Exploring Spatiotemporal Patterns By Integrating Visual Analytics With A Moving Objects Database System

Mahmoud A. Sakr [#,*,1], Thomas Behr [#,2], Ralf Hartmut Güting [#,3], Gennady Andrienko [⋆,4], Natalia Andrienko [⋆,5], Christophe Hurter [◦,6]

[#] *Database Systems for New Applications, FernUniversität in Hagen*
58084 Hagen, Germany

[*] *Faculty of Computer and Information Sciences, University of Ain Shams*
Cairo, Egypt

[⋆] *Fraunhofer Institute IAIS, Germany*

[◦] *ENAC, Toulouse, France*

[1] mahmoud.sakr@fernuni-hagen.de

[2] thomas.behr@fernuni-hagen.de

[3] rhg@fernuni-hagen.de

[4] gennady.andrienko@iais.fraunhofer.de

[5] natalia.andrienko@iais.fraunhofer.de

[6] christophe.hurter@aviation-civile.gouv.fr

Abstract

In previous work, we have proposed a tool for *Spatiotemporal Pattern Query*. It matches individual moving object trajectories against a given movement pattern. For example, it can be used to find the situations of *Missed Approach* in ATC data (Air Traffic Control systems, used for tracking the movement of aircrafts), where the landing of the aircraft was interrupted for some reason. This tool expresses the pattern as a set of predicates that must be fulfilled in a certain temporal order. It is implemented as a Plugin to the Secondo DBMS system. Although the tool is generic and flexible, domain expertise is required to formulate and tune queries. The user has to decide the set of predicates, their arguments, and the temporal constraints that best describe the pattern. This paper demonstrates a novel solution where a *Visual Analytics* system, *V-Analytics*, is used in integration with this query tool to help a human analyst explore such patterns. The demonstration is based on a real ATC data set.

1 Introduction

This paper demonstrates the use of the visual analytics system *V-Analytics* [3] in integration with the moving objects DBMS SECONDO [1] to explore *Spatiotemporal Patterns* (STP) to analyze a real world data set. SECONDO brings an extensive set of query operations accessible through a query language to let the user express arbitrarily complex queries and efficiently evaluate them on large moving objects databases. On the other hand, V-Analytics enables the user to pre-process the data, select an interesting subset for analysis, tune parameters of the sophisticated queries in SECONDO and interpret their results.

In this paper, we demonstrate two pattern exploration tasks on a data set that includes one day (Feb 22, 2008) radar recording of aircraft positions (Lon, Lat, Alt) over France and neighboring territories, consisting of 17,851 trajectories and 427,651 recorded positions. The sampling interval is variable, ranging from about one to about three minutes. The data has been anonymized by removing aircraft identities and airline information. In this data set, we search for the following landing patterns:

1. Missed Approach: while the aircraft is in its *final approach* (i.e., the landing phase) the pilot for some reason decides to climb again. In order to improve the safety, it is worthwhile to analyze such hazardous situations. Such analysis can highlight redundant situations that can be solved by changing airways, approach procedures, etc.

2. Stepwise Descent Landing: a landing pattern in which the aircraft alternates between descent and cruise till it touches the ground. This is a less desirable landing pattern compared to the *Diving* pattern, where the aircraft constantly descends till it lands. The latter is preferable because of lower fuel consumption and lower pollution and noise.

Studying such patterns in the ATC data helps to improve the air traffic. The rest of this paper is organized as follows. Section 2 briefly describes the two systems SECONDO and V-Analytics, and discusses issues related to their integration. Section 3 briefly describes the *Spatiotemporal Pattern Predicate*. Section 4 illustrates what will be demonstrated. Finally we conclude in Section 5.

2 Integrating SECONDO and V-Analytics

SECONDO [1] is an open source extensible DBMS platform. It consists of three modules: the kernel, the GUI and the query optimizer. It contains an extensible set of *algebras*, each of which is defining database types and query operations. It contains algebras implementing a large part of the moving objects database model in [6]. We provide the implementation of the spatiotemporal pattern predicate as a SECONDO Algebra which is included in the SECONDO distribution version 3.2 or later.

V-Analytics [3] incorporates various visualization techniques, interactive tools, and computational methods for analyzing spatial, temporal, and spatiotemporal data. Among them are time-aware maps, space-time cubes, other types of graphs and diagrams; interactive dynamic filtering of data according to their spatial, temporal, and thematic (attributive) components; computational procedures oriented to movement data such as clustering of trajectories [4], generalization and summarization [2], extracting various types of events from movement data, etc.

