OLAP for Trajectories *

O. Baltzer † **F. Dehne** ‡ **S. Hambrusch** § **A. Rau-Chaplin** ¶

Abstract

In this paper, we present an OLAP framework for trajectories of moving objects. We introduce a new operator GROUP_TRAJECTORIES for group-by operations on trajectories and present three implementation alternatives for computing groups of trajectories for group-by aggregation: *group by overlap*, *group by intersection*, and *group by overlap and intersection*. We also present an interactive OLAP environment for resolution drill-down/roll-up on sets of trajectories and parameter browsing. Using generated and real life moving data sets, we evaluate the performance of our GROUP_TRAJECTORIES operator.

1 Introduction

Global positioning (GPS) and RFID systems are creating vast amounts of spatio-temporal data for *moving objects*. A set of moving objects is typically stored as a relational table *objects* where each record contains a value *trajectory* $= [(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots (x_m, y_m, t_m)]$ representing the movement of the respective object as a sequence of positions at time $t_1, t_2, \ldots t_m$. The development of spatio-temporal databases for storing and manipulating spatio-temporal data is well advanced (see Section 2 below). However, in order to efficiently *analyze* large scale data sets representing moving objects, it is also important to have available the well established set of tools for OLAP analysis. In order to apply OLAP tools towards moving object datasets, it is necessary to aggregate with respect to *trajectory* as a *feature* dimension as well as a *measure* dimension.

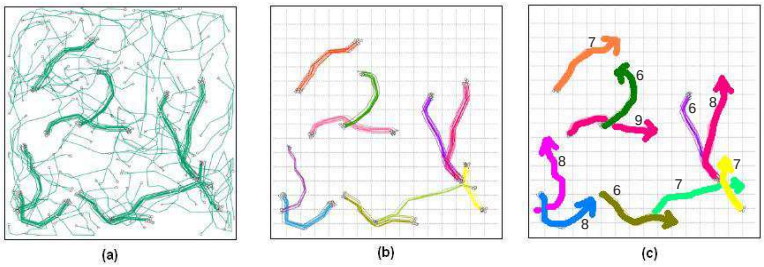


Figure 1: *OLAP For Trajectories* Example. (a) Input data. (b) Groups with minimum support. (c) Aggregate results reported (aggregate trajectories and counts).

We illustrate this with the example shown in Figure 1. Consider the trajectories shown in Figure 1a. We observe a number of individual objects that move on random paths plus 10 *groups* of objects that move together on similar paths. Each group consists of more then five objects moving on similar paths which, taken together, appear to the human eye as "bold" paths.

*Research partially supported by the Natural Sciences and Engineering Research Council of Canada.

†Faculty for Computer Science, Dalhousie University, Halifax, Canada, obaltzer@cs.dal.ca

‡School of Computer Science, Carleton University, Ottawa, Canada, frank@dehne.net, http://www.dehne.net

§Department of Computer Science, Purdue University, West Lafayette, Indiana, USA, seh@cs.purdue.edu, http://www.cs.purdue.edu/people/seh

¶Faculty for Computer Science, Dalhousie University, Halifax, Canada, arc@cs.dal.ca, http://users.cs.dal.ca/~arc/

Consider the following SQL query where *trajectory* is both, a *feature* dimension as well as a *measure* dimension:

```
SELECT AGGREGATE(trajectory) AS trajectory
       COUNT(trajectory) as count
FROM objects
GROUP BY GROUP_TRAJECTORIES(trajectory,
         resolution)
HAVING COUNT(*) >= 5
```

For this example, the aim of the GROUP BY operation with respect to *feature* dimension *trajectory* is to group similar trajectories and eliminate groups with less than minimum support (less than 5 similar trajectories). The resulting set of groups is shown in Figure 1b. Once the groups of trajectories have been determined, we report for each group an *aggregate trajectory* representing the trajectories in the group. In this example, the aggregate trajectory is the average trajectory computed by calculating for each time t_i the average of the locations (x_i, y_i) of the trajectories in the group. The result is shown in Figure 1c, where each group is represented by the aggregate trajectory and size of the group (count).

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 a relational table *objects*. The main problem arising is how to aggregate with respect to feature dimension *trajectory*. It is very unlikely that any two trajectories are exactly the same. Hence, standard aggregation of records with equivalent *trajectory* values is not very useful in most cases. We propose to partition the given trajectories into disjoint *groups* of trajectories using a new operator which we term GROUP_TRAJECTORIES. This operator returns for each trajectory a *group identifier*, and then OLAP can proceed with standard aggregation according to the group identifiers instead of the trajectories themselves.

The main problem addressed in this paper is how to define and compute the operator GROUP_TRAJECTORIES such that the resulting groups allow for a meaningful analysis of object movements via OLAP. We propose *three different versions* of the operator GROUP_TRAJECTORIES which compute groups of trajectories that are appropriate for OLAP analysis of trajectories for different circumstances and applications:

- GROUP_TRAJECTORIES: *Group by Overlap*
- GROUP_TRAJECTORIES: *Group by Intersection*
- GROUP_TRAJECTORIES: *Group by Overlap and Intersection*

Section 3 will show in detail how these three different versions of our GROUP_TRAJECTORIES operator are defined and computed. Here, we only outline the intuition behind these operators. Our *Group by Intersection* method aggregates subsets of trajectories that correspond to similar or synchronous movements. Figure 1 shows an example where movements that are along similar trajectories are aggregated. *Group by Intersection* will also aggregate parallel movements such as "marching band" style parallel trajectories. A schematic illustration is shown in Figure 2a. The trajectories shown could e.g. represent a group of four people walking together, and the aggregate would be a simplified representation of that movement. Our *Group by Overlap* method aggregates subsets of trajectories that correspond to sequences of movements with sufficient overlap between subsequent trajectories. A schematic illustration is shown in Figure 2b. The trajectories shown could e.g. represent movements of people who pass on a disease virus, and the aggregate would then represent the total movement of the virus. The *Group by Overlap and Intersection* method aggregates subsets of trajectories that correspond to a combination of sequences of movements and similar or synchronous movements. For the above virus spread example, this would account for the fact that the virus would also spread "sideways" among groups of people walking on parallel paths.

