0-494-68622-5
Our file Notre référence
ISBN: 978-0-494-68622-5

Table of Contents

List of Tables

List of Figures

Abstract

The rapid development of mobile computing and technologies such as RFID and GPS, has led to the generation of massive temporal and spatial data; demand to process this data has increased. Due to this trend, a growing number of researchers are interested in analyzing the trajectories of moving objects. Many methods have been proposed to solve this problem. In particular, Baltzer et al [1] proposed a new Group-By operator GROUP_TRAJECTORIES for analyzing trajectories, which is implemented by three computing group methods: Group By Overlap, Group By Intersection, and Group By Overlap and Intersection. The purpose of this research is to expand upon Baltzer et al's methods by improving the results of computing groups of trajectories, trying to optimize parameters and simplifying the usage of the Group-By operator. The Shifting Grid Algorithm and Auto Parameters Algorithm are proposed for improving computing group results, and applied for both Group By Intersection and Group By Overlap. The Group By Intersection is intended for parallel movement of moving objects. Group By Overlap is desirable for the analysis of sequences of movements. The Shifting Grid Algorithm and Auto Parameters Algorithm can improve the resulting groups of trajectory computation for both cases in data sets with and without noise. The Auto Parameters Algorithm is proposed for improving the result combined with the Shifting Grid Algorithm, automatically calculated groups of trajectories, and determined a better result according to the definition of better result. Both the Shifting Algorithm and Auto Parameters Algorithm are further validated by a GUI simulation tool.

Acknowledgements

I am heartily appreciative of my supervisor, Frank Dehne, who has provided invaluable guidance and support during my research and study at Carleton University.

Chapter 1

Introduction

1.1 Introduction

Spatial and temporal data are easily captured via new mobile technology tools such as RFID and Global Position System (GPS). This data is often used in a variety of applications, including land management, environmental, ecological, and biodiversity studies, tracking of mobile devices, navigation systems and merchandize shipping. For example, it is estimated that on a daily basis, NASA typically captures several gigabytes of data, and Wal-Mart Corporation generates upwards of one terabyte of data due to the use of RFID. Analysis of individual data entries in databases are rarely feasible due to the sheer size of the data sets, and, in some cases, not possible due to legal restrictions (i.e. keeping track of the trajectory followed by a cell phone user). There is a need for extracting general characterizations of large subsets of the data from the ever growing large data sets. It is useful to develop techniques that can efficiently summarize and discover trends in data and help in business intelligence. There are two types of moving objects: moving objects on the earth such as cars and humans, and moving objects in space such as aircrafts and satellites. For moving objects in space, the trajectory needs three dimensions to represent its locations, for example $[x_1, y_1, z_1, t_1]$, $[x_2, y_2, z_2, t_2]$, $\cdots$, $[x_i, y_i, z_i, t_i]$, $[x_n, y_n, z_n, t_n]$, where x_i, y_i and z_i determine the location of moving objects in space, and t_i determines the time. The previous studies mainly focus on moving objects on the earth, but the study of moving objects in space is a good future research topic. However, this research only focuses on moving objects on the earth. Conceptually, all moving objects on the earth can be tracked by their locations and time, and they can be represented as $[x_1, y_1, t_1]$, $[x_2, y_2, t_2]$, $\cdots$, $[x_i, y_i, t_i]$, $[x_n, y_n, t_n]$, where x_i and y_i determine the location, and t_i determines the time. Each moving object makes a series of movements, and can be represented by $[id, x_1, y_1, t_1]$, $[id, x_2, y_2, t_2]$, $\cdots$, $[id, x_n, y_n, t_n]$, called a **trajectory**.

Due to the large amounts of data and other concerns such as privacy, raw trajectory data cannot be stored and maintained for extended periods of time. However, people may be interested in the knowledge hidden in the data since the data can be widely used in different applications [23], [20], [7], and [16].

One of the most challenging issues of trajectory data warehousing (TDW) (see Section 2.2) is the manipulation of spatial-temporal data since such data have no direct relation between two moving objects. Trajectories are more meaningful and useful if they are aggregated as groups. Many papers present the aggregation of trajectories such as [14] and [18]. The result of trajectory aggregation could be diverse based on different approaches of aggregation. The aggregation of trajectories is a useful tool for trajectory data warehousing. Therefore, it is important to have a well-established set of tools for online analytical processing (OLAP) analysis in order to analyze large scale data sets representing moving objects. It is necessary to aggregate with respect to trajectory as a feature dimension as well as a measure dimension in order to apply OLAP tools towards moving object datasets. The analysis tool can be applied in a trajectory data warehouse. The purpose of this research is to expand Baltzer et al's method by improving computing groups of trajectories and simplifying on this tool.

Trajectories that cannot be aggregated into any groups are considered to be **noise**, and noise should be eliminated by a pruning process. The OLAP tool allows for extraction of useful data from a raw trajectory database. For example, in Figure 6.29 on page 57, the red pink trajectories are useful trajectories, and other black trajectories are considered to be noise data.

The rest of this chapter is organized as follows: Section 1.2 presents the motivation of this research; Section 1.3 describes the statement of the problem; Section 1.4 sums up the contributions and results of this research; and Section 1.5 presents the organization of this thesis.

1.2 Motivation

Temporal and spatial data can provide very useful information in trajectory data warehousing. Particularly, TDW can provide answers to the following questions:

1. How busy is a particular street? How many cars are there on the street at a particular point of time? At what time is the street busy? The number of cars can be counted based on the aggregation of the trajectory data. For example, in a traffic control application, rather than studying the precise position of every single vehicle on a particular road, it may be of interest to know the overall number of cars crossing an intersection during rush hour. This information may be useful to support decision making in areas such as construction of new roads and underpasses or the addition of new traffic lights.

2. How many school buses are there in the community? Do we need more school buses in this area? What quality of service should be provided? The trajectory data can show how many school buses are in a particular area; based on this, one can decide whether this area needs more school buses.

3. How does an infection spread? How can we prevent people from being infected by a particular disease? For example, in a SARS situation, spatial data can be used to capture the movement of the virus. Actions can be taken to quarantine, limiting the spread of the virus.

4. How do the natural elements in a nature reserve distribute in a particular region? For biodiversity studies, it might be of interest to determine the biodiversity distribution of Ontario in a particular region, rather than the specific position of each plant or animal. Once a region rich in biodiversity has been detected it can be considered as a declaration of a natural reserve.

These queries can be answered in OLAP for trajectories. The OLAP for trajectories leads to a new research problem in TDW.

1.3 Statement of the Problem

The problem of OLAP for trajectories can be demonstrated by Figure 1.1. The numbers of trajectories in Figure 1.1 (a) on page 4 can be considered as input data of this research problem. The similar trajectories are captured by the Frequent Itemsets Mining Program in Figure 1.1 (b), e.g. Apriori. Each group contains a

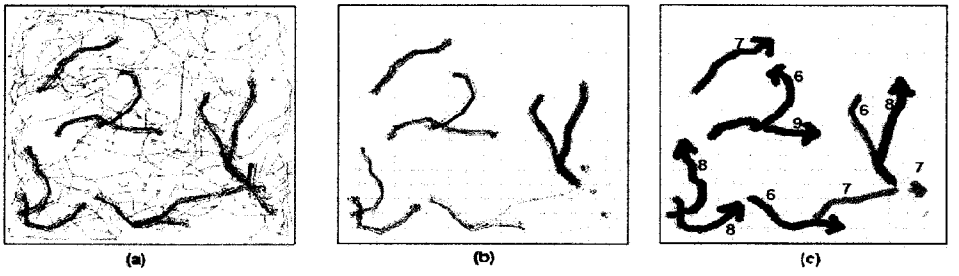

Figure 1.1: Example of OLAP for trajectories: (a) input data (b) Groups with minimum support (c) output data (aggregate trajectories and counts) [1]

minimum number of moving objects (e.g. at least 6 objects); the output in Figure 1.1 (c) are similar trajectories, which are grouped together, shown as "prominent" paths. **Group** is a minimum number of moving objects that have similar paths, and are aggregated together. The OLAP for trajectories can be described as follows:
Input: numbers of trajectories
Output: set of Groups (including Group Identifier and size of each group)

The difficulty of this research problem is to determine how to aggregate the trajectories. The trajectory data cannot simply be added together like normal aggregation in a traditional database because of its data structure. For example, the first trajectory of the moving object is $[1, x_1, y_1, t_1]$, $[1, x_2, y_2, t_2]$, $\cdots$, $[1, x_n, y_n, t_n]$, and the second trajectory of the moving object is $[2, x_{11}, y_{11}, t_{11}]$, $[2, x_{22}, y_{22}, t_{22}]$, $\cdots$, $[2, x_{nn}, y_{nn}, t_{nn}]$. The two trajectories cannot directly be added together since two moving objects have different moving locations.

Baltzer et al [1] propose Group-By operator termed Group-Trajectories to aggregate the trajectories. The Group-By operator divides the given trajectories into disjointed *groups* of trajectories, and returns *Group Identifier* and *size of each group.* The OLAP application can process with standard aggregation based on the group identifier instead of trajectories themselves. This Group-By operator is implemented by three methods: Group By Overlap, Group By Intersection, and Group-By Overlap and Intersection. Group By Overlap is intended for sequences of movement, and Group By Intersection is intended for parallel movement. The Group-By operator

first performs a high level aggregation; this process can eliminate some noise trajectories by the Frequent Itemsets Mining program (e.g. Apriori) under the conditions of minimum length and minimum support. **Noise** is a trajectory that cannot be aggregated into any groups in the group processes, and does not meet the requirements such as minimum support and minimum length. **Minimum length** is the number of frequent itemsets to be exceeded in data mining processes. **Minimum support** is the level needed to qualify as "frequent" for counts of occurrences of the same items in databases of data mining processes. The remaining trajectories are further processed by these three group methods, and then returned the group identifier and the size of each group. The result of Group-By operator can be processed like normal aggregation in TDW.

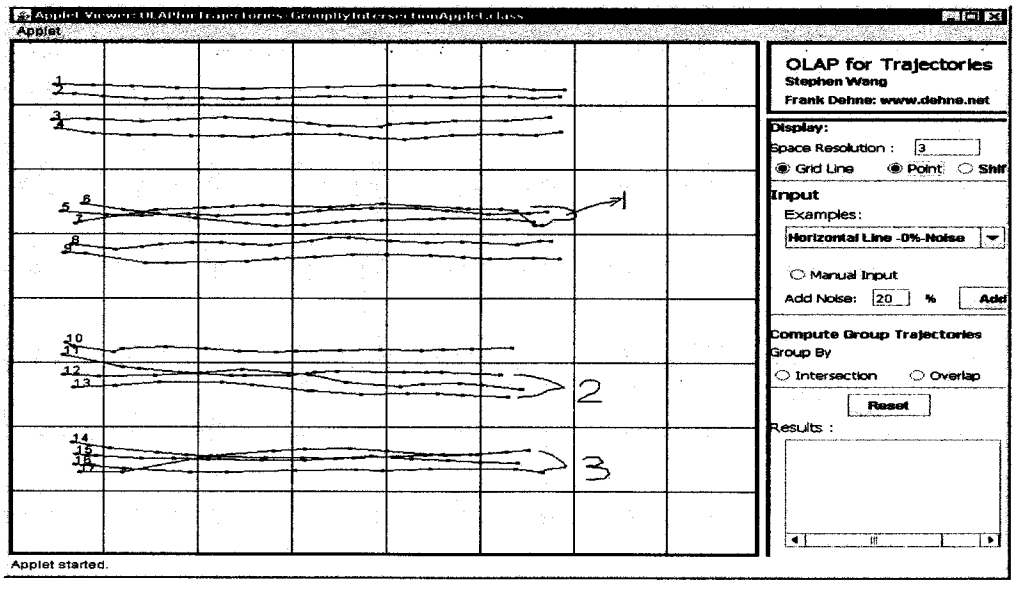

Figure 1.2: Statement of the Problem

There are two limitations of Baltzer et al's solution: firstly, some useful trajectories are eliminated by the pruning process in a high level aggregation. For example, some trajectories that are parallel with the x axis or y axis are removed by the pruning process. In Figure 1.2 on page 5. There are 17 trajectories in total. The parameters are as follows: minimum support (s) is 3, minimum length (l) is 5, and space resolution is 3. **Space resolution** (r_{space}) represents the degree to which the coordinate system

is divided into squares using the formula $2^{r_{space}}$ by $2^{r_{space}}$. For instance, the space resolution is 3, then the coordinate system is divided into 2^3 by 2^3 by squares. Based on the previous Group-By operator, the result is returned: group 1 (trajectory 5, 6, 7), group 2 (trajectory 11, 12, 13), group 3 (14, 15, 16, and 17). Other trajectories are removed by the pruning process because the minimum support is 3. Ideally, the result should be returned: group 1 (trajectory 1, 2, 3 and 4), group 2 (trajectory 5, 6, 7, 8 and 9), group 3 (trajectory 10, 11, 12, and 13), and group 4 (trajectory 14, 15, 16, and 17). Secondly, the Group-By operator requires multiple parameters such as minimum support and minimum length. These parameters are only input once, and it does not matter what result is returned. It is difficult for the OLAP user to determine what parameters should be used for obtaining a better result.

The research problem of this thesis is to propose a solution for the two limitations of previous research work: how to identify more useful trajectories that are removed by the pruning process, and what parameters can produce a better result. The input and output of this research problem are the same as the previous work. The goal of this research is to improve the result of Group-By operator: identify more useful trajectories that are removed by pruning process, and try to optimize all parameters. The Shifting Grid Algorithm is proposed for identifying more useful trajectories. The Auto Parameters Algorithm is proposed for trying to optimize all parameters. To this end, an interactive simulation GUI tool was designed and developed to validate the two proposed algorithms.

1.4 Contributions and Results

Three main contributions of this research can be summarized as follows:

1. The computing group result of Group-By operator can be improved by two new algorithms - the Shifting Grid Algorithm and the Auto Parameters Algorithm. The results show more identified trajectories than previous research [1] in most experimental cases. With the new algorithms, more trajectories are identified in the new resulting groups (See the experimental section Chapter 6). The new resulting groups are more reasonable for real world queries since the new resulting groups provide more accurate and reasonalbe information for decision

making in real applications. For example, with the new Shifting Grid Algorithm, group 1 (trajectory 1, 2, 3, 4) can be obtained in Figure 1.2 on page 5. If these trajectories were considered to be SARS virus paths, then group 1 can provide more accurate information that four people are infected. Once the health department receives the results, proper prevention procedures of SARS can be adapted in this area.

2. The new Group-By operator provides another feature: try to optimize all parameters, automatically calculate groups of trajectories without inputting parameters. The usage of Group-By operator is simpler than before, but the result of Group-By operator is much better. The Auto Parameters Algorithm calculates the combination of all parameters, and determines a better result by comparing the computing groups of trajectories. Finally, a better result is returned. The OLAP users can easily use this operator to obtain a desired group result.

3. An interactive GUI simulation tool is provided to demonstrate OLAP for trajectories. The program can directly process spatial data and temporal data experiments, and dynamically display the computing results. The interactive simulation tool provides the following functions: generating random spatial examples, drawing artificial trajectories, saving temporal data, loading temporal data files, and visualizing the spatial data. The purpose of saving the trajectory data is to allow future data analysis and compare the computing results later. The GUI simulation tool is very convenient for testing the GROUP-TRACJECTORIES operator.

The first two contributions are very valuable in the trajectory data warehousing world. The new GROUP_TRACJECTORIES operator can provide more accurate and reasonable information in decision making for OLAP users than previous research. The last contribution is very useful in the trajectory research world, since the GUI simulation tool can implement all kinds of trajectory experiments.

This research does not run the large data sets from real world scenarios. There are three reasons: firstly, it is very difficult to find real trajectory data sets since most

companies consider these data as sensitive proprietary and private data. Secondly, the experimental computer cannot process large data sets since the computation of these algorithms are very time-consuming. Thirdly, the size of large data sets can be very large, for example, Wal-Mart Corporation has terabyte RFID data. However, small data sets are used for proving the ideas and concepts of the two algorithms.

1.5 Organization of Thesis

The rest of this thesis is organized as follows: Chapter 2 introduces the background information for understanding this research. Chapter 3 summarizes the latest results of the analysis of trajectories for moving objects. Chapter 4 details the new approach for the Group-By operator. Chapter 5 describes the experimental method for demonstrating the results of this research. Chapter 6 presents the results of experiments. Chapter 7 summarizes this research work and concludes with a discussion of future work.

Chapter 2

Background

2.1 Introduction

This chapter concentrates on the background information of the OLAP tool. The chapter is divided into four sections. Section 2.2 introduces the basic concept of Data Warehousing. Section 2.3 introduces the concepts of OLAP and aggregation. Section 2.4 provides the summary for the chapter. The Group-By operator is a special category of aggregation. The aggregation is widely used for speeding up data retrieval in an OLAP application. The OLAP application is the part of data warehousing that provides reports. In order to implement the special Group-By operator, Frequency Item sets must be used. This chapter introduces the basic concepts of these topics.

2.2 Data warehousing

Data warehousing is one of the major business intelligence applications. It helps business managers to make better decisions. There are two types of data warehousing in term of datasets: traditional data warehousing [12] and trajectory data warehousing [20], [23], [16] and [7]. For the purpose of this study, this research only focuses on trajectory warehousing.