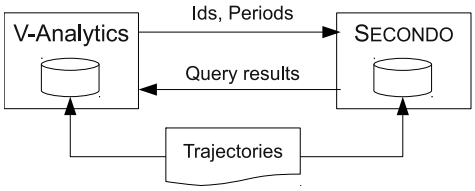

Figure 1: Integrating SECONDO with V-Analytics

SECONDO and V-Analytics are integrated so that it is possible to exchange data between the two systems in both directions (Figure 1). The moving object trajectories are initially loaded into the databases of the two systems. One can start by visualizing the data in V-Analytics, select a set of candidate trajectories, and/or interesting time periods within their life times, then send the trajectory identifiers and periods to SECONDO for further processing. In the other direction, one can perform sophisticated query operations in SECONDO (e.g., STP queries) and compute derived attributes, then transfer the query results to V-Analytics for visualization and further analysis. This process of transferring data between the two systems allows the user to explore the patterns in the data effectively.

3 The Spatiotemporal Pattern Predicate

Consider the *Missed Approach* procedure in the Section 1. It can be described by three predicates: aircraft comes close to destination, aircraft descends till it is below a certain altitude, aircraft climbs up again. Temporally, the third predicate must be fulfilled after the second predicate, and both of them must be fulfilled during the fulfillment time of the first predicate.

The *STP Predicate* is proposed in [7]. It is based on the *abstract data types* model for moving objects in [6] [5]. The model defines the *moving* type constructor. It constructs the time-dependent counterpart of every static type. The moving types are denoted by prefixing m to the standard types (e.g., *mpoint*). We use the italic underline style all over the paper to denote types. The *mpoint*, for example, is represented as a temporally ordered list of *units* (*upoint*), each of which consists of a time interval and a line function. The coordinates of the *mpoint* at any time instant within the interval are obtained by evaluating the line function.

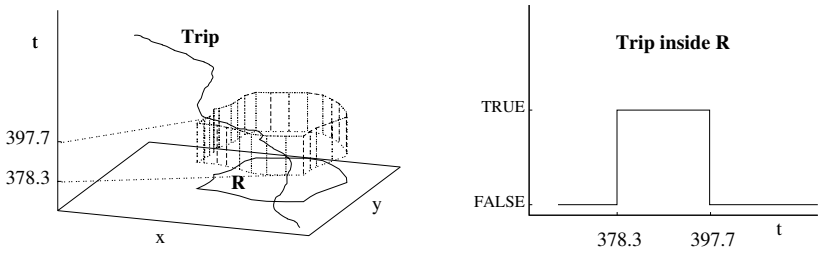

Figure 2: Time-dependent predicates

The STP predicate is a pair (P, C), where P is a set of *time-dependent* predicates, and C is a set of *temporal constraints* on the fulfillment times of the predicates in P. Time-dependent predicates were first introduced in [6]. Figure 2 illustrates an example. The *Trip* is an *mpoint* object, that crosses the *region* object R between the time instants 378.3 and 397.7. The evaluation of the time-dependent predicate *Trip inside R* is a time-dependent boolean (*mbool*) having the value *true* between these two time instants and *false* otherwise. A time-dependent predicate, hence, is an operation that yields an *mbool*. All the static predicates defined for non-moving types can uniformly and consistently be made applicable to the corresponding moving types as time-dependent predicates, see [6].

The temporal constraints specify the order of fulfillment of the time-dependent predicates. The *Missed Approach* procedure can be expressed as follows:

```
... stpattern[
  Close: distance(.Position, .Destination) < 5000.0,
  Down: ((.AltitudeDerivative < 0.0) and
    (.Altitude < 1000.0)),
  Up: .AltitudeDerivative > 0.0;
  stconstraint("Close", "Down", vec("abba","a.bba","baba")),
  stconstraint("Close", "Up", vec("abba","aba.b","abab")),
  stconstraint("Down", "Up", vec("aabb","aa.bb"))] ...
```

Here we show only the STP predicate (expressed in SECONDO using the *stpattern* operator) and omit the rest of the query for simplicity. We assume that it gets a tuple of type:

> *tuple*[Position: *mpoint*, Destination: *point*, Altitude: *mreal*, AltitudeDerivative: *mreal*],

where *Position* is the time-dependent (Lon, Lat) of the flight. The *stpattern* operator accepts two argument lists, separated by a semicolon. The first list contains three time-dependent predicates having the aliases *Close, Down,* and *Up*. The second list contains three temporal constraints, expressed by the *stconstraint* operator, each of which specifies a temporal relation between two time-dependent predicates.

Each of the terms/arguments of the *vec* operator specifies a relation between two time intervals. The start and the end time instants of the first interval are denoted *aa*, and those of the second interval are denoted *bb*. The order of the symbols describes the interval relationship visually. The dot symbol denotes the equality. For example, the relation *aa.bb* between the intervals i_1, i_2 denotes the order: $((i_1.t_1 < i_1.t_2) \wedge (i_1.t_2 = i_2.t_1) \wedge (i_2.t_1 < i_2.t_2))$. The temporal relation expressed by the *vec* operator is the disjunction of its components. Now we generalize this relation between two time intervals into a relation between two *mbool* objects.