The definitions and algorithms for the three different versions of operator GROUP_TRAJECTORIES presented in Section 3 are guided by various parameters, including spatial and time resolution.

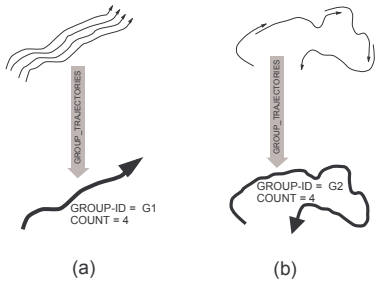


Figure 2: Illustration of two different versions of operator GROUP_TRAJECTORIES (a) Group by Intersection. (b) Group by Overlap.

This allows for analyzing trajectories at various levels of detail/resolution and provides another opportunity for OLAP analysis of trajectories. In Section 4, we present an *interactive OLAP environment* for analysis of trajectories that allows resolution drill-down and roll-up as well as parameter browsing. Our new interactive OLAP environment can be accessed at `http://OLAP-T.cgmlab.org`.

An experimental evaluation of our *OLAP analysis for trajectories* methods is presented in Section 5. The main goal of the experiments is to determine how well the three different versions of the GROUP_TRAJECTORIES operator allow for a meaningful analysis of object movements via OLAP. We have used various generated and real life moving object data sets and tested whether GROUP_TRAJECTORIES operator is appropriate for OLAP analysis of trajectories for different circumstances and applications.

The main contributions of this paper can be summarized as follows:

- An OLAP framework for moving object databases containing a feature dimension *trajectory*.
- A new operator GROUP_TRAJECTORIES for calculating group-bys with respect to feature dimension *trajectory*, with algorithms for three different scenarios: *Group by Overlap*, *Group by Intersection*, and *Group by Overlap and Intersection*. (Section 3)
- An *interactive OLAP environment* for the analysis of trajectories that allows resolution drill-down and roll-up as well as parameter browsing. (Section 4)
- An experimental analysis of our methods for various generated and real life moving object data sets. (Section 5)

2 Related Work

There is a wealth of literature on spatiotemporal data analysis and aggregation. See e.g. [8] for a survey. This work studies aggregation by specific temporal dimensions such as "by day" or "by year", or by strict topological association such as "by location square" or "within 10 km of" (e.g. [10]). In our case, we wish to aggregate entire trajectories. The FlowMiner approach [17] does also not relate trajectories to each other but rather tries to find relationships among events that occur over a period of time. For the detection of relationships among trajectories in a moving object database we found in the literature five groups of approaches:

- Variations of frequent pattern or association rule mining (e.g. [2, 3, 4, 5, 9, 15, 18]).
- Clustering techniques (e.g. [7, 11]).
- Computational Geometry techniques (e.g [6]).
- Neural network based techniques (e.g. [14]).

- Edit distance, warping techniques and longest common subsequence (LCSS) extraction (e.g. [12, 13, 16, 19]).

The closest work to our proposed OLAP framework for trajectories is Gidófalvi and Pedersen's work [3] on detecting shareable patterns in trajectories of moving objects for applications like ride sharing (and similar work in [9, 2]). They use frequent itemsets to detect long shareable paths and cover a *special case* of our approach where one only wants to aggregate trajectories that are very similar and have a very large overlap. In this paper, we present algorithms that cover various kinds of grouping and aggregation that are important in an OLAP setting. In particular, we can also detect and aggregate groups of trajectories that correspond to "marching band" style parallel movements as well as groups of trajectories corresponding to sequences of movements that are linked to each other. None of these types of group movements can be detected by Gidófalvi and Pedersen's methods. Algorithmically, their work is based exclusively on analyzing frequent item sets, whereas our methods consist of three phases: frequent itemset mining, reverse matching, and group merging. Our additional reverse matching and group merging phases are, to our knowledge, novel approaches. They lead to more meaningful and more flexible aggregation of trajectories and they also lead to a much smaller number of groups which results in more OLAP aggregation. It is generally the case that for frequent pattern mining and edit distance derived approaches, the number of patterns that are detected can be very large resulting in a very large number of groups, which can lead to very little OLAP aggregation. For example, [18, 4] define equivalence regions around locations and line segments, and propose modified version of Apriori and FP-growth for frequent pattern mining. [9, 5] approximate the locations of trajectories by regions and performs frequent pattern mining across those regions. [2] approximates the segments of a trajectory with coarser line segments and performs frequent pattern mining over those line segments. Similar arguments apply to clustering techniques (e.g. [7, 11]), Computational Geometry techniques (e.g [6]), and Neural Network based techniques (e.g. [14]) which focus on finding similarity between paths but do not cover the various kinds of grouping and aggregation that are important in an OLAP setting. The edit distance, warping technique or longest common subsequence based methods listed above do perform trajectory comparisons but are designed for entirely different types of applications and input data. The methods in [12] use LCSS for matching and identifying motion from video. Similarly, [16] uses LCSS for matching similar multi-dimensional trajectories for sign-language recognition, and [13] uses k-warping algorithms to index soccer games recorded on video.