The concept of traditional data warehousing was proposed in the late 1980s by IBM researchers, Barry Devlin and Paul Murphy. The data warehouse provides an architectural model (show in Figure 2.1 on page 10) for the flow of data from operational systems to decision support environments. The process of gathering, cleaning and integrating data from various sources, usually long existing operational systems (called legacy system), is typically, in part, replicated for each environment. Operational systems are optimized for preservation of data integrity and for performance in the recording of business transactions through the use of database normalization and

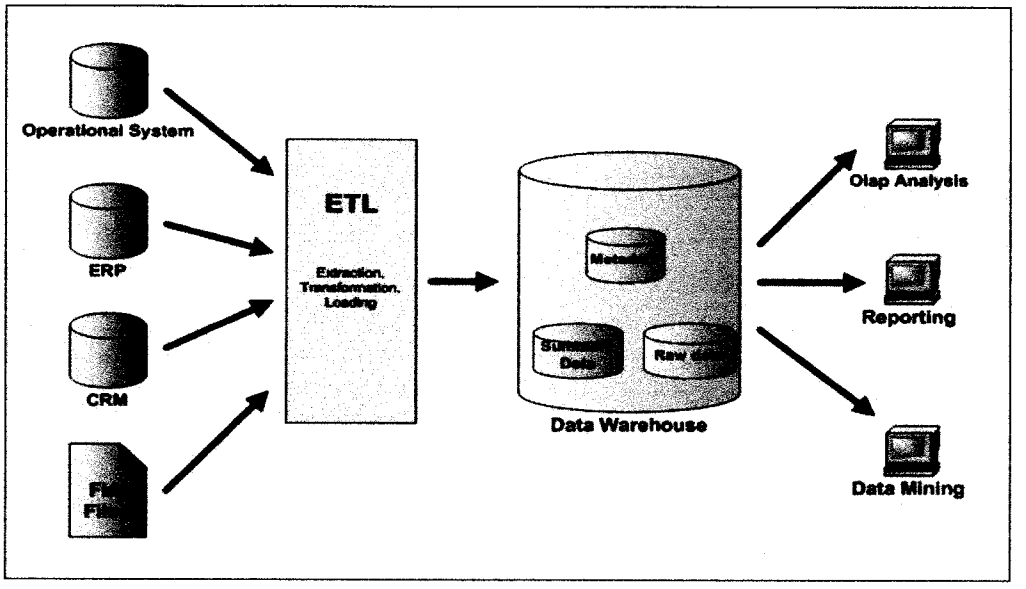

Figure 2.1: Data Warehousing Architecture Framework [13]

entity-relationship models. To speed up data retrieval, the data of data warehousing is often stored multiple times - in their granular form and in summarized forms. This is called aggregation. Data warehouse data is gathered from the operational system and held in the data warehouse even after the data has been purged from the operational system. The data warehouse architecture contains four layers: (1) operational database layer, (2) data access layer, (3) Metadata layer, and (4) information access layer. The operational database layer is the source data of a data warehouse - often comes from an organization's Enterprise Resource Planning (ERP) systems. The data access layer provides a set of tools to extract, transform and load data into the data warehouse. The metadata layer describes data about data, i.e., data dictionary. The information access layer consists of tools for data report and analysis. There are many data warehousing applications including credit card chum analysis, insurance fraud analysis and logistics management.

Traditional data warehousing systems and techniques were not designed for analyzing trajectory data. For trajectories data such as data generated by mobile devices and GPS devices, the trajectory data warehouse is required. Previously, many researchers have worked on Trajectory Data Warehousing [20], [7], [16], and [23]. A

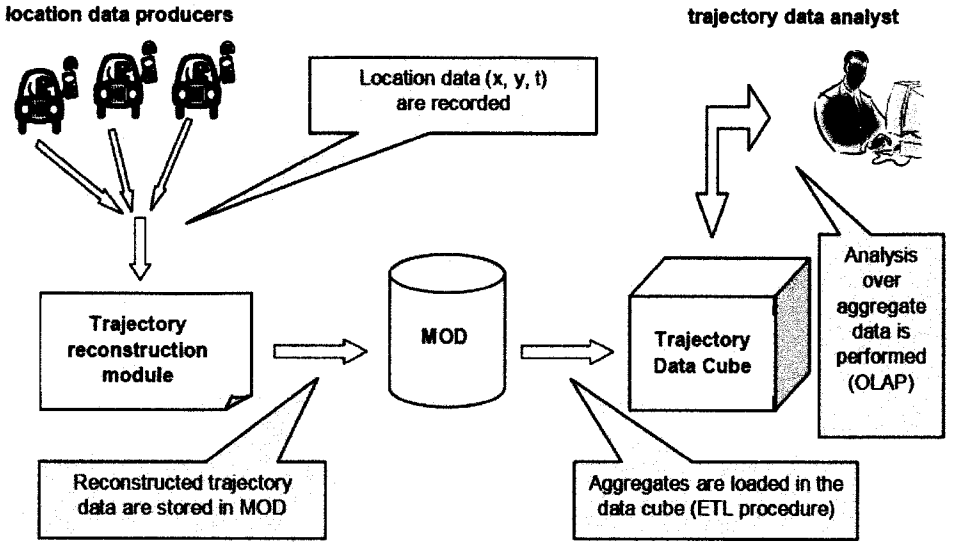

Figure 2.2: Trajectory Data Warehousing Architecture Framework[21]

framework for a trajectory data warehouse is proposed in [20]. The framework for TDW takes into consideration the complete flow of tasks required during a TDW development. The complete lifecycle is illustrated in Figure 2.2 on page 11. There are three processes. A Trajectory Reconstruction process is applied to the raw time-stamped location data in order to generate trajectories which are then stored in a Moving Object Data (MOD). Then an Extract-Transform-Load (ETL) procedure that feeds the data cube(s) with aggregate information on trajectories. The final step of the process offers OLAP (and, eventually, DM) capabilities over the aggregated information contained in the trajectory cube model. This study deals with step 3, Group-By operator is applied in the OLAP part.

The implementation of TDW is discussed in [7]. Escribano et al. [7] suggested a GIS-OLAP (Piet). As described, the Piet consists of two processes as follows. Firstly, a process called sub-polygonization decomposes each thematic layer from a GIS, into open convex polygons. Secondly, another process computes and stores in a database, the overlay of those layers for later use by a query processor. The experiment is provided: overlay precomputation can outperform GIS systems that employ indexing scheme based on R-trees.

[16] and [23] present the idea of storing and aggregating spatial-temporal patterns by using a TDW. Particularly, [16] discusses the idea of allowing analysts to quickly evaluate frequent patterns mined from trajectories of moving objects occurring in a specific spatial zone and during a given temporal interval. TDW is resorted to base on a data cube model, having spatial and temporal dimensions, discretized according to a hierarchy of regular grids, and the facts are sets of trajectories which intersect the spatial-temporal cells of the cube. The idea is to increase such a TDW with a new measure: frequent patterns obtained from a data-mining process on trajectories. As a consequence these patterns can be analyzed by the user at various levels of granularity by means of OLAP queries. [16] focuses on the extraction/mining of the patterns to be stored in each cell, which requires an adequate projection phase of trajectories before mining; and the spatial-temporal aggregation of patterns to answer roll-up queries, which poses many problems due to the holistic nature of the aggregation function. However, Orlando et al. [23] discuss the issues of storing server aggregate measures about trajectories of moving objects. They suggested that deal with the overwhelming streams of trajectory observations, possibly produced at different rates, and arrive at in an unpredictable and unbounded way; and focus on number of trajectories that lie in a spatial region during a given temporal interval. They proposed a new approach to compute an approximate, but very accurate, presence aggregate function, which algebraically combines a bounded amount of measures stored in the bas cells of the data cube.

The Group-By operator of this study is not only related to aggregation, but also related to parallel and sequence patterns for moving objects.

2.3 OLAP and Aggregation

The aggregation must be used in OLAP to accelerate data retrieval. This section discusses both OLAP and aggregation. OLAP is a method to quickly answer multi-dimensional analytical queries. OLAP is part of data warehousing. OLAP is typically applied to businesses, reporting for sales, marketing, management reporting, business process management (BPM), budgeting and forecasting, financial reporting and similar areas. The core of the OLAP system is the OLAP cube (multidimensional cube

or hypercube). It consists of numeric facts called measures which are categorized by dimensions. OLAP dimensions are further organized in hierarchies that favour the data aggregation process. The biggest challenge of trajectory OLAP is the aggregation of trajectory because trajectory is hard to represent by relational tables. Many papers have discussed the aggregation of spatial-temporal data [14], [18], [26].

The goal of OLAP analysis for trajectories is to answer aggregate queries with respect to the spatial movements of a set of objects represented in relational table objects. The main problem that arises is how to aggregate with respect to a feature dimension trajectory. It is very unlikely that any two trajectories are exactly the same. Therefore, the standard aggregation method does not fit the trajectory OLAP cube. The most challenging problem of aggregation is to filter the noise data, and to gather the most useful trajectories together. The most common method is to use data mining techniques. There are many aggregation methods as described in [18]. Although most trajectory aggregations apply the data mining techniques [22], this research does not discuss them here since the existing data mining program (Borgelt's data mining program [3]) is used. This study concentrates on a Group-By operator GROUP_TRAJECTORIES [1] for computing trajectory aggregation.

2.4 Summary

This chapter introduces the basic concepts of data warehousing, OLAP and aggregation. The next chapter summarizes other relevant research on analysis of trajectories for moving objects: reviews Blazter et al's work since this thesis is based on Blazter et al's research, and discusses other relevant works.

Chapter 3

Literature Review: Trajectories for moving objects

3.1 Introduction

In recent years, many researchers have been working on trajectories for moving objects. These research areas include the trajectory data warehouse, trajectory OLAP and trajectory aggregation techniques. This research only focuses on trajectory aggregation techniques.

This chapter is organized as follows: a description of Baltzer et al's [1] framework of Group-By operator and its three implementations, and summaries of other popular analysis of trajectories for moving objects that have emerged in recent years.

3.2 OLAP for Trajectories

Baltzer et al propose a solution for how to aggregate trajectories in OLAP for trajectories, where the input is the number of trajectories, and output is the group identifier and size of each group. The trajectories cannot directly process normal aggregation like traditional databases since they do not have a direct relationship, and any two trajectories are unlikely exactly the same. Baltzer et al propose a Group-By operator termed Group-Trajectories to aggregate trajectories. The result of the Group-By operator can be processed like normal aggregation. The usage of the Group-By operator is as follows:

Select AGGREGATE (trajectory) AS trajectory
    COUNT (trajectory) as count
FROM objects
GROUP BY **GROUP_TRAJECTORIES (trajectory, resolution)**
HAVING COUNT () ≥ 5

The Group-By operator is implemented in three methods: Group By Overlap,

Group By Intersection, and Group By Overlap and Intersection. The Group By Overlap is intended for sequences of movement. The Group By Intersection is intended for parallel movements. The Group By Overlap and Intersection is the combination of Group By Overlap and Group By Intersection. However, this paper only reviews Group By Overlap and Group By Intersection since the new Shifting Algorithm and Auto Parameters Algorithm are applied to these two group methods.

The high level trajectory aggregation framework (Figure 4.2 on page 27) is executed before the three group methods are used. The high level trajectory aggregation can eliminate some noise data to be further processed by the three group methods. There are three steps in high level aggregation framework: Firstly, map all trajectories $\mathcal{T}$ into a 2D (time, space) coordinate (line 1 in Figure 4.2). The result of mapping produces trajectories $\mathcal{T}'$. Second, compute the frequent itemsets for the mapping set of trajectories $\mathcal{T}'$ (line 2 in Figure 4.2). Thirdly, match the frequent itemsets f with trajectories, which trajectories contains f, and then put these trajectories into one group (line 3-7 in Figure 4.2). In other words, each frequent itemset f corresponds to an original group c of trajectories and creates $\mathcal{C}$ of resulting (f, c). The result of high-level trajectory aggregation returns a set of groups that have common frequent itemset.

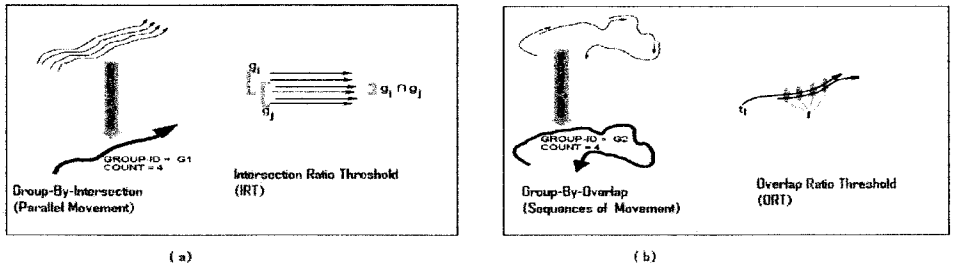

Figure 3.1: Group-By operator: (a) Group By Intersection with Intersection Ratio, and (b) Group By Overlap with Overlap Ratio) [1]

Once the result of high-level trajectory aggregation is returned, one of three group methods is chosen to calculate the groups of trajectories. Only Group By Overlap (Figure A.2 on page 65) and Group By Intersection (Figure A.3 on page 66) are introduced here. Group By Overlap is intended for sequences of movements. Group

By Overlap is to create a graphic $\mathcal{F}$, where each vertex corresponds to a trajectory, and all pairs of trajectories are edges of $\mathcal{F}$. All edges with label Overlap Ratio (Figure 3.1 (b) on page 15) $OS = \frac{2.|f|}{|t_i|+|t_j|}$ are computed, where $|f|$ is the size of frequent itemset (common items) for trajectory t_i and t_j. A value of the parameter Overlap Ratio Threshold is given. The trajectory t_i and t_j is connected if $OS \geq ORT$. Then calculate all pairs of trajectories, remove the edges if $OS < ORT$. And then remove a single vertex (trajectory) since one trajectory cannot be a group. Finally, once $\mathcal{F}$ is created completely, the identifier (Group Identifier) of the connected edges (Groups) and the size of connected edges (Groups) are returned.

Group By Intersection is intended for parallel movement for moving objects. The procedure of Group By Intersection is described as follows: firstly, use the result from the high-level trajectory aggregation, and crate an initial set $\mathcal{G}$ groups of trajectories, where each group c corresponds to a frequent itemset f in reverse matching of high-level trajectory aggregation process. A group strength GS is assigned to each group, where the value of GS is the size of the corresponding frequent itemset. Secondly, merge all pairs of groups in $\mathcal{G}$ by iterating the following loop: calculate the intersection ratio (Figure 3.1 (a) on page 15) $AS(g_i, g_j)$ of each pair $g_i, g_j \epsilon \mathcal{G}$, where $AS(g_i, g_j) = min(\frac{|g_i \bigcap g_j|}{g_i}, \frac{|g_i \bigcap g_j|}{g_j})$ is the number of trajectories that occur in both g_i and g_j, relative to the sizes of g_i and g_j. The value of Intersection Ratio Threshold (IRT) is given. If $AS(g_i, g_j)$ is greater than IRT, then merge g_i and g_j with merge strength MS ($MS(g_i \bigcup g_j) = \frac{GS(g_i)+GS(g_j)}{2}$). Otherwise, $MS(g_i \bigcup g_j)$ is assigned the value 0. All candidate pairs are ranked by their MS, and then merged the pair group g_{i*}, g_{j*} with maximum MS. The loop process is repeated until all MS are 0. Finally, the Group Identifier of groups and size of each group are returned.

Baltzer et al also discuss the combination method of group by overlap and intersection, but the new algorithms only apply to Group By Overlap and Group By Intersection.

Baltzer et al present a framework of Group-By operator and three group methods, but there are limitations of the Group-By operator: some useful trajectories are removed by the pruning process in high-level trajectory aggregation; and the parameters are not adjustable. The goal of this research is to improve the result of

Group-By operator by identifying more useful trajectories, and trying to optimize the parameters.

3.3 Other relevant studies

Other studies also cover the topic of analyzing the trajectories for moving objects, but they mainly focus on these five areas: discovering the patterns of movement for moving objects, exploring the features of cluster moving objects, detecting regular visiting patterns in terms of time and space dimensions, finding "popular" locations for moving objects, and grouping trajectories.

The first research area of trajectories focuses on discovering the new patterns of moving objects through trajectories [15], [8], [19], [2] and [28]. [15] describes a geographic data mining approach to detecting generic aggregation patterns such as flocking behaviour and convergence of geospatial lifeline data. This method considers the object's motion properties in an analytical space as well as spatial constraints of the object's lifelines in geographic space. The paper also discusses the geometric properties of the formalized patterns with respect to their efficient computation. [8] describes an extension of the sequential pattern mining paradigm that analyzes the trajectories of moving objects. The trajectory patterns describe frequent behaviours of moving objects based on space, i.e., the duration of movements. A general formal statement of the novel mining problem is defined and then several different instantiations of different complexity are studied. [19] presents periodic patterns which means the objects follow almost the same routes over regular time intervals. The framework that analyzes, manages, and queries object movements is proposed based on the periodic pattern. The fast mining algorithm for retrieving maximal periodic patterns and a specialized index structure are proposed for solving periodic mining problem. [2] discusses reporting flock patterns based on the trajectories of wildlife GPS tracking. The information is hidden in large data sets in the form of interesting patterns, where a pattern can be any configuration of some moving objects in a certain area and /or objects moving along paths close to each other for a certain pre-defined time. The paper describes a new definition which is more realistic than the previous ones, using techniques from computational geometry. The fast algorithm is proposed to detect

and report flocks and is analyzed both theoretically and experimentally. [28] describes the concept of flow patterns. The flow patterns are intended to describe the change of events over space and time. The paper presents a disk-based algorithm, FlowMiner, which utilizes temporal relationships and spatial relationships amid events to generate flow patterns. All of these papers tend to find certain patterns such as flock patterns and periodic patterns. This research concentrates on reporting parallel and sequences patterns and works for in an OLAP environment.