3 Computing Groups Of Trajectories

Consider N moving objects on a 2D spatial grid. Each object is identified by a unique *tag* number (similar to EPC in RFID). Object movements are recorded through a set of readings $((x, y), i, t)$ indicating that object (tag) i was detected at time t within the grid cell located at (x, y). The N moving objects are represented by a relational table *objects* with N records. Each record contains values *tag, name, size, color,* etc. describing one object according to a star schema. Among them is a value *trajectory* representing the movement of the respective object as a sequence $[(x_1, y_1, t_1)$, (x_2, y_2, t_2), ... $(x_m, y_m, t_m)]$ of positions at time $t = t_1, t_2, \ldots t_m$. Our goal is to aggregate with respect to dimension *trajectory*. For this purpose, we define a new operator GROUP_TRAJECTORIES which returns for each trajectory a *group identifier*, and then proceed with standard OLAP aggregation according to the group identifiers instead of the trajectories themselves.

In this section we present three different implementations of the operator GROUP_TRAJECTORIES which compute groups of trajectories that are appropriate for OLAP analysis of trajectories for different circumstances and applications: Group by Overlap, Group by Intersection, and Group by Overlap and Intersection.

Algorithm 1 outlines our high level framework for the three implementations of operator GROUP_TRAJECTORIES. We first apply a time and space resolution mapping of our initial set $\mathcal{T}$ of trajectories (Line 1). This allows for the resolution drill-down and roll-up within our interactive OLAP framework for trajectories discussed in Section 4. Here, we want to allow the OLAP user to switch between different levels of resolution. For example, time granularity "day" may be sufficient for a

high level analysis of GPS data for the movement of a fleet of ships. However, a drill-down to the set of paths taken by a group of ships entering a port may require a time granularity "minute". Next, we compute frequent itemsets for the mapped set of trajectories (Line 2) and then apply a reverse mapping step (Lines 3-7). Here, we determine for each frequent itemset f, the corresponding *original* group c of trajectories and create a set $\mathcal{C}$ of resulting (f, c) pairs.

The most important part of our method is the group merging phase (Line 8). Here, we present three different methods: (a) Group by Overlap (Algorithm 2), (b) Group by Intersection (Algorithm 3), and (c) Group by Overlap and Intersection (Algorithm 4). These three methods are discussed in detail in the following Sections 3.1, 3.2 and 3.3, respectively.

Algorithm 1 High-level Trajectory Aggregation Framework

Input:
1. set $\mathcal{T}$ of trajectories,
2. space resolution r_{space},
3. time resolution r_{time},
4. minimum support s,
5. minimum length l.

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

1: map $\mathcal{T}$ to resolution (r_{space}, r_{time}) resulting in $\mathcal{T}'$
2: compute frequent itemsets $\mathcal{F}$ in $\mathcal{T}'$ with minimum support s and minimum length l
Reverse matching
3: $\mathcal{C} \leftarrow \emptyset$
4: **for all** $f \in \mathcal{F}$ **do**
5: determine the subset c of $\mathcal{T}$ that corresponds to f
6: $\mathcal{C} \leftarrow \mathcal{C} \cup \{(f, c)\}$
7: **end for**
Group merging
8: select one of
(a) Group by Overlap (Algorithm 2)
(b) Group by Intersection (Algorithm 3)
(c) Group by Overlap and Intersection (Algorithm 4)
9: **return** the resulting set of groups $\mathcal{G}$

3.1 Group By Overlap

Our *Group By Overlap* method presented in Algorithm 2 introduces a tunable parameter *overlap ratio threshold ORT* which controls the strength of the grouping process. The interactive OLAP framework for trajectories discussed in Section 4 will allow for an interactive tuning of this parameter.

Algorithm 2 is based on an *overlap graph* Γ, where each vertex corresponds to a trajectory (Lines 1 and 2). For each frequent item set f and corresponding set c of trajectories, we consider all pairs of trajectories $t_i, t_j \in c$ and add for each pair an edge (t_i, t_j) with label *overlap ratio* $OS = \frac{2 \cdot |f|}{|t_i| + |t_j|}$ (Lines 3-7). The intuition behind the definition of *overlap ratio* is illustrated in Figure 3a. It measures the size of the overlap relative to the sizes of the trajectories. We then remove all edges where the *overlap ratio* OS is smaller than the chosen *overlap ratio threshold* ORT (Line 8) and compute the connected components of the remaining graph (Line 9). These components correspond to the groups of trajectories that are reported.

The nature of the obtained groups of trajectories is determined by two factors. (1) The *overlap ratio threshold* ORT determines how much two neighboring trajectories within a group have to overlap. (2) The graph connected component construction (Line 9) allows for an "adding up" of trajectories corresponding to a "relay" type of movement. Depending on the chosen *overlap ratio threshold* ORT, the "relay" parties will have to move in unison for more or less of their own individual movements.

Algorithm 2 Group by Overlap

Input:

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

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

Build overlap graph $\Gamma = (V_\Gamma, E_\Gamma)$

1: initialize set of vertices $V_\Gamma \leftarrow \mathcal{T}$
2: initialize set of labeled edges $E_\Gamma \leftarrow \emptyset$
3: **for all** $(f, c) \in \mathcal{C}$ **do**
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 $OS = \frac{2 \cdot |f|}{|t_i| + |t_j|}$
6: **end for**
7: **end for**

Determine overlap groups in Γ

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

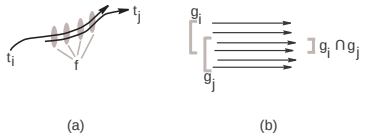


Figure 3: Illustration of (a) *Overlap Ratio* and (b) *Intersection Ratio*.

3.2 Group By Intersection

Our *Group By Intersection* method presented in Algorithm 3 introduces a tunable parameter *intersection ratio threshold* IRT which controls the strength of the grouping process. The interactive OLAP framework for trajectories discussed in Section 4 will allow for an interactive tuning of this parameter.