The second research area of trajectories is in exploring cluster moving objects [22] and [17]. [22] presents the clustering problem applied to the trajectory data domain, and proposes an adaptation of a density-based clustering algorithm to trajectory data based on a simple notion of distance between trajectories. The density-based clustering method for moving object trajectories according to [22], is aimed at properly exploiting the intrinsic temporal semantics for the purpose of discovering interesting time intervals, where and when. The quality of the achieved clustering is optimal. Temporal focusing is sketched, with the aim of exploiting the intrinsic semantics of the temporal dimension to improve the quality of trajectory clustering. [17] describes how to catch interesting pattern changes during the motion process and provides better insight into the essence of mobile data points. Micro-clustering is employed in order to catch the spatial-temporal regularities of moving objects and handle large amounts of data. Efficient techniques are proposed to keep the moving micro-clusters geographically small. Important events such as the collisions among moving micro-clusters are also identified. High quality moving micro-clusters are dynamically maintained which leads to fast and competitive clustering result at any given time instance. The two papers are aimed at clustering moving objects, but this study focuses on general moving objects.

The third research area of trajectories concentrates on detecting time dimension in terms of time and space dimensions [6], [24]. [6] describes how to detect regular visit patterns based on the dimensions. The purpose is to solve the following problem: a trajectory T and an area A are given, T might intersect with A several times, and the aim is to detect whether T visits A with some regularity. The problem is similar to bitstrings LDS (LONGEST DENSE SUBSTRING). The bits of the bitstring

correspond to a sequence of regular time points. LDS is a core problem for many applications that aim at detecting the regularity of T intersecting A. An optimal algorithm to solve LDS is proposed. [24] describes a Space-Time GIS Approach to Exploring Large Individual-based Spatiotemporal Datasets. The fundamental concept in this approach is to derive a small number of representative space-time paths (GSTP) approach to facilitating visualization and exploiting spatiotemporal changes among individuals in a large dataset. A Space-time GIS environment is developed to implement the GSTP concept. The operational space-time GIS prototype is developed to provide a proof-of-concept study of this approach in ESRI's ArcScene and ArcMap. These two papers tend to find pattern paths, but this research is inclined to group the paths of trajectories, and apply it to OLAP applications.

The fourth research area is interested in finding "popular" locations [2] and [31] or trajectories [25], [4] and [30]. [2] discusses finding "popular place" through analyzing spatio-temporal movement patterns in large tracking data sets which are captured by location aware devices such as GPS devices. The place is popular since it is often visited by many entities such as people, animals and other moving objects. [31] presents an idea for capturing personal, meaningful place by two spatio-temporal clustering algorithms TDJ and R-TDJ from personal paths. [25] describes a similar trajectory retrieval schema for efficient retrieval on both a single trajectory of a moving object and multiple trajectories of multiple moving objects. The similar trajectory retrieval scheme can support multiple properties including direction, distance, and time and can provide the approximate matching that is superior to the exact matching. The paper also presents the k-warping distance algorithm which enhances the existing time warping distance algorithm by permitting up to K replications for an arbitrary motion of a query trajectory so that the similarity between two trajectories can be measured accurately. [4] presents an algorithm based on time. Similarity measures for two trajectories are their average distance at corresponding times. The algorithm can be used to find the most similar subtrajectories under this measure. $(1 + \varepsilon)$-approximation algorithm is also given for finding the most similar subtrajectories with a time shift, for both cases where the duration is specified and where a minimum duration is specified. [30] introduces a distributed Spatio-Temporal Similarity

Search problem: give a query trajectory Q, then find the trajectories that follow a motion similar to Q when each of the target trajectories is segmented across a number of distributed nodes. Two novel algorithms UB-K and UBLB-K are proposed. The solutions can be applied in various domains such as cellular networks, wildlife monitoring and video surveillance. These approaches are proposed to find the "popular" locations, but they do not deal with the group trajectories.

The fifth research area focuses on grouping trajectories [27], [29], [11], [5], and [9]. [27] describes techniques for analysis and retrieval of object trajectories in a two or three dimensional space. The non-metric similarity functions are formalized based on the Longest Common Subsequence (LCSS) which are very resistant to noise and furthermore it provides an intuitive notion of similar portions of the sequences. Efficient approximate algorithms are provided to compute these similarity measures. The new algorithm shows better performance based on Euclidean and Time Warping distance functions, especially under the strong presence of noise. [29] presents a group pattern mining approach to deriving the grouping information of mobile device users based on the spatio-temporal distances among them. Group patterns of users are determined by a distance threshold and a minimum duration. The **A**priori-like algorithm for mining valid **G**roup **P**atterns (AGP) and **V**alid **G**roup (VG)-growth algorithms are derived from the Apriori and **F**requent **P**attern (FP)-growth algorithms, respectively, to discover group patterns. Similarly, [11] introduces another group pattern mining approach to deriving the grouping information of mobile device users based on a trajectory model. ATGP algorithm and TVG-growth come from the Apriori and VG-growth algorithms respectively in order to discover group patterns. [5] describes the problem of discovering sequential patterns, which are routes frequently followed by the object. The pattern elements are defined as spatial regions around frequent line segments. The proposed algorithm finds patterns by employing a newly proposed substring tree structure and improves Apriori technique. [9] presents a database projection based on method of efficiently extracting such long, shareable frequent routes. The method prunes the search space by making use of the minimum length and shareable requirements and avoids the generation of the exponential number of subroutes of long routes. A SQL-based implementation is described. Experiments on

real life data show the effectiveness of the method. This study is very close to this research area, but the algorithm of this paper can be more widely applied to grouping trajectories.

3.4 Summary

This chapter briefly discussed Baltzer et al's work, and reviewed other related limitations of analyzing the trajectories. The next chapter describes the new Shifting Grid Algorithm and Auto Parameters algorithm, as well as how they are applied to both Group By Overlap and Group By Intersection.

Chapter 4

Description of Algorithmic Designs

4.1 Introduction

As discussed in previous chapters, the goal of this research is to improve the result of the Group-By operator based on Baltzter et al's work [1]: identify more trajectories, and try to optimize all parameters of the Group-By operator. The Shifting Algorithm is proposed for identifying more trajectories. The Auto Parameters Algorithm is proposed for trying to optimize all parameters of the Group-By operator and simplifying the usage of the Group-By operator. This chapter formally introduces these two new algorithms and algorithmic designs. The rest of this chapter is organized as follows: Section 4.2 describes the basic algorithm, including the Shifting Grid Algorithm, Auto Parameters Algorithm and Compare Results Algorithm. Section 4.3 discusses the design issues, including the *GC* parameter and time-consuming problem. Section 4.4 provides the summary for this chapter.

4.2 The Basic Algorithm

4.2.1 The Shifting Grid Algorithm

In Blatzer et al's work, trajectories are mapped into a 2D (time resolution r_{time}, space resolution r_{space}) coordinate system (line 1 in Figure A.1 on page 64). A location of the moving object is represented as a point. The coordinate system is equally divided into $2^{r_{space}}$ by $2^{r_{space}}$ squares. Space resolution (r_{space}) represents the aggregation level of the OLAP cube. The coordinate system is divided into a grid pattern so that different trajectories that have several positions based on movement can be located in common grids. An identifier of each cell in the grid, called **GridID**, is an integer, and is made of x and y values. In order to compute the frequent itemsets, all trajectories need to be represented by GridID because mining x and y is not meaningful and is

time-consuming. The pattern of trajectories represented by GridID is called **GridID patterns**. For example, a moving object has a moving location (id, 385, 450, 10), and other parameters are: r_{space} is 3, both minimum x and y of the coordinate system are 0, and both maximum x and y of the coordinate system are 800. The bound of this coordinate system is 800 by 800 based on minimum and maximum x and y values. Therefore, the coordinate system is equally divided into 2^3 by 2^3 partitions, where each grid size is $800/(2^3) = 100$. x for GridID is 3 (385/100), and y for GridID is 4 (450/100). In order to convert x and y into one integer, an assistant value "1000" is introduced. The value "1000" is chosen because it is not too big or too small, and it is easy to know where the point is located in the coordinate system, when a GridID is divided by 1000. Hence, the GridID of (id, 385, 450, 10) is 3004 (1000*(x for GridID) +(y for GridID)). The trajectory can be converted into GridID patterns. For example, a trajectory (1, 220, 234, 10), (1, 320, 430, 20), (1, 420, 534, 10), is presented by GridID pattern: (1, 2002, 3004, 4005), where "1" comes from object ID 1, "2002" represents (1, 220, 234, 10), "3004" represents (1, 320, 430, 20), and "4005" represents (1, 420, 534, 10). In computing groups of trajectories, it makes senses to mine data in terms of GridID. Two problems will arise if only x and y are mined: firstly, the data is not meaningful for trajectories, and secondly, the data is very large. Borgelt's Apriori program [3] (Frequent Item Sets Program) is used for mining GridID pattern data (line 2 in Figure A.1 on page 64). Most noise trajectories can be pruned by the data mining program since the noise does not meet the requirements of minimum length and minimum support. The remaining trajectories are further processed by the Group-By methods.

One limitation of the previous research is the removal of some "useful" trajectories. To identify more trajectories that are removed by the pruning process, the proposed solution is to shift the grid. The Shifting Grid Algorithm can identify more trajectories because some useful trajectories are added into the original group result after shifting grid. When shifting grid, three situations (Figure 4.1 on page 24) are generated: firstly, more trajectories can be grouped into one group after the mining process; secondly, some trajectories are not grouped into one group, but the shifting trajectories can generate another group that has common trajectories with the

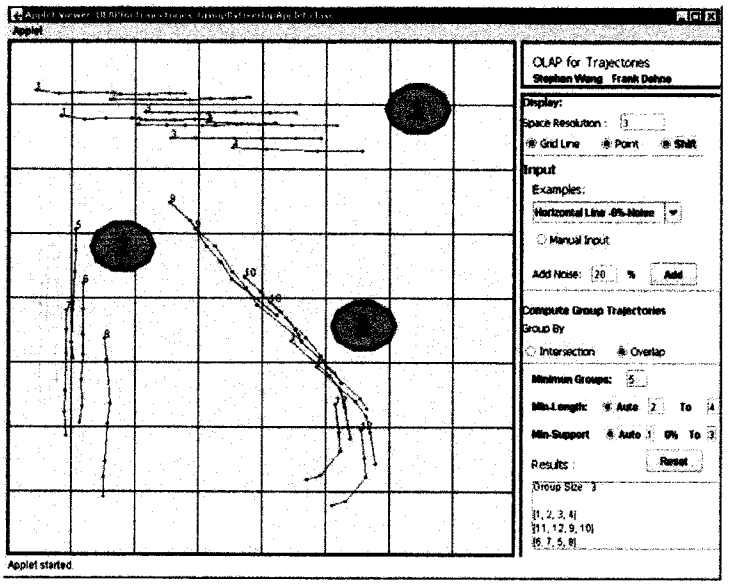

Figure 4.1: Three situations after shifting grid

groups of original data mining group sets; thirdly, the result produces the same as the original data set mining group result. There are two ways of implementing Shifting Grid: the whole coordinate system is shifted for a certain distance, and all moving objects are shifted for a certain distance. For instance, all moving objects are moved the same distances such as 20% or 30% of each grid unit size. In this research project, the second method is chosen because it is easier to implement.

The shifted data set is generated by adding a shifting distance d to both x and y. The shifting distance d is grid unit size $\times$ shifting percentage, where **the shifting percentage** is the shifting distance divided by the grid unit size. For example, if the grid unit size is 100, the shifting percentage is 10%, and then d is 10. For instance, if an original trajectory is (1, 220, 234, 10), (1, 320, 430, 20), (1, 420, 534, 30), then the shifting trajectory data is (1, 220 + 10, 234 + 10, 10), (1, 320 + 10, 430 + 10, 20), (1, 420 + 10, 534 + 10, 30). If both x and y are added the same distance, then both x and y can keep positive values. If both x and y are subtracted by the same distance, then x or y, or both may produce negative values. If negative values are generated, then the computation is more complex. Therefore, in this research, the shifting distance value will be added. If all trajectories only shift once, the result

may be better, or even worse. Therefore, multiple times shifting grids are necessary. The Auto Parameters Algorithm is proposed for the purpose of multiple times grid shifting.

Although the shifting concept has been mentioned in some papers, e.g. shifting time [4] and shifting space [10], the Shifting Grid concept is new. It is applied to the high-level trajectory aggregation framework Algorithm, called the Shifting Grid Algorithm. The purpose of this algorithm is to identify more trajectories for further processing by the group methods.

The detailed Shifting Grid Algorithm is implemented using the following procedures:

1. All trajectories $\mathcal{T}$ are mapped into a 2D coordinate system (line 1 in Figure 4.2 on page 27). All trajectories $\mathcal{T}$ convert into GridID patterns $\mathcal{T}'$. Each trajectory is processed as one transaction in the data mining program. Borgelt's Aprori [3] program mine the frequent itemsets $\mathcal{F}$ based on GridIDs patterns $\mathcal{T}'$ (line 2 in Figure 4.2 on page 27). The procedure of reverse matching is used to determine which trajectories contain all items of each frequent itemset and to put these trajectories into one group (line 3 in Figure 4.2 on page 27). The first mining groups $\mathcal{C}$ are generated under the conditions of minimum support and minimum length.

2. The shifting data set $\mathcal{T}_{shifting}$ is generated based on the original trajectory data $\mathcal{T}$ and shifting percentage (line 4 in Figure 4.2 on page 27). The shifting distance is calculated using the formula d = grid unit height $\times$ shifting percentage. For example, if one trajectory is (Id, x_1, y_1, t_1), $(Id, x_2, y_2, t_2) \cdots (id, x_m, y_m, t_m)$, then the shifting trajectory is:
$(Id, x_1 + d, y_1 + d, t_1)$, $(Id, x_2 + d, y_2 + d, t_2 + d) \cdots (id, x_m + d, y_m + d, t_m + d)$

3. The shifting data sets $\mathcal{T}_{shifting}$ are processed by the same procedures in Step 1 (line 5 in Figure 4.2 on page 27). All shifting trajectories are mapped into a 2D coordinate system, and are converted into GridID patterns. Each trajectory is processed as a transaction by Borlget's Aprori program. The same reverse matching procedure of Step 1 is processed. The second mining groups $\mathcal{C}_{shifting}$

are generated based on the shifting trajectories $\mathcal{T}_{shifting}$.

4. The first resulting groups $\mathcal{C}$ and the second resulting groups $\mathcal{C}_{shifting}$ are merged together (line 3 in Figure 4.2 on page 27). Any two groups, if they have common trajectories, are merged together. Then the set of groups $\mathcal{C}$ with more trajectories are generated.

5. Process Group By Overlap or Group By Intersection with the groups $\mathcal{C}$ (line 7-8 in Figure 4.2 on page 27).

The trajectory IDs of shifting data sets are stored after the reversed procedure is completed. The shifting trajectory data sets can be discarded since the trajectory IDs need to be further processed. In order to validate this algorithm, the shifting trajectory data sets are stored. An interactive GUI simulation tool (see Section 5.2.2) is able to visualize both the original trajectory data sets and the shifting trajectory data sets. The two trajectory datasets are shown with different colors in the GUI simulation tool.

Group By Overlap and Group By Intersection can be computed the same way as Blatzer's work once the larger set $\mathcal{C}$ of resulting (f, c) pairs are obtained. The only thing that changes is line 5 in Figure A.2 on page 65. The size of both common itemsets of t_i and t_j are recomputed since this is more accurate than using the common number of trajectories. Although this procedure takes more time, it is more accurate than using the size of frequent item sets.

4.2.2 Algorithm 2: Compare Results Algorithm

In Auto Parameters Algorithm (Section 4.2.3 on page 28), multiple results are generated. The Compare-Results Algorithm is proposed for determining which result is better. In order to determine a better result, a reference value, **Group Cardinality**(GC) is introduced (See Section 4.3.1 on page 30), and the average number of trajectories for each group need to be calculated. **The average number of trajectories** for each group is represented as $\overline{T}$, and is computed by the formula $\overline{T} = \frac{\text{Sum of All Trajectories in Current Computing Result}}{\text{size of Groups}}$. The size of the resulting

The Shifting Grid Algorithm

Input:

1. set T of trajectories,
2. space resolution r_{space},
3. time resolution r_{time},
4. minimum support s,
5. shifting percentage X,
6. shifting percentage Y,
7. minimum length l.