Algorithm 3 first creates an initial set $\mathcal{G}$ of groups of trajectories (Lines 1-5), where each group c corresponds to a frequent itemset f determined in the reverse matching in lines 3-7 of Algorithm 1. Each group c is assigned a *group strength* $GS(c)$ which is initially set to the size of the respective frequent itemset (Line 4).

The remainder of Algorithm 3 (Lines 6-21) merges groups in $\mathcal{G}$ by iterating the following loop. We compute for each pair $g_i, g_j \in \mathcal{G}$ a value *intersection ratio* $AS(g_i \cup g_j)$ which represents the number of trajectories that occur in both g_i and g_j, relative to the sizes of g_i and g_j (Line 8). The intuition behind our definition of *intersection ratio* is illustrated in Figure 3b. We will consider as candidates for merging all pairs g_i, g_j whose intersection ratio is larger than our input parameter *intersection ratio threshold* IRT and compute for each such pair a value *merge strength* $MS(g_i \cup g_j)$ which is the average of their *group strength* values (Line 9-14). All candidate pairs are ranked by their *merge strength* and we will merge the pair g_i*, g_j* with maximum merge strength, or one of the maximal pairs if there are multiple (Lines 15-17). The *group strength* $GS(g_{i*} \cup g_{j*})$ of the new merged group will be the *merge strength* calculated above (Line 18). This process is repeated until there are no more pairs of groups with non zero *merge strength* (Line 20), that is, until there are no more pairs of groups with *intersection ratio* larger than the *intersection ratio threshold* IRT (Lines 9 and 12).

Our *Group by Intersection* method aggregates subsets of trajectories that correspond to "march-

ing band" style parallel movements. The nature of the obtained groups of trajectories is determines by two factors. (1) The *intersection ratio threshold* IRT determines how many shared trajectories between two groups is "sufficient" for them to be merged. (2) The merging process in Lines 6 to 21 which is similar in nature to a minimum spanning tree calculation. We merge first the largest groups with sufficient shared trajectories and then work our way down to the smaller groups. Unlike the *Group by Overlap* method which combines sequences of movements, the *Group by Intersection* method combines parallel of movements.

Algorithm 3 Group by Intersection

Input:
1. set $\mathcal{C}$ determined in lines 3-7 of Algorithm 1,
2. Intersection Ratio Threshold IRT.

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

Create initial set of intersection groups

1: $\mathcal{G} \leftarrow \emptyset$
2: **for all** $(f, c) \in \mathcal{C}$ **do**
3: $\quad \mathcal{G} \leftarrow \mathcal{G} \cup \{c\}$
4: $\quad$ set initial Group Strength $GS(c) = |f|$
5: **end for**

Merge intersection groups

6: **repeat**
7: $\quad$ **for all** $g_i, g_j \in \mathcal{G}, g_i \neq g_j$ **do**
8: $\quad\quad$ set Intersection Ratio $AS(g_i \cup g_j) = \min\left(\frac{|g_i \cap g_j|}{|g_1|}, \frac{|g_i \cap g_j|}{|g_2|}\right)$
9: $\quad\quad$ **if** $AS(g_i \cup g_j) > IRT$ **then**
10: $\quad\quad\quad$ set Merge Strength $MS(g_i \cup g_j) = \frac{GS(g_i)+GS(g_j)}{2}$
11: $\quad\quad$ **else**
12: $\quad\quad\quad$ set Merge Strength $MS(g_i \cup g_j) = 0$
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: $\quad$ find $g_{i*} \cup g_{j*}$ for which $MS(g_{i*} \cup g_{j*})$ is maximal
16: $\quad$ **if** $MS(g_{i*} \cup g_{j*}) \neq 0$ **then**
17: $\quad\quad \mathcal{G} \leftarrow (\mathcal{G} \setminus \{g_{i*}, g_{j*}\}) \cup \{g_{i*} \cup g_{j*}\}$
18: $\quad\quad$ set Group Strength $GS(g_{i*} \cup g_{j*}) = MS(g_{i*} \cup g_{j*})$
19: $\quad$ **end if**
20: **until** $MS(g_{i*} \cup g_{j*}) = 0$
21: **return** $\mathcal{G}$

3.3 Group By Intersection and Overlap

The goal of our *Group by Intersection and Overlap* method is to group both, sequences of movements and parallel movements. Algorithm 4 is a combination of Algorithm 2 and Algorithm 3. In Lines 1-20, we create the same set $\mathcal{G}'$ of groups of trajectories as in Algorithm 3. In Lines 21-28, we create the same overlap graph Γ as in Algorithm 2. The combination occurs in Line 29, where we add to Γ a clique for each $g \in \mathcal{G}'$ (i.e. edges between all pairs of trajectories $t_1, t_2 \in g$). Line 30 then computes the connected components of the modified graph Γ. Each connected component corresponds to a group of trajectories.

The resulting groups are strings of overlapping trajectories as in our *Group by Overlap* method to which we add parallel trajectories as in our *Group by Intersection* method. The aggregation is guided by two parameters, the *intersection strength threshold* IRT and the *overlap ratio threshold*

ORT, which control the width and length, respectively, of the generated groups.

Algorithm 4 Group by Intersection and Overlap

Input:
1. set $\mathcal{T}$ of trajectories,
2. set $\mathcal{C}$ determined in lines 3-7 of Algorithm 1,
3. Intersection Ratio Threshold IRT,
4. Overlap Ratio Threshold ORT.

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

Group by Intersection:
Create initial set of intersection groups

1: Lines 1-5 of Algorithm 3

Group by Intersection:
Merge intersection groups

2: Lines 6-20 of Algorithm 3

Group by Overlap:
Build overlap graph $\Gamma = (V_\Gamma, E_\Gamma)$

3: Lines 1-7 of Algorithm 2

Group by Overlap:
Determine overlap groups in Γ

4: Lines 8 of Algorithm 2

Combine groupings

5: For each $g \in \mathcal{G}$, add a clique connecting all $t \in g$ to the graph Γ
6: compute the connected components $\mathcal{G}'$ of Γ
7: remove all singletons from $\mathcal{G}'$
8: **return** $\mathcal{G}'$

4 Interactive OLAP For Trajectories

The algorithms for the three different versions of operator GROUP_TRAJECTORIES presented in Section 3 are guided by the following parameters: space resolution, time resolution, minimum support, intersection ratio threshold and overlap ratio threshold. This allows to analyze groups of trajectories for various levels of resolution or connectedness, and provides another opportunity for OLAP analysis of trajectories. For example, for a high level analysis of GPS data for the movement of a fleet of ships, time granularity "day" may be sufficient. However, a drill-down to viewing the paths taken by a group of ships when entering a port may require a time granularity "minute". As an example for browsing a parameter like *overlap ratio threshold*, consider a set of trajectories representing movements of people who pass on a disease virus. The aggregate, using our *Group by Overlap* method, could be used to analyze the total movement of the virus. In this example, our parameter *overlap ratio threshold* would represent to amount of interaction between individuals required to pass on the virus. Changing the threshold value allows to evaluate how far the virus will spread based on different assumption about its transmission.

We have built a prototype *interactive environment for the analysis of trajectories* that allows resolution drill-down and roll-up as well as parameter browsing. It can be accessed at `http://OLAP-T.cgmlab.org`. A screen image is shown in Figure 4. At this point, our system visualizes an implementation of the operator GROUP_TRAJECTORIES presented in this paper. It does not yet include other OLAP functionality. Our system allows to explore the results of operator GROUP_-TRAJECTORIES depending on different resolution and threshold values for several synthetic and real life data sets.

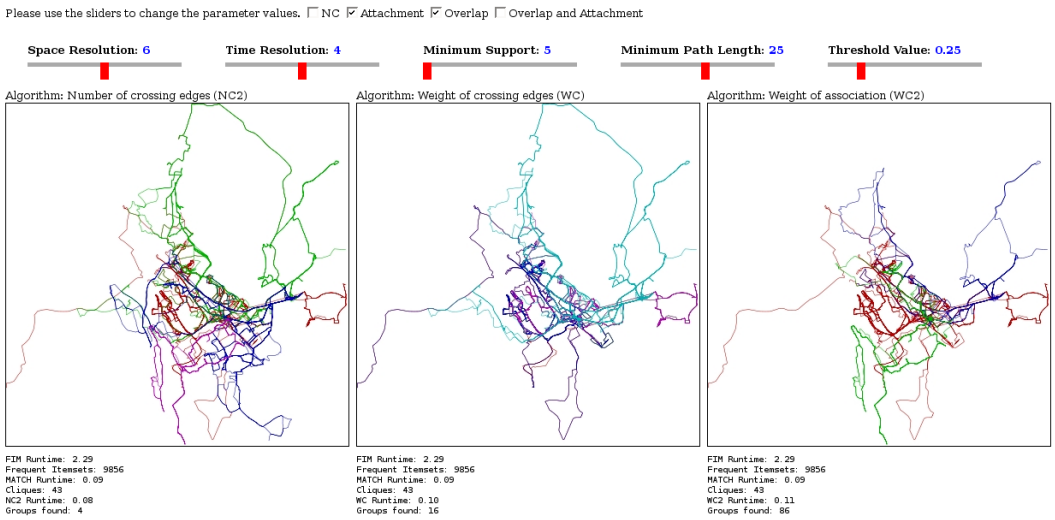


Figure 4: Interactive Environment For The Analysis Of Trajectories at `http://OLAP-T.cgmlab.org`

5 Experimental Evaluation

Our experimental evaluation consists of three parts. In Section 5.1, we study the robustness of our GROUP_TRAJECTORIES operator against background noise. In Section 5.2, we study the influence of the algorithms' parameters on the results produced. Section 5.3 presents experimental results with real world data and compares them to results obtained by mining frequent itemsets only.

5.1 Robustness Against Noise

We first tested the robustness of our GROUP_TRAJECTORIES implementations against background noise. For that, we created groups with 10 trajectories each, and then added random trajectories as background noise. The question is, what level of noise (ratio between number of random trajectories and total number of trajectories) can we have while maintaining a correct result where GROUP_TRAJECTORIES reports the original groups and discards the randomly added trajectories as groups with less than minimum support? We tested this for our *Group by Intersection* and *Group by Overlap* methods on groups of trajectories that should be grouped together for these methods. For *Group by Intersection* we chose groups of parallel paths, and for *Group by Overlap* we chose groups of paths that connect into spiral arrangements (via intersection). We then added the noise (random trajectories). The input data and results obtained are shown in Figures 5 and 6.

Our *Group by Overlap* method has a surprising resilience against background noise. On the example shown in Figures 5, as well as many other examples that we tested, the *Group by Overlap* method has no trouble reporting the correct result for noise levels of 50%, 75% and even as high as 95%. At a noise level of 95%, the human eye can no longer visually detect the original groups of parallel paths but our *Group by Overlap* method has no problem reporting the correct result.

Our *Group by Intersection* method has a somewhat lower resilience against background noise, as shown in Figures 6. This is not surprising because the intersection between trajectories that need to be assigned to the same group is somewhat weaker. For 5% and 25% background noise, it still detects the original groups plus one additional group consisting of a subset of the random noise trajectories that happen to get connected. For 50% background noise, the number of these random groups becomes larger than the number of original groups. Other experiments have shown that, in general, our *Group by Intersection* method reports useful results for up to 5% noise.