`Output: set of Groups` $\mathcal{G}$

1. map T to resolution (r_{space}, r_{time}) resulting in T'
2. compute frequent itemsets $\mathcal{F}$ in T' with
   minimum support s and minimum length l
3. reverse matching:
   $f \longrightarrow (f, c)$, $c \subseteq \mathcal{T}$
   $\mathcal{C}$: set of (f, c) pairs clique
4. Generate the $T_{shifting}$ based on X, Y and T
5. repeat steps from line 1 to 3 for $T_{shifting}$
   $f_{shifting} \longrightarrow (f_{shifting}, c_{shifting})$,
   $c_{shifting} \subseteq \mathcal{T}_{shifting}$
   $\mathcal{C}_{shifting}$: set of $(f_{shifting}, c_{shifting})$ pairs clique
6. $\mathcal{C} \longleftarrow \mathcal{C}$ merge $\mathcal{C}_{shifting}$

Group Merging

7. (a) Group By Overlap (A.2 in Appendix A)
8. (b) Group By Intersection (A.3 in Appendix A)

Figure 4.2: Algorithm 1: The Shifting Grid Algorithm

groups is represented by an absolute symbol, e.g. the size of a better result is presented by $|\mathcal{G}_{better}|$. A result that has a bigger average number of trajectories for each group and the size of groups is closer to GC, is considered **a better result**. "Closer to GC" means: the size of group minus GC has a bigger value, for instance, if $|\mathcal{G}_{better}|$ -GC is greater than $|\mathcal{G}_{computing}|$ - GC, then $\mathcal{G}_{better}$ is closer to GC, where $|\mathcal{G}_{computing}|$ is the size of current computing result. **Better results** can be obtained with more trajectories on average, and larger sized groups, though it may be not an optimal result. Parameters that correspond to a better result are called **better parameters**; similarly, these parameters may not be optimal.

The Compare-Results Algorithm (Figure 4.3 on page 29) is described as follows: If the size of $|\mathcal{G}_{better}|$ is equal to the size of $|\mathcal{G}_{computing}|$, or both sizes are greater than GC, then the new value of $\mathcal{G}_{better}$ is determined by the bigger average number of trajectories $\overline{T}$ (Line 1 -3 in Figure 4.3 on page 29). If both $|\mathcal{G}_{better}|$ and $|\mathcal{G}_{computing}|$ are less than GC, then the new value of $\mathcal{G}_{better}$ is determined by whose result is closer to GC (Line 4 -6 in Figure 4.3 on page 29). If other situations exist, then $\mathcal{G}_{better}$ is determined by whose result is closer to GC (Line 7 in Figure 4.3 on page 29). The detailed algorithm is described as Figure 4.3 on page 29.

4.2.3 Algorithm 3: Auto Parameters Algorithm

The Auto Parameters Algorithm is proposed for improving the resulting groups of Group-By operator, and trying to optimize all parameters and simplify the usage of Group-By operator. The Auto Parameters Algorithm can improve the resulting groups of Group-By operator because the Shifting Grid Algorithm runs multiple times, and a better result is determined from multiple results.

The Group-By operator must provide the parameters when it is executed in previous research. These parameters are "static". All current parameters in research are assigned a proper range; the values of these parameters can be changed. These parameters are called "dynamic" parameters. These "dynamic" parameters can be set up in the initial environment of a database, and then they do not need to be provided when the Group-By operator is used.

In the algorithms of the Group-By operator, many parameters such as minimum

Compare-Results Algorithm

INPUT:

1. group cardinality GC.
2. better group result $\mathcal{G}_{better}$,
3. current computing group$\mathcal{G}_{computing}$,

Output: set of groups $\mathcal{G}_{better}$

1. if $|\mathcal{G}_{better}| == |\mathcal{G}_{computing}|$, or
2. $|\mathcal{G}_{better}|$ and $|\mathcal{G}_{computing}| > GC$
3. Choose a better group result with bigger $\overline{T}$
4. Endif
4. if $|\mathcal{G}_{better}|$ and $|\mathcal{G}_{computing}| < GC$
5. compare $|\mathcal{G}_{better}|$ -GC, $|\mathcal{G}_{computing}|$ -GC
   Choose a better group result with whose value is closer to GC
6. Endif
7. ELSE choose the group result with cardinality closer to GC

Figure 4.3: Algorithm 2: Compare Results Algorithm

length, minimum support, shifting percentage x and y are required. The idea of the Auto Parameters Algorithm is to avoid manually inputting these parameters. The new algorithm is able to automatically search parameters. It is called the Auto Parameters Algorithm. The goal of the Auto Parameters Algorithm is to make the Group-By operator simple, e.g.:

Select AGGREGATE (trajectory) AS trajectory
    COUNT (trajectory) as count
FROM objects
GROUP BY GROUP_TRAJECTORIES (trajectory)

The basic idea of the Auto Parameters Algorithm is to run all combinations with all parameters such as space resolution ($r_{min-space}$, $r_{max-space}$), minimum support (s_{min}, s_{max}), minimum length (l_{min}, l_{max}), Group Cardinality (GC), and group ratio ORT or IRT(gr_{min}, gr_{max})(Line 1-5 and Line 9-13 in Figure 4.4 on page 31), where all parameters are integer. Each parameter is limited in value from minimum to maximum. One of the group methods is applied (Line 6 in Figure 4.4 on page 31).

The first result of the combination of all parameters is assigned to initialize the better group $\mathcal{G}_{better}$ (Line 7 in Figure 4.4 on page 31). Each current computing set of groups $\mathcal{G}_{computing}$ is compared with $\mathcal{G}_{better}$ by the Compare-Results Algorithm (Line 8 in Figure 4.3 on page 29). Finally, the better result $\mathcal{G}_{better}$ is returned once all combinations are computed completely (Line 14 in Figure 4.4 on page 31).

This Auto Parameters Algorithm also matches the OLAP concepts since multiple space resolutions can be computed at same time. If all combinations are stored, then these results can be directly used by the trajectory OLAP cube as it drills down or rolls up. However, since storing all the group results requires a large amount of disk space, the experiment results do not store all results of all combinations. Only the better result and current computing group result are stored. The detailed algorithm is described in Figure 4.4 on page 31.

4.3 Discussion of Design Issues

4.3.1 *GC* parameters

The dilemma situations are met when designing the Compare-Results Algorithm: if the bigger average trajectory size is considered as the only factor, in most cases, it works. However, in some situations, it does not work very well. For example, if the trajectories are mapped onto a lower space resolution, e.g. the space resolution r_{space} is 1, the bigger average trajectory number is obtained since there are only four grid cells, but each group contains many trajectories. In this situation, the number of groups is too small. If the size of the group sets is considered as the only factor, similarly, in some cases, it works, but a problem may occur when many groups are computed in higher space resolution. For instance, when the space resolution has a large value (e.g. 10, $2^{10} = 1024$), many groups can be found, but each group may only contain a few trajectories. This is not an expected result since there are too many groups. In this situation, the size of the groups is too big.

It is important to consider group size because if the size of groups is not appropriate, results may differ. In a multi-dimensional world, this can occur so a new parameter Group Cardinality (GC) is proposed. It refers to the expectations that

Auto Parameters Algorithm

DEFAULT INPUT:

1. minimum level $r_{min-space}$,
2. maximum level $r_{max-space}$,
3. minimum support s_{min},
4. maximum support s_{max},
5. minimum length l_{min},
6. maximum length l_{max},
7. minimum shifting g_{min}
8. maximum shifting g_{max}
9. minimum group Ratio (IRT or ORT) gr_{min}.
10. maximum group Ratio(IRT or ORT) gr_{max}.
11. group cardinality GC.

`Output:` a better set of groups $\mathcal{G}_{better}$

1. For (r_{space}: from $r_{min-space}$ to $r_{max-space}$)
2. For (Min-support: from s_{min} to s_{max})
3. For (Min-length: from l_{min} to l_{max})
4. For(Grid Shifting: from g_{min} to g_{max})
5. For(Group Ration:from gr_{min} to gr_{max})
6. Group By Overlap or Group By Intersection
7. Initial $\mathcal{G}_{better}$ with the value of the first result of the first parameters' combination
8. Compare-Results($\mathcal{G}_{better}$, $\mathcal{G}_{computing}$)
9. Endforgroupratio
10. Endforshifting
11. EndforLength
12. EndforSupport
13. EndforSpaceResolution
14. Return $\mathcal{G}_{better}$

Figure 4.4: Algorithm 3: Auto Parameters Algorithm

OLAP users have in terms of how many groups the Group-By operator will produce. As a matter of fact, the actual size of the better result can be bigger or smaller than GC, or the same with GC.

4.3.2 Time-consuming problem

Each algorithm in this research requires much computation including mapping raw trajectory datasets onto coordinates, running data mining programs, and merging group results. These processes consume much time and space, time in particular. In the case of the Auto Parameters Algorithm, the combination of all parameters (space resolution r_{space}, minimum length l, minimum support s, shifting percentage g and IRT or ORT) must be calculated many times throughout the whole process. For example, each parameter runs 9 times in Auto Parameters Algorithm, then it is required to run 59,049 times, i.e., 9 (space resolution) x 9 (support) x 9 (length) x 9 (shifting percentage) x 9 (IRT). It may take days when the trajectory datasets are large (e.g. terabyte RFID data). The problem of time-consuming is that it is difficult to implement large experimental datasets on a personal computer.

The solution of time-consuming problem is to use small datasets in this research. Small datasets are used to verify the idea of the two algorithms in the experiments which are described in chapter 6. In real world, the time-consuming problem will not be a big problem in an enterprise environment since enterprises may need to run the processes once a week or once a month for OLAP applications, and they have better high-end computing machines. Another strategy of the time-consuming problem is to use binary search for particular parameters such as shifting percentage and min-length.

4.4 Summary

In conclusion, the Shifting Grid Algorithm, Auto Parameters Algorithm, and Compare-Results Algorithm are described, as well as the design issues such as parameters and time-consuming problem in the two main Algorithms are discussed. The next chapter describes experimental procedures and testing, including the experimental environment and the testing methodology.

Chapter 5

Experimental Procedures and Testing

5.1 Introduction

This chapter introduces the experimental environment and testing methodology, and is divided into three sections. Section 5.2 describes the experimental environment, including hardware and software settings, as well as the GUI simulation tool. Section 5.3 introduces testing methods, including tests with and without noise, and implementing for four special types of trajectories such as vertical, horizontal, diagonal, and spiral.

5.2 Experimental Environment

5.2.1 Hardware and Software

All experiments are implemented on a personal computer. The hardware configuration of the personal computer is described as follows: one Intel Quad Core 2.40 GHz processor, 4GB DDR2 RAM, and 300GB SATA hard drive. The software configuration of this computer is the following: Windows Server 2008 Enterprise 64bit Edition operation system and NetBean 6.5.1 integrated development environment.

5.2.2 GUI Simulation Tool

There are two reasons for developing a GUI simulation tool for this research. Firstly, a tool needs to generate experimental data since this research requires many experiments to validate the proposed algorithms, and it is difficult to find appropriate real temporal datasets for these experiments. There are two ways to generate experimental data: drawing artificial trajectories and generating random trajectories. The artificial data can be used to validate the special experimental cases, but the drawing function of the GUI simulation is difficult to produce large data sets. The random

data can generate large data sets, but it is hard to validate special cases. Hence, both methods are required. Secondly, as the temporal and spatial data are recorded by locations, they should be visualized in the experiments in order to observe the experiment results. Hence, a GUI simulation tool needs to be developed, and it should be able to be implemented for all experiments.

The GUI simulation tool is designed with five functions: generate spatial datasets since the experiments need many example data sets; store the trajectories to a text file since the trajectory data can be used for later experiments, as well as be compared with other data sets; load a trajectory text file since the experiments are needed to compare different group results; draw particular trajectories for testing since the experiments need to test many special cases; and visualize the spatial data in a GUI since the transformation of grouped trajectories needs to be observed.

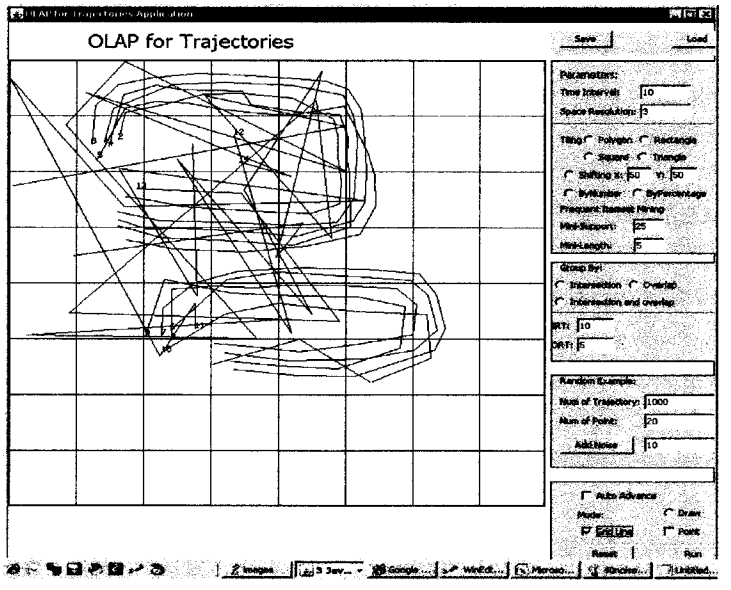

Figure 5.1: The Simulation Tool (Java Applet Version)

The simulation tool provides both a Java version (See Figure 5.1 on page 34) and a Web version (See Figure 5.2 on page 35). The Java language is chosen because it can easily convert Java to Java Applet (Web) Edition and it supports different operating systems, although it runs slower than C or C++. In the Java version, all parameters can be manually entered, hence, the GUI simulation tool is convenient for adjusting

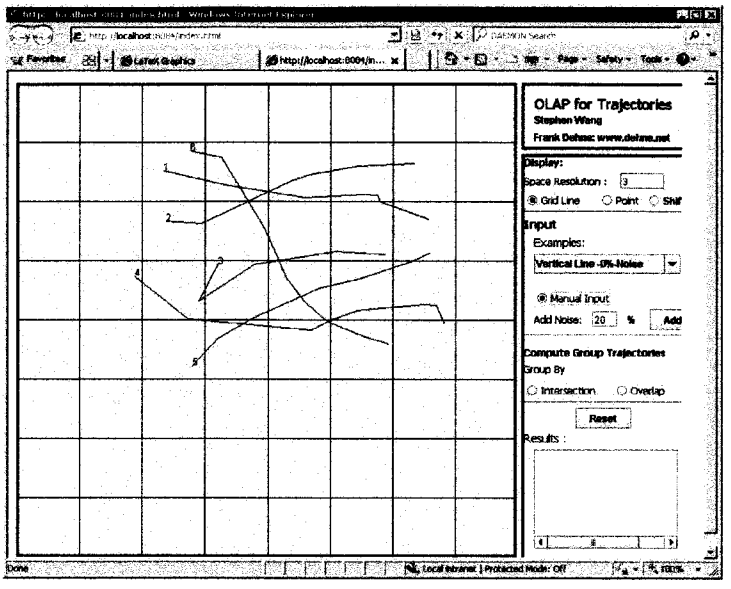

Figure 5.2: The Simulation Tool (Web Version)

parameters. The Web version cannot accept manually entered parameters, but it allows users to run the program without downloading the code. This simulation tool can be used not only for this research, but also for other spatial and temporal research. Therefore, the GUI simulation tool is a very useful tool for trajectory experiments.

5.3 The Testing Methodology

5.3.1 Tests without noise

The testing methodology of this research includes two methods: tests with and without noise. Each testing method includes four special types of trajectories: diagonal, horizontal, spiral, and vertical trajectories. This section focuses on tests without noise. In tests without noise, all trajectories are considered to be useful trajectories. Four special types of trajectories are applied to both Group By Intersection and Group By Overlap.

The purpose of tests without noise is to verify whether the Shifting Grid Algorithm improves the results of computing groups of trajectories. The artificial data in tests without noise is generated by the drawing function of the GUI simulation tool. The experimental results of each special type of trajectories are analyzed and compared

with other special types of trajectories. The detailed experimental results are given in Chapter 6.

5.3.2 Tests with noise

In tests with noise, the noise trajectories are added into the "useful" trajectory data. The noise trajectory data is generated randomly by the random example function of the GUI simulation tool. In the noise trajectory data, the ID of first noise trajectory is generated by the ID of the last trajectory in original datasets plus 1, and other noise trajectories are automatically increased by 1; the location values x and y of each trajectory are generated by the Java Random function; time is generated by automatically increasing 10 (start from 0); the number of noise trajectories are based on the noise percentage and the total number of the "useful" trajectories. For example, noise percentage is 10%, and the total number of original trajectories are 20, and then 2 noise trajectories should be generated. The noise trajectory data sets are added to four special types of trajectories. The trajectory data with noise is used for both Group By Intersection and Group By Overlap. The purpose of the cases with noise is to ascertain how the new Shifting Grid Algorithm handles noise trajectory data. The noise trajectory data sets are added from lower (10%) percentages to higher (40%) percentages. The detailed experimental results are presented in Chapter 6.

5.4 Summary

In summary, the experimental environment is described, including the configuration of hardware and software, a new GUI simulation tool, as well as a discussion of the testing methodology containing tests with and without noise. The next chapter presents all experimental results, as well as those compared with the current and previous research results.