5.2 Input Parameters

In this section we examine the influence of the algorithms' input parameters on the results produced. As input data, we use a synthetic dataset which consists of mix of groups of trajectories. Some groups are of the type that is best for *Group by Intersection and some groups are of the type that is best for* Group by Overlap. The dataset is shown in Figure 7. It consists of three spirals with parallel paths in each spiral. While a spiral-like movement is not a pattern commonly occurring in real world data, this represents a challenging pattern for our methods. Note the subdivision of the spiral into several color-coded segments. Each such segment represents a group of trajectories that are moving in unison. To achieve a more realistic movement, a small random variance is added to the movement of each trajectory. Furthermore, each segment overlaps with the previous and the following segment.

The *overlap ratio threshold* (ORT) and *intersection strength threshold* (IRT) input parameters for our three GROUP_TRAJECTORIES implementations influence their sensitivity towards identifying groups.

Using the dataset in Figure 7, we tested our GROUP_TRAJECTORIES implementations for various values of ORT and IRT. The results are shown in Figure 8.

We observe that for lower values of ORT and IRT, the identified groups become larger in size but fewer groups are identified. This behaviour is expected as a lower threshold provides less constraints on the formation of groups. For larger values for ORT and IRT on the other hand, the algorithms become more restrictive in terms of identifying and merging groups and other, more subtle patterns are detected.

In the example shown in Figure 8, the *Group by Overlap* method identifies larger groups for low values of ORT, and reports the entire spirals. The identified groups become closer to the initial set of groups of trajectories in the spiral as the value for ORT increases. For large values of ORT, there is insufficient overlap between parallel paths and the reported groups become thinner. For the *Group by Intersection* method we observe that the number of groups reported increases with increasing value for IRT. In the example shown in Figure 8, the *Group by Overlap and Intersection* method displays a very interesting behavior.It appears to be nearly indifferent to the choice of the threshold values ORT and IRT. It consistently identifies each spiral as a separate group. Only for higher values of $ORT = IRT = 0.7$ does it show a slightly different result where each spiral is partitioned into very few groups. A summary of the number of groups reported as a function of $ORT = IRT$ is given in Figure 9.

5.3 Real World Data

For the evaluation of our methods on real world data, we have chosen the school buses dataset that can be freely obtained from [1]. The dataset contains 145 trajectories of buses that are moving in and around an urban area.

Figure 10(a) shows the majority of the data set around the urban center. (A few routes going to far out place were removed to better display the data.) Figure 10(b) represents the 76 groups that are identified by applying only frequent itemsets mining (as e.g. in [3, 9, 2]) (plus a minimum length cutoff as used in our methods). The large number of groups reported by frequent itemsets mining based methods is often a disadvantage because it does not lead to signifficant aggregation in an OLAP setting. Figure 10(c) shows the results obtained with our *Group by Overlap* method for ORT values 0.4, 0.5, 0.6, and 0.7. We observe that the parameter ORT in our *Group by Intersection* method allows for a much finer control over the grouping of trajectories reported and that the *Group by Intersection* method reports a considerably smaller number of groups.

References

[1] R-tree Portal. `http://www.rtreeportal.org/`. Last accessed November 16, 2007.

[2] Cao, H., Mamoulis, N., and Cheung, D. W. Mining frequent spatio-temporal sequential patterns. *icdm 0* (2005), 82–89.

[3] Gidófalvi, G., and Pedersen, T. B. Mining Long, Sharable Patterns in Trajectories of Moving Objects. In *STDBM '06: Proceedings of the 3rd Workshop on Spatio-Temporal Database Management* (2006).

[4] Hwang, S., Liu, Y., Chiu, J., and Lim, E. Mining mobile group patterns: A trajectory-based approach. *Proceedings of PAKDD05* (2005), 713–718.

[5] Kim, D., Kang, H., Hong, D., Yun, J., and Han, K. STMPE: An Efficient Movement Pattern Extraction Algorithm for Spatio-temporal Data Mining. *LECTURE NOTES IN COMPUTER SCIENCE 3981* (2006), 259.

[6] Laube, P., van Kreveld, M., and Imfeld, S. Finding REMO–detecting relative motion patterns in geospatial lifelines. *Developments in Spatial Data Handling: Proceedings of the 11th International Symposium on Spatial Data Handling* (2004), 201–214.

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

[8] López, I. F. V., Snodgrass, R. T., and Moon, B. Spatiotemporal Aggregate Computation: A Survey. *IEEE Transactions on Knowledge and Data Engineering 17*, 2 (2005), 271–286.

[9] Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., and Cheung, D. Mining, indexing, and querying historical spatiotemporal data. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining* (2004), 236–245.

[10] Marchand, P., Brisebois, A., Bédard, Y., and Edwards, G. Implementation and evaluation of a hypercube-based method for spatiotemporal exploration and analysis. *ISPRS Journal of Photogrammetry and Remote Sensing 59*, 1-2 (2004), 6–20.

[11] Nanni, M., and Pedreschi, D. Time-focused clustering of trajectories of moving objects. *J. Intell. Inf. Syst. 27*, 3 (2006), 267–289.

[12] Sclaroff, S., Kollios, G., and Betke, M. Motion mining: discovering spatio-temporal patterns in databases of human motion. *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* (2001).

[13] Shim, C., and Chang, J. A new similar trajectory retrieval scheme using k-warping distance algorithm for moving objects. *Proceedings of the 4th International Conference on Advances in Web-Age Information Management,(WAIM 2003)*, 433–444.

[14] Sumpter, N., and Bulpitt, A. Learning spatio-temporal patterns for predicting object behaviour, 1998.

[15] Verhein, F., and Chawla, S. Mining spatio-temporal patterns in object mobility databases. *Data Mining and Knowledge Discovery* (2007).

[16] Vlachos, M., Kollios, G., and Gunopulos, D. Discovering similar multidimensional trajectories. *Data Engineering, 2002. Proceedings. 18th International Conference on* (2002), 673–684.

[17] Wang, J., Hsu, W., and Lee, M. FlowMiner: finding flow patterns in spatio-temporal databases. *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on* (2004), 14–21.

[18] Wang, Y., Lim, E.-P., and Hwang, S.-Y. On Mining Group Patterns of Mobile Users. In *DEXA* (2003), V. Marík, W. Retschitzegger, and O. Stepánková, Eds., vol. 2736 of *Lecture Notes in Computer Science*, Springer, pp. 287–296.

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

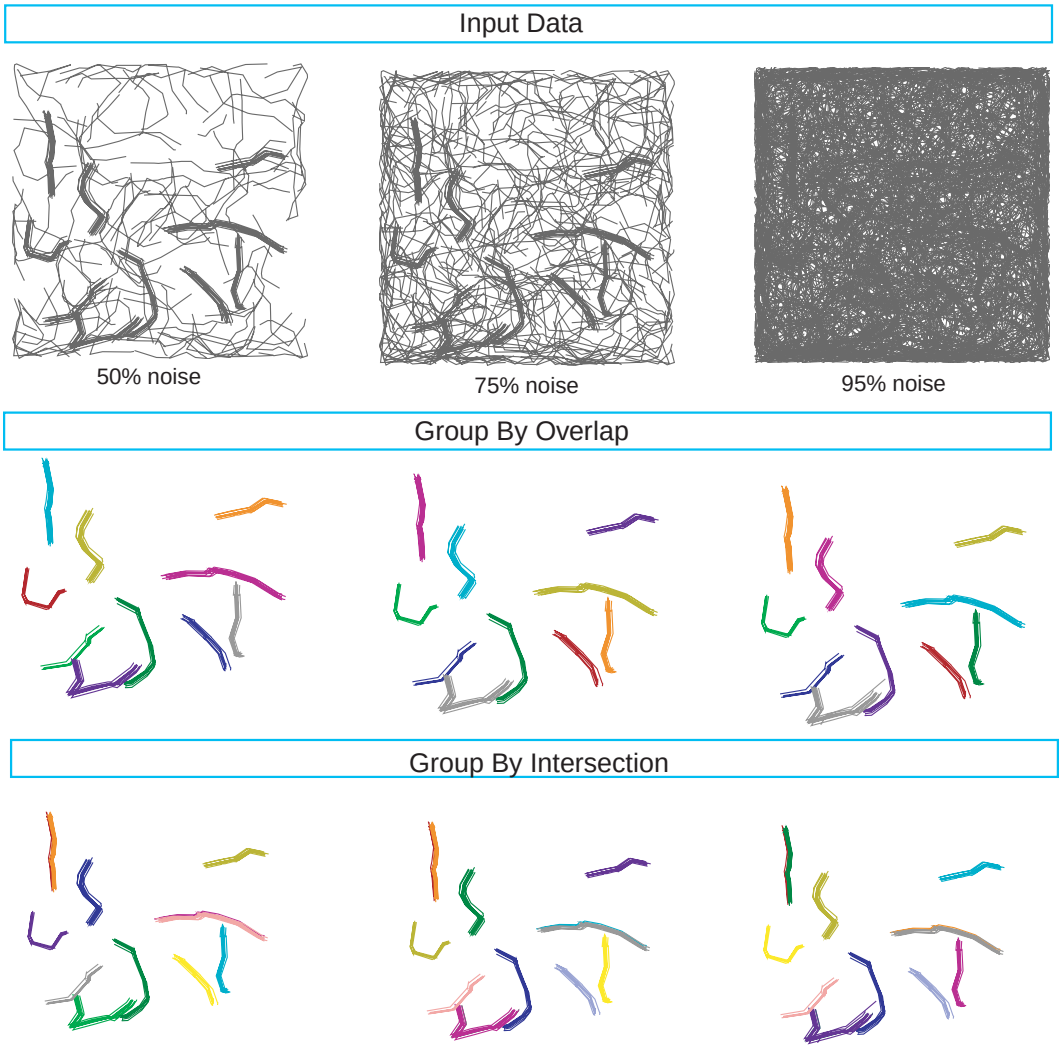


Figure 5: Test of robustness against noise. Top row: input data consisting of 10 groups with 10 similar trajectories each and three levels of noise: 50%, 75% and 95%. Center row: Groups computed by GROUP_TRAJECTORIES: *Group By Overlap* ($ORT = 0.5$, $min\_support = 4$). Bottom row: Groups computed by GROUP_TRAJECTORIES: *Group By Intersection* ($IRT = 0.5$, $min\_support = 4$). Groups are identified by color (*group identifier* = color).