Chapter 6

Experimental Results

6.1 Introduction

This chapter contains five sections. Section 6.2 analyzes the experimental results, including both the Group By Intersection and Group By Overlap methods. Section 6.3 discusses the parameters of anti-noise, including the min-length and min-support, and other parameters. Section 6.4 compares the experimental results with previous research results; and Section 6.5 gives a summary of this chapter.

6.2 Analysis of Results

6.2.1 Group by Overlap

In the Group by Overlap method, the experimental results have the following limitations:

1. All trajectories without noise are intended for sequences of movements, since the Group By Overlap algorithm is aimed for sequences of movements. These trajectories are generated by the drawing function of the GUI simulation tool. All trajectories are assumed to occur in the same time period.

2. All noise is generated by the random example function of the GUI simulation tool.

3. All trajectory datasets are small, since the experiment computer cannot process large data sets.

4. The values of the parameters (shown in Table 6.1 on page 38) are fixed in the Auto Parameters Algorithm.

Table 6.1: Parameters in Auto Parameters Algorithm for Group By Overlap

Parameters / Min-Max	r_{space}	s	min-length	shifting%	ORT	GC
Minimum	3	10%	2	20%	30%	4
Maximum	6	30%	4	60%	30%	4

The parameters are for Group By Overlap (Table 6.1 on page 38):
The following items describe the parameters in Table 6.1 on page 38:

1. r_{space} is the level of space resolution, for example, if $r_{space} = 2$, then the coordinate is divided into $2^2 \times 2^2$.

2. Min-support and min-length is used for the data mining process.

3. The shifting percentage is the percentage of shifting distance.

4. ORT is the Overlap Ratio Threshold.

5. GC is the Group Cardinality.

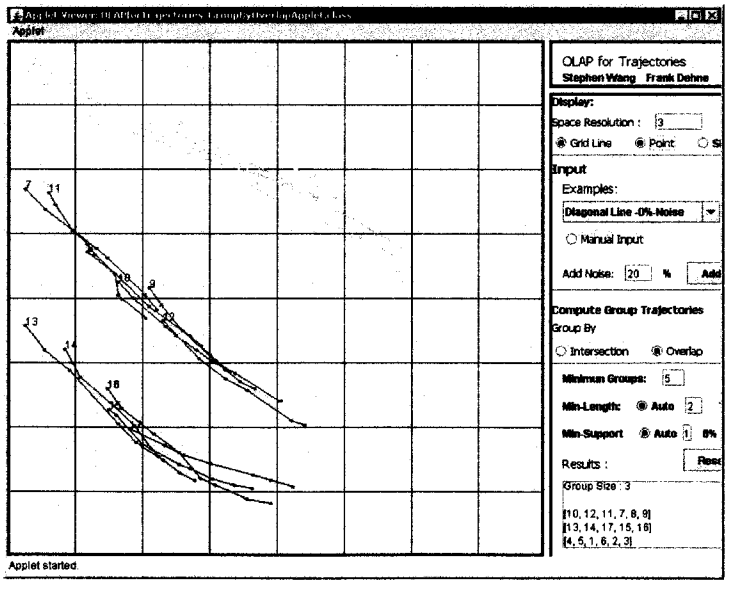

Figure 6.1: Group By Overlap Diagonal trajectories without noise (based on Table 6.1)

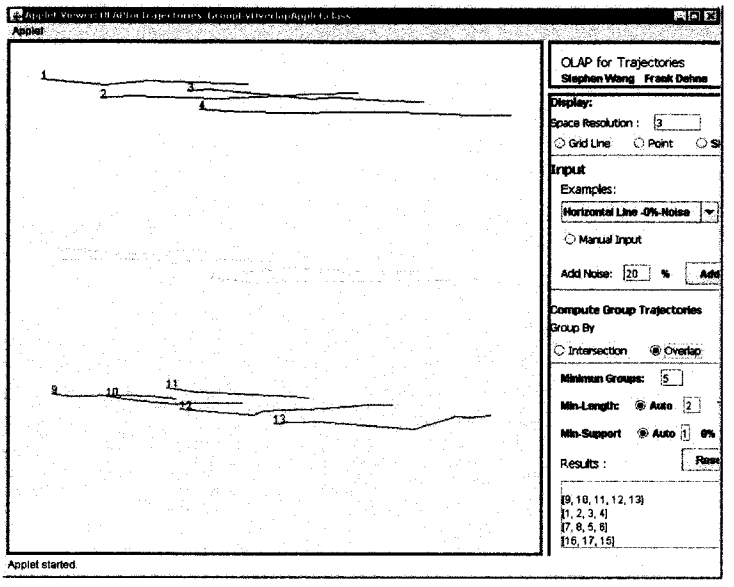

Figure 6.2: Group By Overlap Horizontal trajectories without noise (based on Table 6.1)

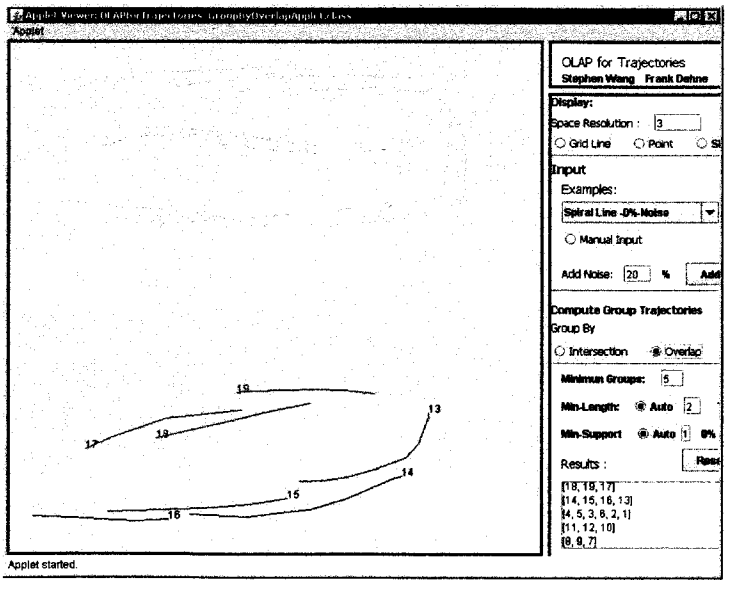

Figure 6.3: Group By Overlap Spiral trajectories without noise (based on Table 6.1)

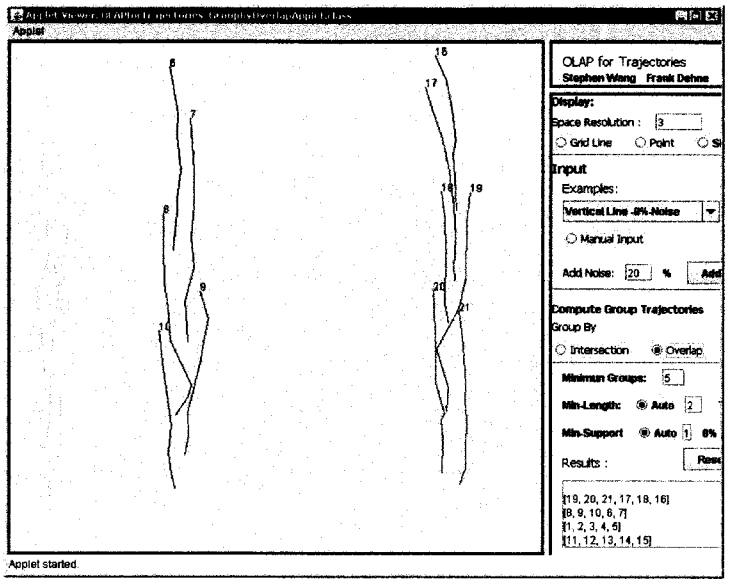

Figure 6.4: Group By Overlap Vertical trajectories without noise (based on Table 6.1)

The experimental results of tests without noise are shown as screenshots in Figure 6.1 on page 38, Figure 6.2 on page 39, Figure 6.3 on page 39 and Figure 6.4 on page 40. In tests of data without noise, all cases show that all trajectories are captured by the Auto Parameters and Shifting Grid Algorithms, as well as demonstrate that both algorithms work for Group By Overlap in theory and in practice. For example, in Figure 6.1 on page 38, there are total 17 trajectories, the result is shown in the lower right corner: group 1 (7, 8, 9, 11, 12), group 2 (13, 14, 15, 16, 17), and group 3 (1, 2, 3, 4, 5, 6). The result is also shown by different color trajectories in the picture. Each group shows one distinguished color, and noise trajectory is represented by black. It clearly shows that all trajectories are captured by the proposed algorithms.

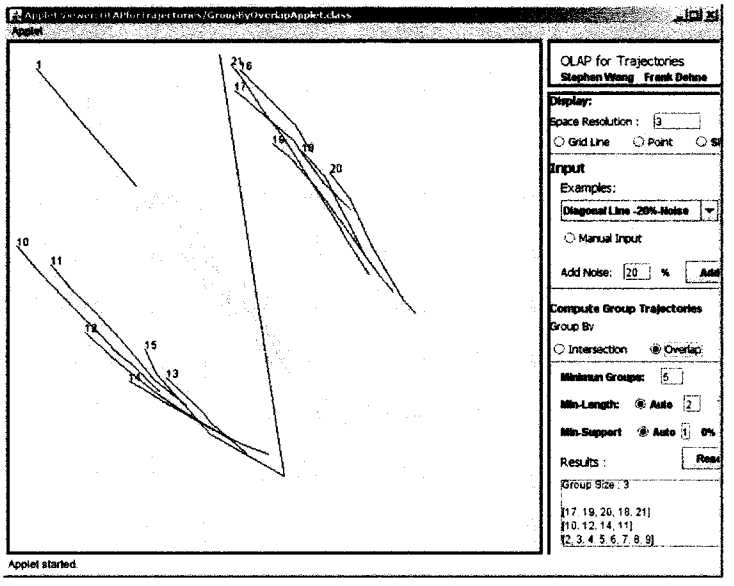

Figure 6.5: Group By Overlap Diagonal trajectories with 10% noise (based on Table 6.1)

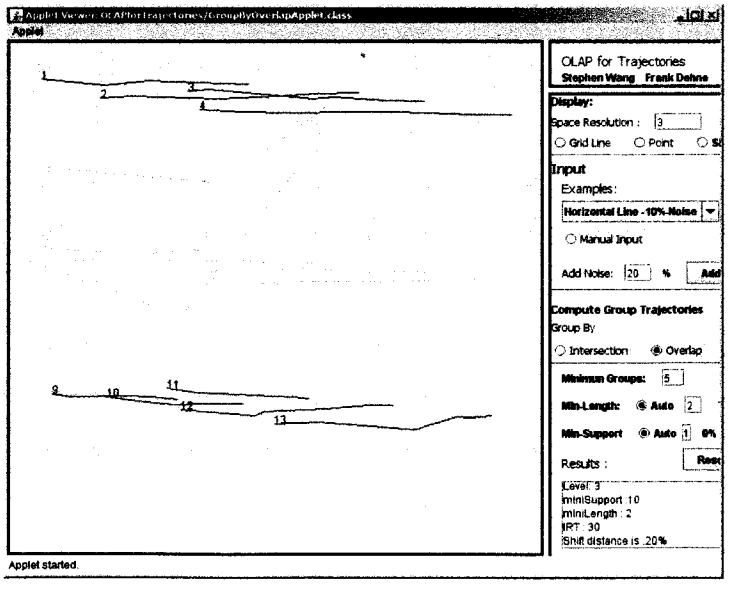

Figure 6.6: Group By Overlap Horizontal trajectories with 10% noise (based on Table 6.1)

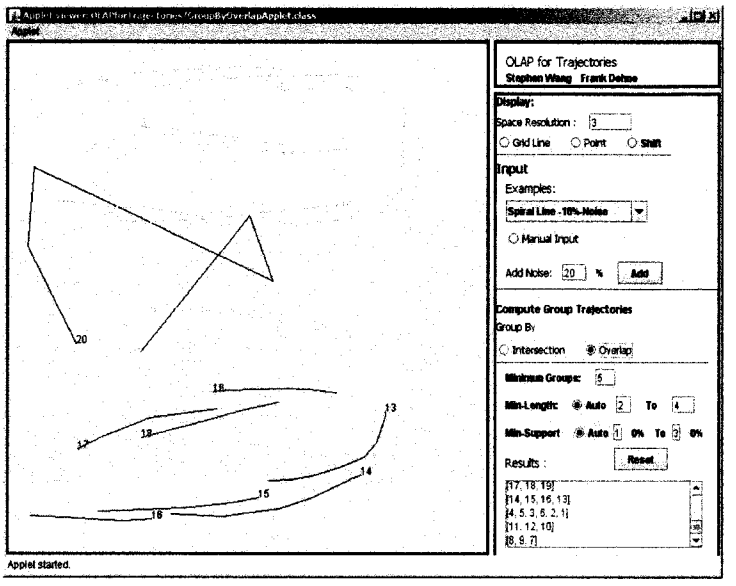

Figure 6.7: Group By Overlap Spiral trajectories with 10% noise (based on Table 6.1)

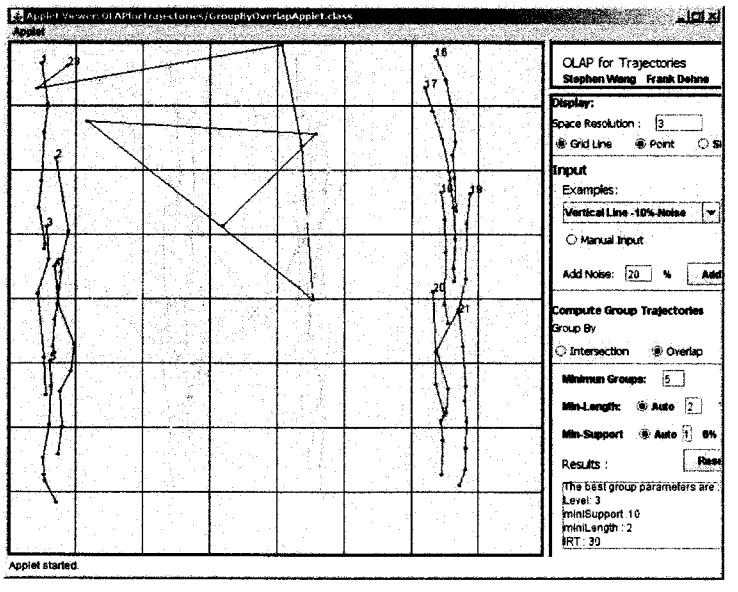

Figure 6.8: Group By Overlap Vertical trajectories with 10% noise (based on Table 6.1)

The results of Table 6.2 on page 43 are based on the parameters in Table 6.1 on page 38. In Table 6.2, "100%" means that all "useful" trajectories are captured by

Table 6.2: Four special types of trajectories with and without noise (based on Table 6.1) for Group By Overlap

noise / cases	0%	10%	20%	30%	40%
Diagonal line	100%	around 80%	around 80%	less 80%	around 80%
Horizontal line	100%	over 90%	around 90%	less 90%	around 80%
Vertical line	100%	over 90%	around 90%	less 90%	around 80%
Spiral line	100%	over 90%	around 90%	less 90%	around 80%

the Shifting Grid Algorithm and the Auto Parameters Algorithm. "over 90%" means that over 90% "useful" trajectories are captured by the two new algorithms. Noise "10%" means that 10% noise trajectories are added into the original data without noise. In tests with noise, some noise cannot be filtered out, and is considered to be useful trajectories. For example, in Figure 6.8 on page 42, some noise is grouped into yellow trajectory group. In Figure 6.5 on page 41, some "useful" trajectories (trajectory 1, 13, 15, 16) are not grouped into any groups, these trajectories are shown as black. There are multiple reasons: 1. the minimum length and minimum support are too small (the values of the parameters are fixed values from minimum to maximum) for the small data sets. The trajectories can be easily satisfied with the requirements; 2. the Auto Parameters Algorithm cannot automatically adjust the range of these parameters. The noise data is filtered out if the proper parameters such as minimum length, minimum support and *ORT* are chosen. For instance, Section 6.3.1 demonstrates the robustness of anti-noise. The example shows the relationship between the robustness of anti-noise and parameters, in particular, the parameters can resist over 80% noise data when the parameter values are changed to higher values. The experimental results are not perfect in cases with noise, but the two algorithms are still able to identify most useful trajectories (90% trajectories) when the noise data for all special cases is from 10% to 40%. The experimental results with 10% noise are shown as: diagonal case (Figure 6.5 on page 41), horizontal case (Figure 6.6 on page 41), spiral case (Figure 6.7 on page 42), and vertical case (Figure 6.8 on page 42). In terms of the experiment results, diagonal cases are the most difficult ones to handle, since the experiment's results are the worst in the four special cases.

6.2.2 Group By Intersection

In the Group By Intersection section, the experimental results are based on the following conditions:

1. All trajectories are intended for parallel movements, since the Group By Intersection algorithm is aimed for parallel movements. These trajectories are generated by the drawing function of the GUI simulation tool. All trajectories are assumed to occur in the same time period.

2. All noise data is generated by the random example function of the GUI simulation tool.

3. All trajectory data sets are small, since the experiment computer cannot process large data sets.

4. The values of the parameters (See Table 6.3 on page 44, and Table 6.5 on page 47) are fixed by the Auto Parameters Algorithm.

Table 6.3: Group 1: Parameters in Auto Parameters Algorithm for Group By Intersection

Parameters / Min-max	r_{space}	min-support	min-length	shifting %	ORT	GC
Minimum	3	10%	4	20%	30%	5
Maximum	6	30%	5	60%	30%	5

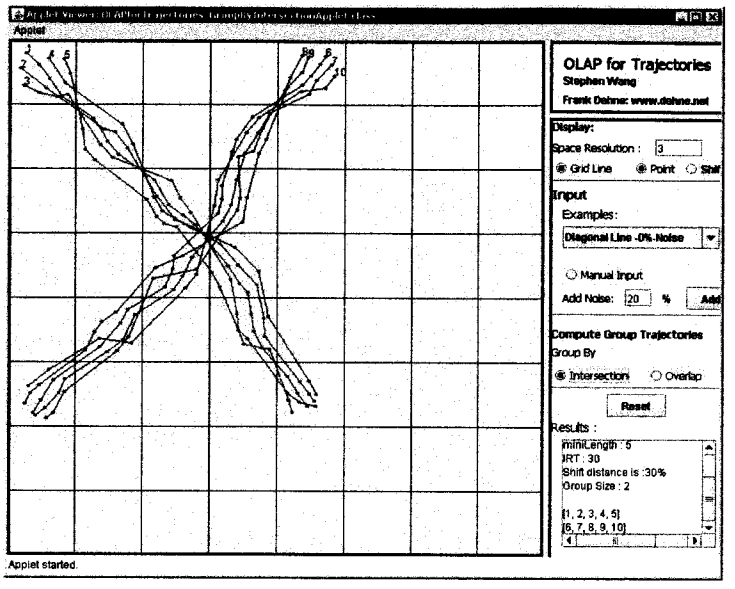

Figure 6.9: Group By Intersection Diagonal trajectories without noise (based on Table 6.3)

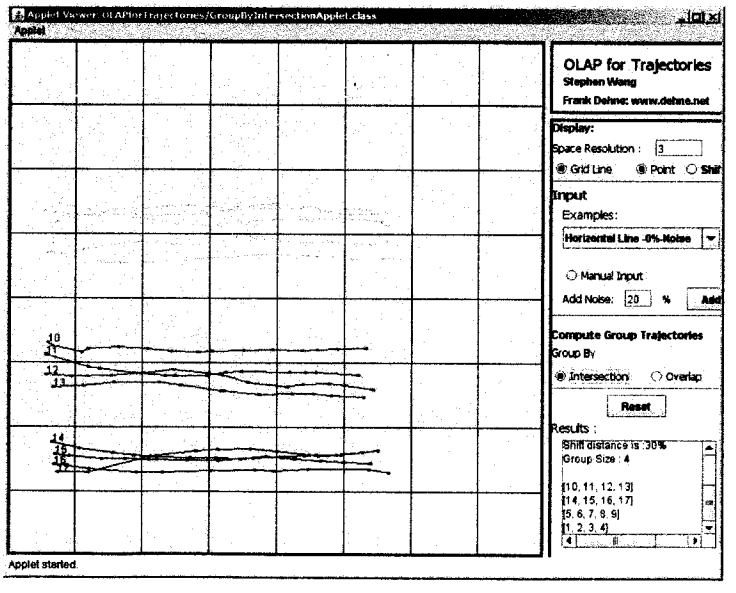

Figure 6.10: Group By Intersection Horizontal trajectories without noise (based on Table 6.3)

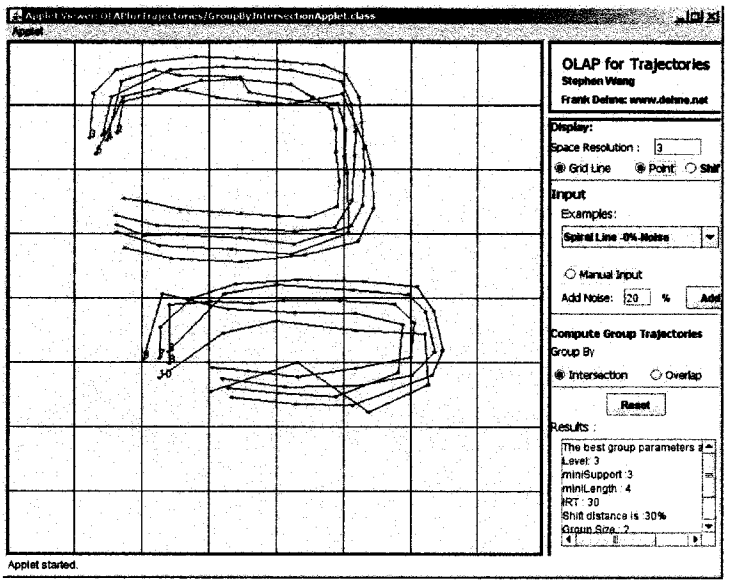

Figure 6.11: Group By Intersection Spiral trajectories without noise (based on Table 6.3)

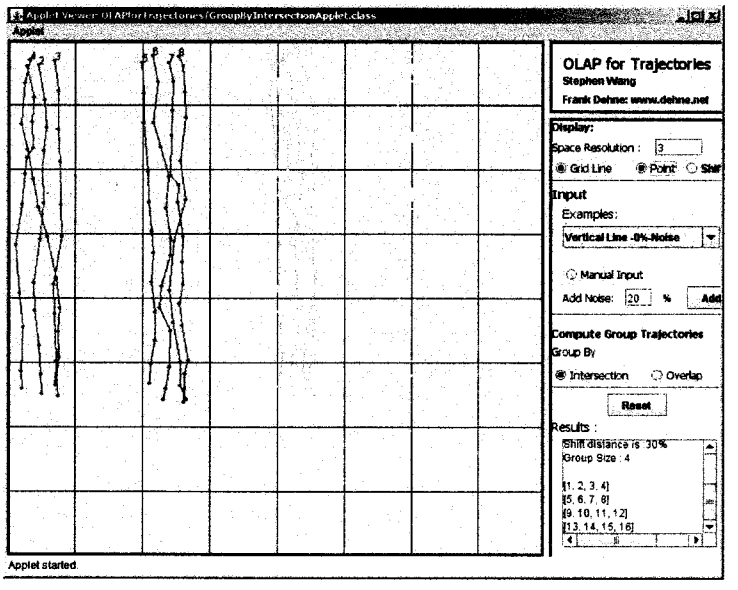

Figure 6.12: Group By Intersection Vertical trajectories without noise (based on Table 6.3)

Based on the first group parameters of Table 6.3, all tests without noise have returned good results (Figure 6.9 on page 45, Figure 6.10 on page 45, Figure 6.11 on

Table 6.4: Four special types of trajectories with and without noise based on Table 6.3 for Group By Intersection

Diagonal line	100%	over 90%	around 90%	less 90%	around 80%
Vertical line	100%	over 90%	around 90%	less 90%	around 80%

page 46, and Figure 6.12 on page 46). However, once trajectories contain noise, the results are not perfect (See Table 6.4 on page 47).

Table 6.5: Group 2: Parameters in Auto Parameters Algorithm for Group By Intersection

Parameters / Min-max	r_{space}	min-support	min-length	shifting	ORT	GC
Minimum	3	20%	5	20%	40%	5
Maximum	6	40%	6	60%	40%	5

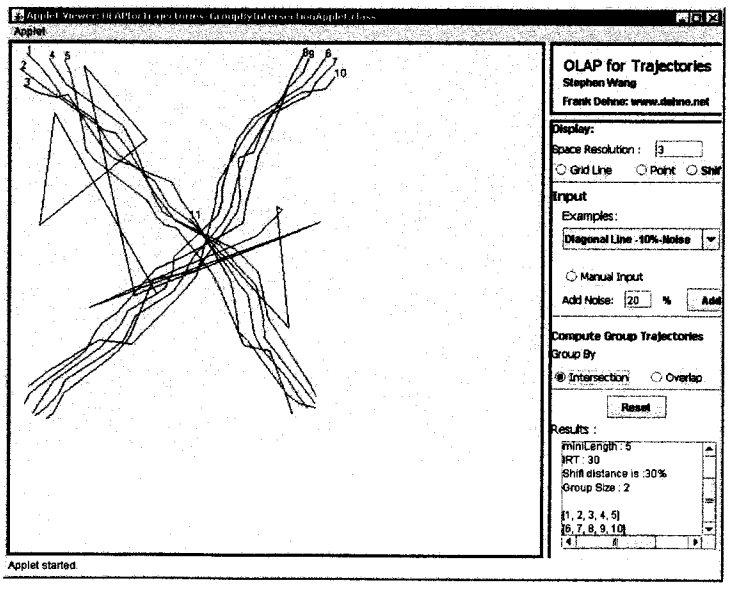

Figure 6.13: Group By Intersection Diagonal trajectories with 10% noise (based on Table 6.5)

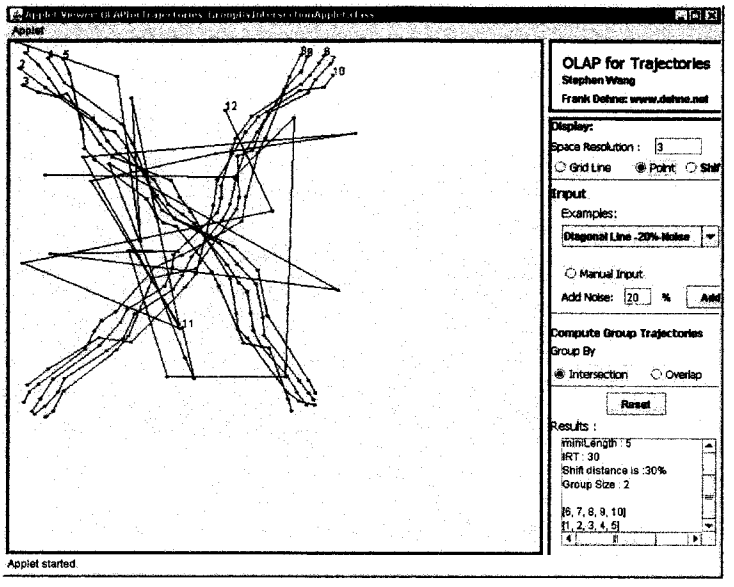

Figure 6.14: Group By Intersection Diagonal trajectories with 20% noise (based on Table 6.5)

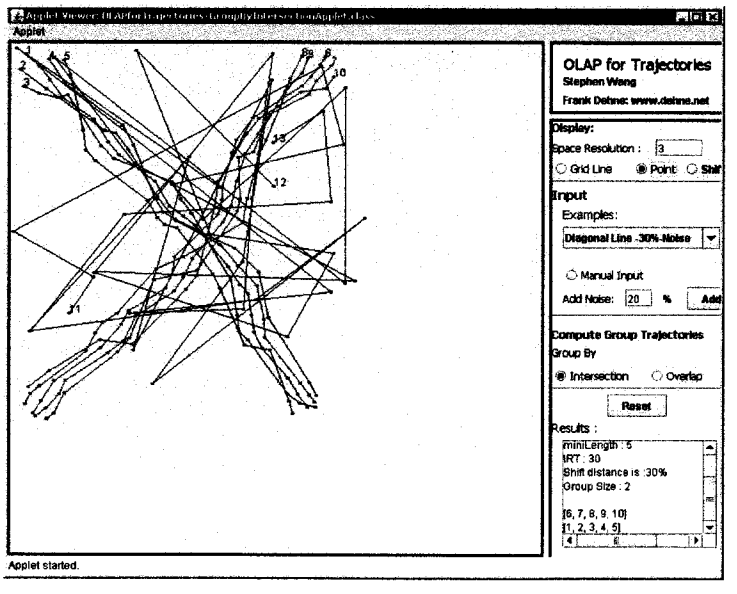

Figure 6.15: Group By Intersection Diagonal trajectories with 30% noise (based on Table 6.5)

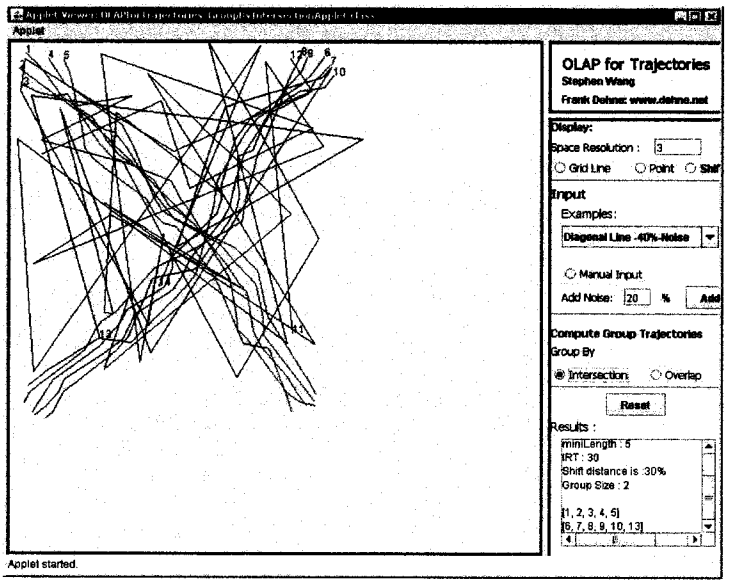

Figure 6.16: Group By Intersection Diagonal trajectories with 40% noise (based on Table 6.5)

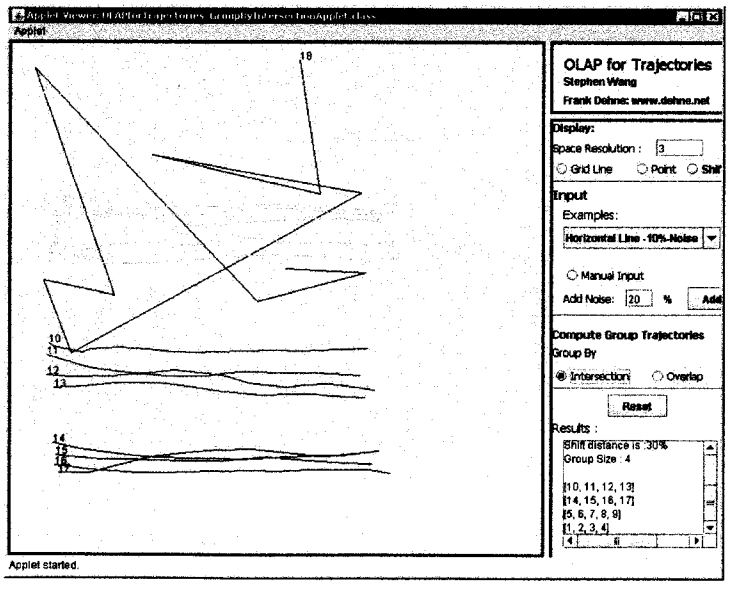

Figure 6.17: Group By Intersection Horizontal trajectories with 10% noise (based on Table 6.5)

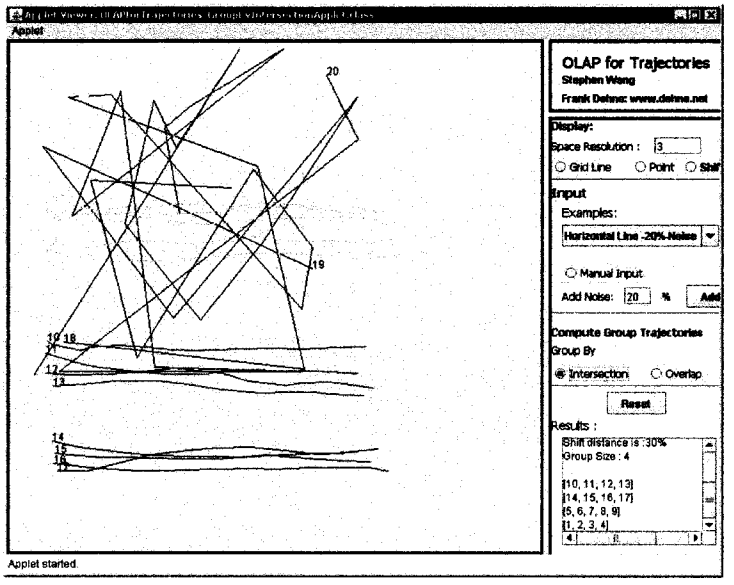

Figure 6.18: Group By Intersection Horizontal trajectories with 20% noise (based on Table 6.5)

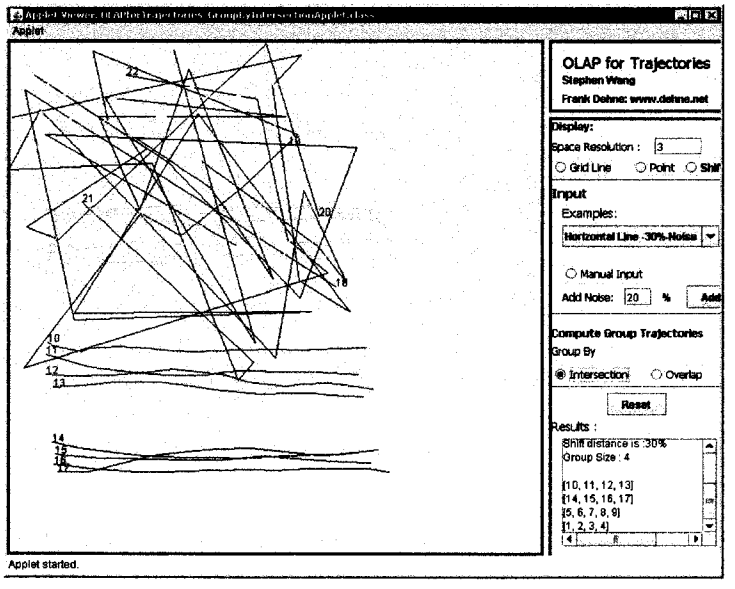

Figure 6.19: Group By Intersection Horizontal trajectories with 30% noise (based on Table 6.5)