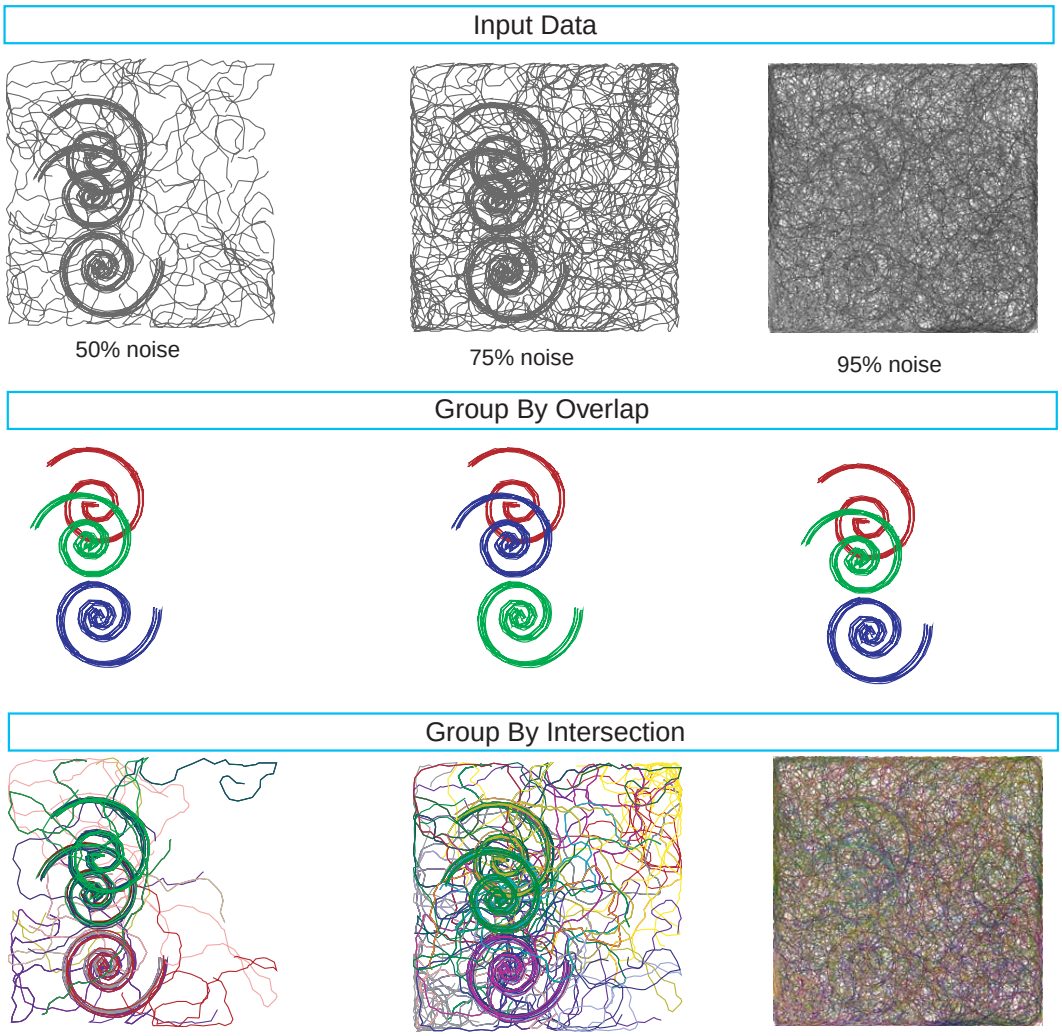


Figure 6: Test of robustness against noise. Top row: input data consisting of 3 groups with 10 spiral shaped trajectories each and three levels of noise: 5%, 25% and 50%. Center row: Groups computed by GROUP_TRAJECTORIES: *Group By Overlap* ($ORT = 0.5$, $min\_support = 4$). Bottom row: Groups computed by GROUP_TRAJECTORIES: *Group By Intersection* ($IRT = 0.5$, $min\_support = 4$). Groups are identified by color (*group identifier* = color).

Figure 7: Synthetic dataset with 24 groups each consisting of 10 trajectories and a partial overlap of approximately 25%.

(a) Group by Overlap

(b) Group by Intersection

(c) Group by Overlap and Intersection

Figure 8: Groups (identified by color) computed by each of our methods for $ORT = IRT = 0.1$, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7 ($min\_support = 4$, $min\_length = 4$).

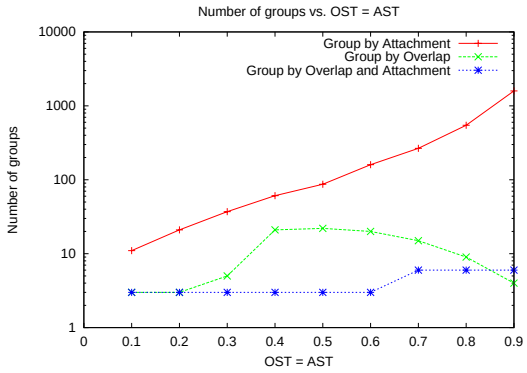
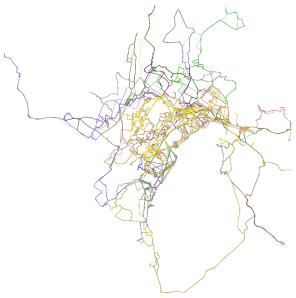
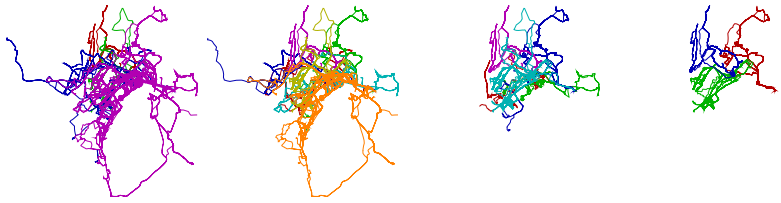


Figure 9: Relationship between the number of identified groups and values for Overlap Strength Threshold ORT and Intersection Ratio Threshold IRT.

(a) Entire dataset



(b) Groups reported (identified by color) using frequent itemset mining only (76 groups).



(c) Groups reported (identified by color) using *Group by Overlap* and ORT = 0.4, 0.5, 0.6, 0.7 ($min\_support$ = 5, min_length = 30).

Figure 10: Results obtained for the School Buses Dataset.