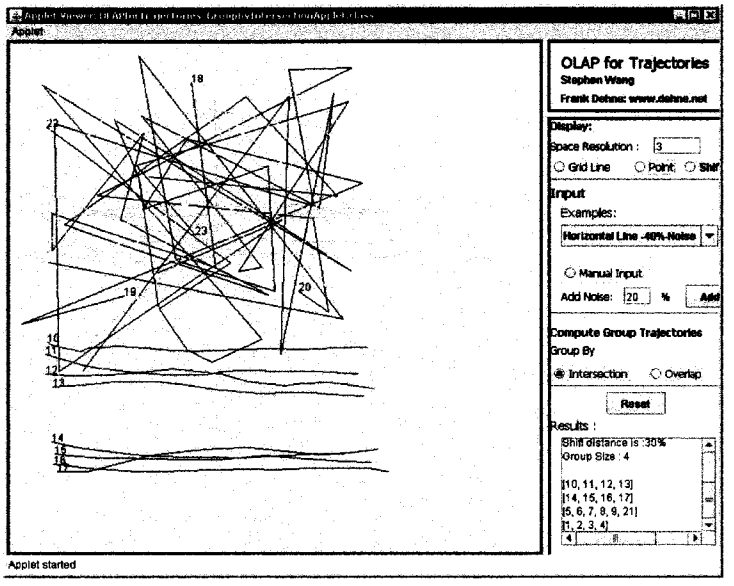

Figure 6.20: Group By Intersection Horizontal trajectories with 40% noise (based on Table 6.5)

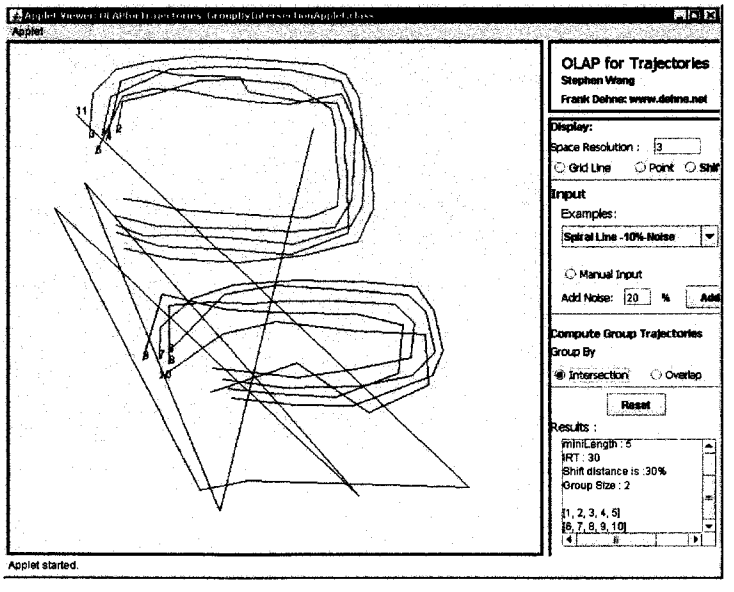

Figure 6.21: Group By Intersection Spiral trajectories with 10% noise (based on Table 6.5)

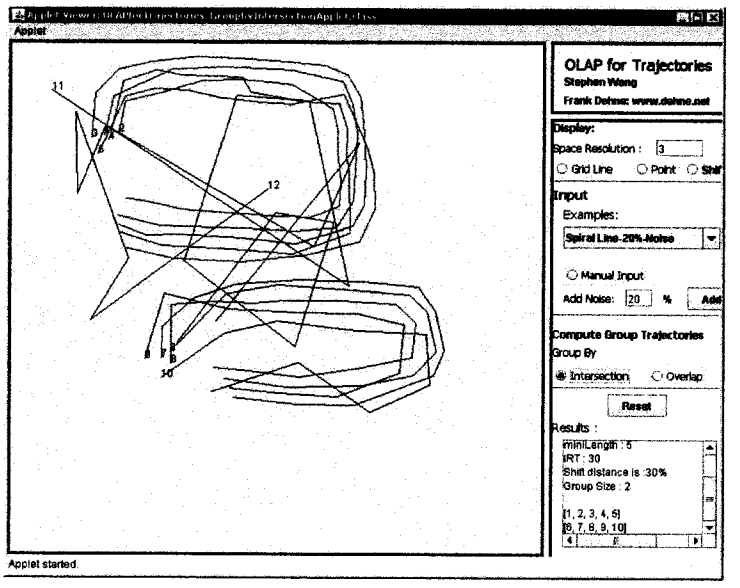

Figure 6.22: Group By Intersection Spiral trajectories with 20% noise (based on Table 6.5)

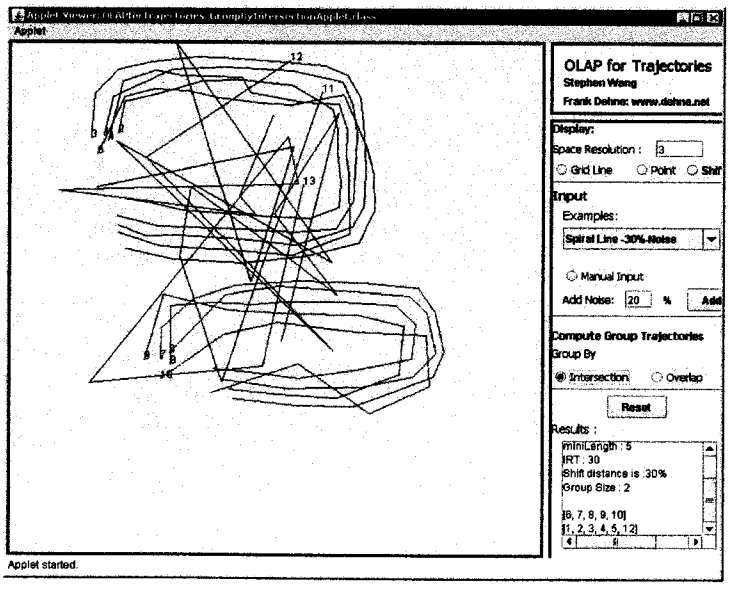

Figure 6.23: Group By Intersection Spiral trajectories with 30% noise (based on Table 6.5)

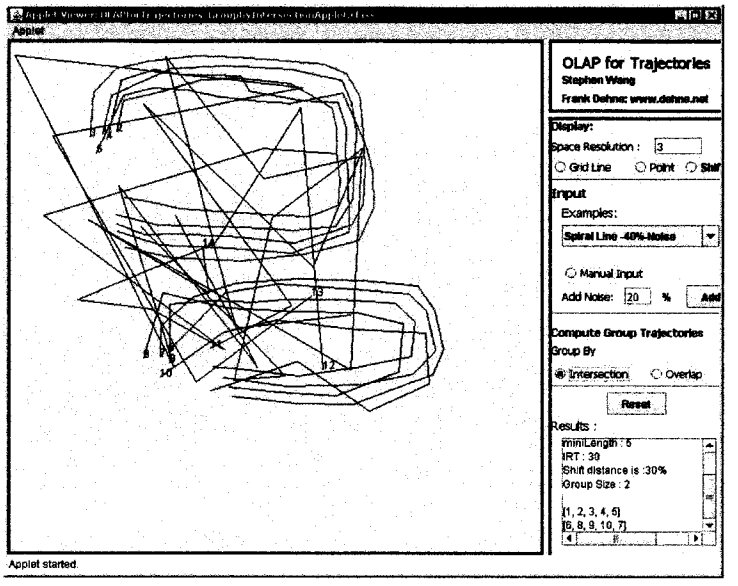

Figure 6.24: Group By Intersection Spiral trajectories with 40% noise (based on Table 6.5)

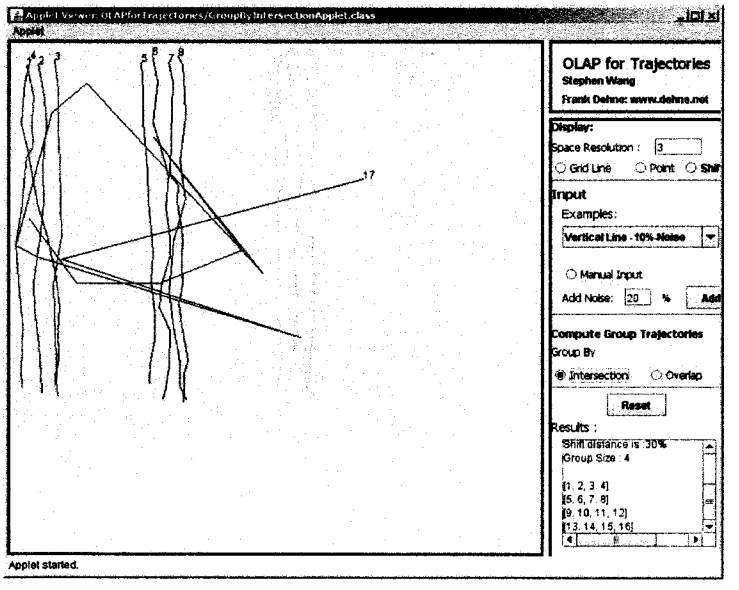

Figure 6.25: Group By Intersection Vertical trajectories with 10% noise (based on Table 6.5)

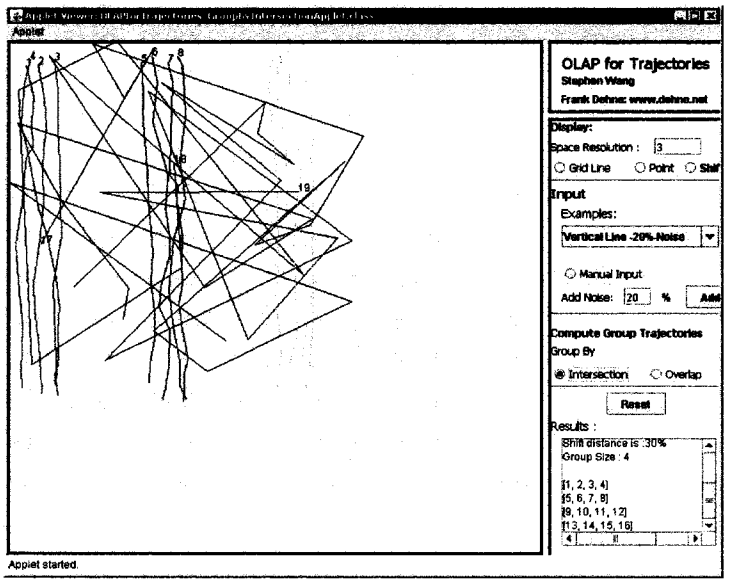

Figure 6.26: Group By Intersection Vertical trajectories with 20% noise (based on Table 6.5)

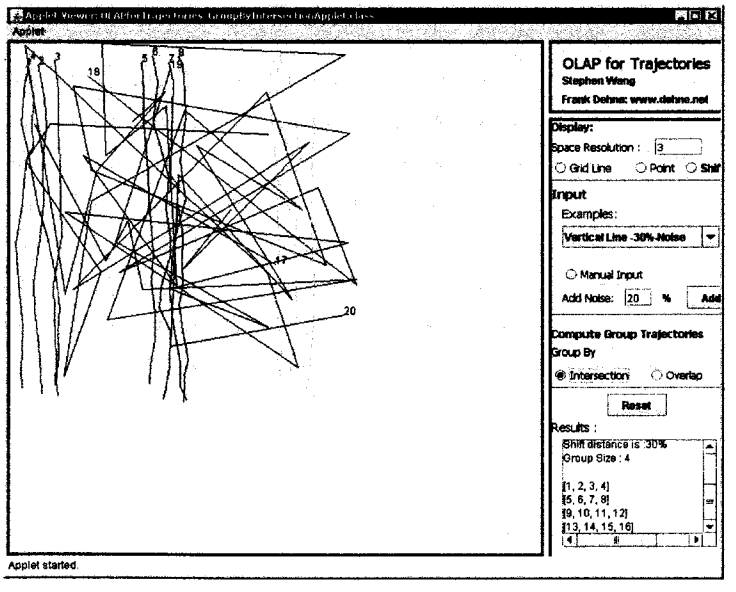

Figure 6.27: Group By Intersection Vertical trajectories with 30% noise (based on Table 6.5)

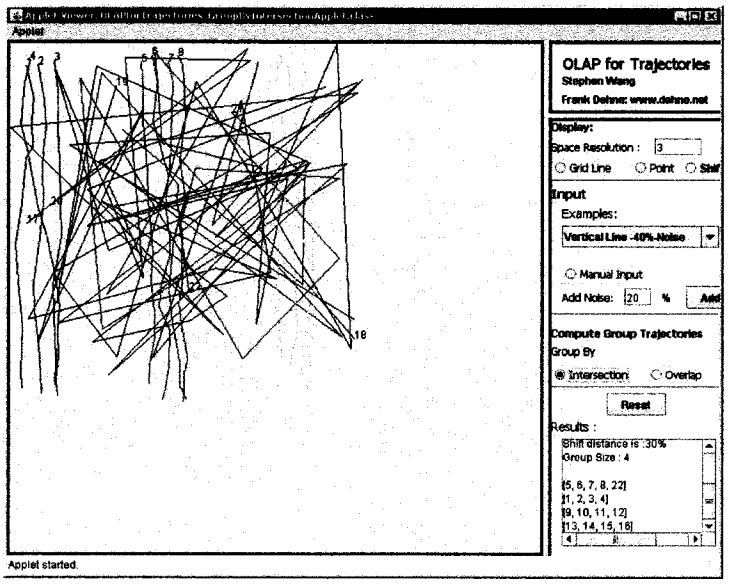

Figure 6.28: Group By Intersection Vertical trajectories with 40% noise (based on Table 6.5)

Table 6.6: Four special types of trajectories with and without noise based on Table 6.5 for Group By Intersection

Diagonal line	100%	100%	100%	100%	around 90%
Vertical line	100%	100%	100%	100%	around 90%

There are two group experiment results for Group By Intersection: one is based on the parameters shown in Table 6.3, another is based on the parameters shown in Table 6.5. The two groups have the same results when four special types of trajectories are made up of data without noise (see Figure 6.9 on page 45, Figure 6.10 on page 45, Figure 6.11 on page 46, and Figure 6.12 on page 46). The parameters in Table 6.5 are more resistant to noise since the min-support and min-length are larger values. The parameters in Table 6.3 cannot even handle 10% (See Table 6.4 on page 47), but the parameters in Table 6.5 can handle up to over 30% noise data. The experimental results for Table 6.5 are demonstrated in diagonal case (Figure 6.13 on page 47, 6.14

on page 48, 6.15 on page 48, and 6.16 on page 49), horizontal case (Figure 6.17 on page 49, 6.18 on page 50, 6.19 on page 50, and 6.20 on page 51), spiral case (Figure 6.21 on page 51, 6.22 on page 52, 6.23 on page 52, and 6.24 on page 53), and vertical case (Figure 6.25 on page 53, 6.26 on page 54, 6.27 on page 54, and 6.28 on page 55). The parameters in Table 6.5 can handle noise up to 30%, when noise reaches 40%, then some noise is considered to be useful trajectories. The Auto Parameters Algorithm works well, only when the range of the parameters is chosen properly. Section 6.3.1 discusses these parameters. The different special trajectories of anti-noise may be different. For example, diagonal case can handle noise over 40% even when the values of the parameters are the same.

Two problems are encountered when large random examples are implemented. Firstly, if the range of parameter values is too small, the experiment computer cannot finish running the examples because of a lack of memory and disk space. Secondly, if the range of parameter values is very wide, then the Auto Parameters Algorithm report empty group results.

In summary, the experiment results are expected. The purpose of the two algorithms is achieved. All results are reported by Auto Parameters Algorithm; and the final group results are improved, since some trajectories cannot be captured in previous research results but this research is able to capture them.

6.3 Parameter Space

6.3.1 Robustness of Anti-noise

The experiment results show that the robustness of anti-noise is related to min-length and min-support, but the robustness of anti-noise depends on three parameters: min-support, min-length, and IRT or ORT. The basic relationship between the values of parameters and robustness of anti-noise is that larger values of the parameters have stronger anti-noise. The following example shows that the algorithm can resist over 80% noise. The range of minimum length is from 5 to 8, and the range of minimum support is from 50% to 70%. The values of the minimum length and minimum support are larger than the previous experiments. All useful trajectories are captured, even

with the noise data over 80% (See Figure 6.29 on page 57).

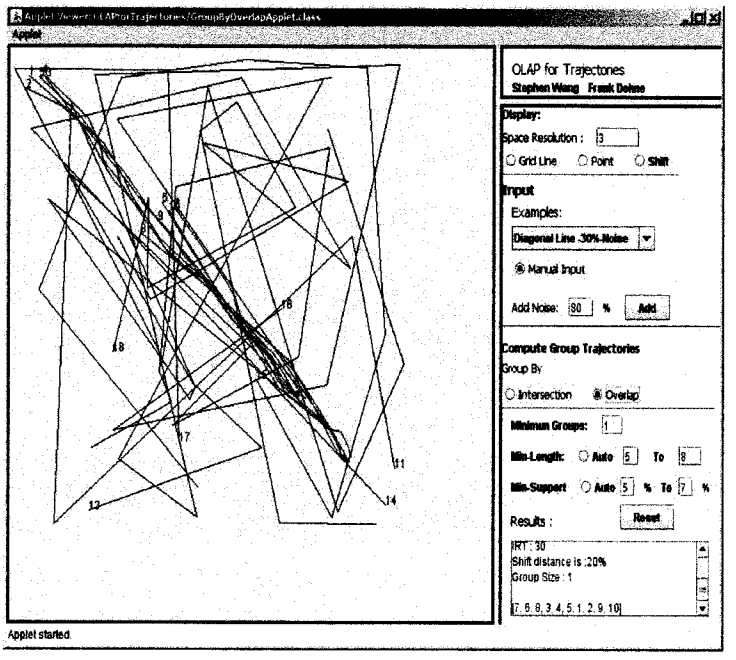

Figure 6.29: Trajectory Data With 80% noise

In the real world, large data sets are used, as well as long paths of each moving object. Noise data is filtered by larger values of min-length and min-support. The experiments use the same ORT or IRT, but it also resists noise data with larger values.

In most situations of analyzing trajectories, the noise filter is a big challenge. The Shifting Grid Algorithm is reliable and robust for anti-noise since there are two levels of filters. The first level is in the data mining program. This level depends on two parameters: min-length and min-support. The second level is determined by the parameter of group method algorithms: Intersection Ratio Threshold (Group by Intersection) or Overlap Ratio Threshold (Group By Overlap). The two levels of filters are very powerful, as a result, most noise is eliminated.

6.3.2 Other parameters

The previous section discusses the minimum length and minimum support used for anti-noise. This section only concentrated on three parameters: GC, shifting percentage and r_{space}. The importance of GC is discussed in Section 4.3.1. The final group results are determined by GC. The shifting percentage is described in the Shifting Grid Algorithm section. The experiment results indicate that the shifting percentages can influence final results. Shifting percentages can determine which trajectories can be considered useful. The space resolution r_{space} also determines the number of groups. For example, a very few number of groups may be obtained if the r_{space} is very small such as 1 (there are only four grid boxes if r_{space} is 1.).

6.4 Comparison of Performance against Other Techniques

The Shifting Grid Algorithm is better at capturing the trajectories. The experiment results show that both Group By Intersection and Group By Overlap results are improved since the Shifting Grid Algorithm and Auto Parameters Algorithm can obtain more accurate and reasonable results. The trajectories are not in the same grid as the original data sets, but the new algorithm is still able to identify these trajectories. The experiment results are much better than those represented in previous research work [1]. The resulting groups are more reasonable than those reported in previous studies.

The Auto Parameters Algorithm simplifies the usage of Group-By operator since it can automatically search parameters to provide a better result. Users do not need to provide multiple parameters when the Group-By operator is used. Previous research work requires entering multiple parameters, and the resulting groups come out without comparison, even the results are not reasonable. In this research, the dynamic parameters are provided in a certain range. The Auto Parameters Algorithm automatically located the better parameters. This is an additional feature for Group-By operator compared with the previous studies.

The inadequacy of the Shifting Grid and Auto Parameters Algorithms is a time-consuming problem since some processes need to run multiple times. The two algorithms require much more running time than previous research did when they compute the same SQL query, but a better result is returned.

6.5 Summary

In summary, this chapter presents the experiment results for both Group By Overlap and Group By Intersection, discusses parameters, and compares the results of this research with previous research. In tests without noise, all experiment results are very good. In tests with noise, the appropriate range of parameters produces a better result. The experiment results demonstrate that both Shifting Grid Algorithm and Auto Parameters Algorithm work in theory and practice. The result of Group-By operator is improved by both algorithms since they can identify more trajectories than previous research. Moreover, the Auto Parameters Algorithm tries to optimize all parameters, and returns a better result. The goal of this research is achieved based on the experiment results.

Chapter 7

Conclusions and Future Work

7.1 Introduction

This chapter includes four sections. Section 7.2 summaries the research results. Section 7.3 outlines the conclusions of this research. Section 7.4 discusses future work in this research area.

7.2 Summary of Results

This section outlines the results of this research. Firstly, the Shifting Gird Algorithm and Auto Parameters Algorithm are implemented for both Group By Intersection and Group By Overlap. The experiments are implemented based on four special cases since the four cases cover most situations of trajectories for moving objects.

Secondly, in terms of the experiment results, both the Shifting Grid Algorithm and Auto Parameters Algorithm work very well with the trajectories without noise for both Group By Overlap and Group By Intersection. In cases of trajectories with noise, if the parameters of Auto Parameters Algorithm are properly chosen, then the two algorithms also perform well. The experimental results show that the final group results are much better than those of Baltzer et al for both Group By Intersection and Group By Overlap. The objective of the Shifting Algorithm is attained since the group results are improved based on the experiment results. Similarly, the objective of the Auto Parameters Algorithm is also achieved since a better result is reported with better parameters.

Thirdly, the GUI simulation tool can be used for all experimental cases. The simulation tool is very convenient for the experimental tests. The java version can be used for programming to debug parameters and the Java Applet (Web) version can be used for the purpose of the demonstration.

Finally, all experiment results are small group sets because it is difficult to find large data sets from the real world, and the normal experiment personal computer cannot handle this time-consuming problem. However, the ideas and concepts of this research are demonstrated by the experiment results.

7.3 Conclusions

In conclusion, the Shifting Algorithm is proposed for identifying more useful and accurate trajectories, and producing a better result. The objective of this algorithm is attained based on the experiment results, although the experiments do not run large data sets from the real world. The Auto Parameters Algorithm shows the expected values in terms of the experiment results since the better result is automatically computed. The experiment results demonstrate the ideas of the two algorithms.

There are still two remaining problems. The first problem is time-consuming. Experiment performance is hindered because both algorithms require substantial computer resources. The second problem is that the parameters of Auto Parameters Algorithms still need to assign the proper value ranges, and the ranges of values influence the final results based on the experiment results. If the parameters are not chosen properly, then users may not get the result they expect. Many parameters are introduced in this research, therefore, it is difficult to make all parameters optimal.

7.4 Future Work

In the future, much work can be done in this area of research. Firstly, the experiments in this research did not run large data sets from the real world. The reason is: the Auto Parameters Algorithm and the Shifting Algorithm include a large number of computing problems; the memory and disk space required is very large if the large trajectory data sets from the real world are implemented; the experiments can take a lot of time. In order to complete this work, all algorithms need to be parallelized. The challenge of parallelizing these algorithms is that if it is done improperly, the data may compromise. The merge procedure needs to avoid some trajectories being eliminated by the pruning process. The parallelized algorithms need to run on a

high performance computer or other grid computing machines. Another challenge of running large data sets is to find someone who would be willing to provide large data sets for the research experiment since companies may not want to share their data. Hence, in order to run large dataset experiments, parallelizing the algorithms and finding large data sets from real world become challenging.

Secondly, this research only concentrates on one tiling (grid)-box. The different tilings such as circle, polygon, triangle, and star can be used for trajectories research problems. The challenge of future research work is to determine how to generate these different shapes and how to compare their results. Different results could be obtained because the raw trajectories may map into different tilings. If a different tiling is used, then results of each tiling can be compared, and can be determined the shape that is the best for the two algorithms.

Thirdly, these algorithms of this research have not been implemented in a real data warehousing environment. A possible research option is to implement these algorithms on an open source database product such as MySQL and PostgreSQL. The challenge of the research work is to understand the API interface of one open database product. Another challenge for the future work is to improve the performance of algorithms. The real performance can be demonstrated after they are running on a real database product.

Appendix A

Algorithms

A.1 Algorithm 1: High-level Trajectory Aggregation Framework[1]

A.2 Algorithm 2: Group By Overlap Algorithm [1]

A.3 Algorithm 3: Group By Intersection Algorithm [1]

High-level Trajectory Aggregation Framework

Input:

1. set T of trajectories,
2. space resolution r_{space},
3. time resolution t_{time},
4. minimum support s,
5. minimum length l

`Output: set of Groups G`

1. map $\mathcal{T}$ to resolution (r_{space}, r_{time}) resulting in T'
2. computing frequent itemsets $\mathcal{F}$ in T' with s and l

Reverse matching

3. $\mathcal{C} \longleftarrow \emptyset$
4. for all $f \epsilon \mathcal{F}$ do
5. determine the subset c of $\mathcal{T}$ that corresponds to f
6. $\mathcal{C} \longleftarrow \mathcal{C} \bigcup \{(f, c)\}$
7. end for

Group merging

8. select one of group methods

(a) Group By Overlap (A.2 in Appendix)

(b) Group By Intersection (A.3 in Appendix)

(c) Group by Overlap and Intersection(Algorithm 4)

11. return the resulting set of groups $\mathcal{G}$

Figure A.1: Algorithm: High-level trajectory aggregation framework

Group By Overlap

Input:

1. set T of trajectories,
2. set $\mathcal{C}$ determined in line 3-7 of A.1 in Appendix
3. Overlap Ratio Threshold ORT

`Output: set of Groups G`

Build overlap Graph $\Gamma = V_\Gamma, E_\Gamma$

1. initialize set of vertices $V_\Gamma \longleftarrow T$
2. initialize set of labeled edges $E_\Gamma \longleftarrow 0$
3. for all $(f, c) \in \mathcal{C}$
4. for all pairs t_i, t_j in c do
5. add an edge (t_i, t_j) to E_Γ with label Overlap Ratio $OS = \frac{2.|f|}{|t_i|+|t_j|}$
6. end for
7. end for

Determine overlap group in Γ

8. remove all edges in $\mathcal{T}$ for which $OS < ORT$
9. compute connected components $\mathcal{G}$
10. remove singletons from $\mathcal{G}$ of remaining graph Γ
11. return $\mathcal{G}$

Figure A.2: Algorithm 3: Group By Overlap

Group By Intersection

Input:

1. set T of trajectories,
2. set $\mathcal{C}$ determined in lines A.1 in Appendix
3. Intersection Ratio Threshold IRT

`Output: set of Groups G`

1. $\mathcal{G} \longleftarrow \emptyset$
2. for all $(f, c) \epsilon \mathcal{C}$
3. $\mathcal{G} \longleftarrow \mathcal{G} \bigcup \{c\}$
4. set initial Group Strength $GS(c) = |f|$
5. endfor

Merge Intersection Groups

6. repeat
7. for all $g_i, g_j \epsilon \mathcal{G}, g_i \neq g_j$ do
8. set Intersection Ratio $AS(g_i \bigcup g_j) = min(\frac{g_i \cap g_j}{g_j}, \frac{g_i \cap g_j}{g_i})$
9. if $AS(g_i \bigcup g_j) > IRT$ then $MS(g_i \bigcup g_j) = \frac{GS(g_i)+GS(g_j)}{2}$
10. else $MS(g_i \bigcup g_j) = 0$
11. end if
12. end for
13. find $g_{i*} \bigcup g_{j*}$ for which $MS(g_{i*} \bigcup g_{j*})$ is maximal
14. if $MS(g_{i*} \bigcup g_{j*}) \neq 0$ then
15. $\mathcal{G} \longleftarrow (\mathcal{G}\{g_{i*}, g_{j*}\})$
16. $GS(g_{i*} \bigcup g_{j*}) = MS(g_{i*} \bigcup g_{j*})$
17. end if
18. until $MS(g_{i*} \bigcup g_{j*}) = 0$
19. return $\mathcal{G}$

Figure A.3: Algorithm 3: Group By Intersection

Bibliography

[1] O. Baltzer, F. Dehne, S.Hambrusch, and A.Rau-Chaplin. Olap for trajectories. *Proc. 19th Int. Conference on Database and Expert Systems Applications (DEXA)*, pages 340–347, 2008.

[2] M. Benkerta, J. Gudmundssonb, F. Hbnera, and T. Wolle. Reporting flock patterns. *ScienceDirect*, 41(3):111–125, 2008.

[3] C. Borgelt. *http : //www.borgelt.net/software.html.* Retrieved December 11, 2009.

[4] K. Buchin, M. Buchin, M. v. Kreveld, and J. Luo. Finding long and similar parts of trajectories. 2009.

[5] H. Cao, N. Mamoulis, and D.W. Cheung. Mining frequent spatio-temporal sequential patterns. *Data Mining, Fifth IEEE International Conference on*, page 8, 2005.

[6] B. Djordjevic, J. Gudmundsson, A. Pham, and T. Wolle. Detecting regular visit patterns. *SpringerLink*, 5193:344–355, 2007.

[7] A. Escribano, L. Gomez, B. Kuijpers, and A. A. Vaisman. Piet: a gis-olap implementation. *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, pages 73–80, 2007.

[8] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. *International Conference on Knowledge Discovery and Data Mining, Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 330–339, 2007.

[9] G. Gidfalvi and T. B. Pedersen. Mining long, sharable patterns in trajectories of moving objects. *GeoInformatica*, 13(1):27–55, 2007.

[10] K. Hornsby. Temporal zooming. *Transactions in GIS*, 5:255 – 272, 2002.

[11] S. Hwang, Y. Liu, J. Chiu, and E. Lim. Mining mobile group patterns: A trajectory-based approach. *Advances in Knowledge Discovery and Data Mining*, 3518, 2005.

[12] W. H Inmon. Data warehouse. *Prism Solutions*, 1, 1995.

[13] W. H. Inmonna. http://www.datawarehouse4u.info/images/data_warehouse_architecture.jpg. Retrieved December 11, 2009.

[14] B. Kuijpers and A. Vaisman. A data model for moving objects supporting aggregation. *Proceedings of the ICDE Workshop on Spatio-Temporal Data Mining (STDM'07)*, 2007.

[15] P. Laube, M. v. Kreveld, and S. Imfeld. Finding remo detecting relative motion patterns in geospatial lifelines. *Developments in Spatial Data Handling 11th International Symposium on Spatial Data Handling*, pages 201–215, 2005.

[16] L. Leonardi, S. Orlando, A. Raffaet, A. Roncato, and C. Silvestri. Frequent spatio-temporal patterns in trajectory data warehouses. *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1433–1440, 2009.

[17] Y. Li, J. Han, and J. Yang. Clustering moving objects. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 617–622, New York, NY, USA, 2004. ACM.

[18] I. F. Vega Lopez, R. T. Snodgrass, and B. Moon. Spatiotemporal aggregate computation: A survey. *IEEE Transactions On Knowledge and Data Engineering*, 17(2):271–286, 2005.

[19] N. Mamoulis, H. Cao, and G. Kollios. Mining, indexing, and querying historical spatiotemporal data. *International Conference on Knowledge Discovery and Data Mining archive Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 236–245, 2004.

[20] G. Marketos, E. Frentzos, and I. Ntoutsi. Building real-world trajectory warehouses. *MobiDE 2008*, 2008.

[21] G. Marketos, E. Frentzos, and I. Ntoutsi. A framework for trajectory data warehousing. *Proceedings of ACM SIGMOD Workshop on Data Engineering for Wireless and Mobile Access*, 2008.

[22] M. Nanni and D. Pedreschi. Time-focused clustering of trajectories of moving objects. *PJournal of Intelligent Information Systems*, 27(3):267–289, 2006.

[23] S. Orlando, R. Orsini, A. Raffaet, A. Roncato, and C. Silvestri. Spatio-temporal aggregations in trajectory data warehouses. *Data Warehousing and Knowledge Discovery*, pages 66–77, 2007.

[24] S. Shaw, H. Yu, and L. S Bombom. A space-time gis approach to exploring large individual-based spatiotemporal datasets. *Transactions in GIS*, 12(4):425 – 441, 2008.

[25] C. Shim and J. Chang. A new similar trajectory retrieval scheme using k-wraping distance algorithm for moving objects. *WAIM 2003*, pages 433 – 444, 2003.

[26] I. Timko, M. H. Bhlen, and J. Gamper. Sequenced spatio-temporal aggregation in road networks. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, 360:48–59, 2009.

[27] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. *ICDE Proceedings of the 18th International Conference on Data Engineering*, page 673, 2002.

[28] J. Wang, W. Hsu, M. Lee, and J. Wang. Flowminer: Finding flow patterns in spatio-temporal databases. *Tools with Artificial Intelligence, IEEE International Conference on*, 0:14–21, 2004.

[29] Y. Wang, E. Lim, and S. Hwang. On mining group patterns of mobile users. *DEXA*, (2736):287–296, 2003.

[30] D. Zeinalipour-Yazti, S. Lin, and D. Gunopulos. Distributed spatio-temporal similarity search. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 14–23, New York, NY, USA, 2006. ACM.

[31] C. Zhou, S. Shekhar, and L. Terveen. Discovering personal paths from sparse gps traces. *1st International Workshop on Data Mining in conjunction with 8th Joint Conference on Information Sciences*, 2005.