Given $P = \{p_1, ..., p_m\}$ a set of time-dependent predicates, $C = \{c_1, ..., c_n\}$ a set of temporal constraints, and a tuple u. Let $p_i(u)$ denote the evaluation of p_i for the tuple u (i.e., $p_i(u)$ is of type *mbool*). Let $[p_i(u)]_j$ denote the j^{th} time interval during which $p_i(u)$ is true. The evaluation of the STP predicate (P, C) for the tuple u is true iff: $\exists j_1..j_m$ such that the set of time intervals $[p_1(u)]_{j_1}..[p_m(u)]_{j_m}$ fulfills all the temporal constraints $c \in C$, and we call $[p_1(u)]_{j_1}..[p_m(u)]_{j_m}$ a *supported assignment*. The STP predicate yields true iff at least one supported assignment is found.

Note that the *stpattern* predicate does not allow for constraints on the temporal difference between the starts/ends of the fulfillment times of the time dependent predicates. For example, it cannot express that the *Down* predicate in this query must be fulfilled for at least 2 minutes. Such conditions can be expressed using an extended version of the predicate called *stpatternex* [7]. Note also that both *sptattern* and *stpatternex* do not report the *supported assignments* in their output. The user, hence, cannot locate the parts of the trajectory that fulfill the pattern. Therefore, we have added four operators *stpatternextend, stpatternexextend, stpatternextendstream* and *stpatternexextendstream*. The first two operators report the first *supported assignment*, while the other two operators report all of them.

So far, we have shown that the STP predicates, and its variants, are very flexible, and can be used to express arbitrarily complex patterns. However, it remains tricky to decide the parameter values of the STP predicate, that best describe the intended pattern. In fact, STP queries are not trivial. They require fine tuning by checking the sensitivity of their results to the parameters. Moreover, real data is often incomplete or erroneous, the sampling rate might be low, and the user might lack a good description of the pattern in terms of time-dependent predicates. The integration with a visual analytics system allows for such fine tuning through user interaction. This is illustrated in more detail in the following section.

4 What Will Be Demonstrated

In this Demo we explore the *Missed Approach* and the *Stepwise Descent Landing* procedures, described in Section 1, using the proposed integration scheme. The typical use scenario for exploring patterns is: (1) in V-Analytics, select a subset of candidate trajectories, and transfer their ids to Secondo, (2) in Secondo, issue queries to locate the pattern instances (i.e., using the variants of the STP predicate), and to compute additional attributes, (3) transfer the query results to V-Analytics for visualization and analysis, and (4) if necessary, adjust the parameters and return to steps (1), (2) and (3).

4.1 Missed Approach

In this exploration task, we started in V-Analytics and selected the trajectories that landed at the end of their observed trips, to filter out noisy and incomplete data and to eliminate transit flights. In Secondo, we received the identifiers of this subset, and issued the query in Section 3 for *Missed*

Approach. After several cycles of adjusting the parameters, the final query that we have reached is:

```
...stpatternexextend[
  Close: distance(gk(.Position), gk(.Destination))< 15000,
  Down1: ((.AltitudeDerivative < 0.0) and
    (.Altitude < 600.0)),
  Up: .AltitudeDerivative > 0.0,
  Down2: .AltitudeDerivative < 0.0;
  stconstraint("Close", "Down1",
    vec("abba","a.bba","baba")),
  stconstraint("Close", "Up", vec("abba","aba.b","abab")),
  stconstraint("Down1", "Up", vec("aabb","aa.bb")),
  stconstraint("Up", "Down2", vec("aabb","aa.bb"));
  is_within_total_turn_range(.Pos atperiods theRange(
    end("Up"), inst(final(.Position)), TRUE, FALSE),
    50.0, 270.0)]
  filter[isdefined(.Close)] ...
```

Again we show only the part of the query that expresses the STP. The pattern is expressed, using the *stpatternexextend* operator, as a sequence of decreasing, increasing, then decreasing altitude, that all happen when the aircraft is close to the destination (i.e., within 15 km). The *gk* operator, that is used within the *Close* predicate, transforms the geographical coordinates (Lon, Lat) into the *Gauss Krüger* coordinate system, so that the *distance* operator is able to compute the Euclidean distance in meters. The aircraft is required to do a total turn between 50° and 270° after the *Up* event and before it lands. This is expressed using the user defined function *is_within_total_turn_range* which aggregates the turns of an *mpoint* object and asserts that the absolute total turn is within the given range. The trajectories/tuples that fulfill this pattern will be extended with the attributes *Close, Down1, Up,* and *Down2* storing the time periods during which the corresponding time-dependent predicates are fulfilled. The tuples that do not fulfill the pattern are extended with undefined values. These time periods are used in the rest of the query to compute other attributes for the visualization in V-Analytics.

Figure 3 illustrates several visualizations of the results. The *Trajectory Globe View* in (a) illustrates an aircraft coming from the top left corner of the figure, making a sharp dive in order to land, but then missing this first approach, climbing up again, making a cycle, and finally landing. The *Trajectory Globe View* is a tool developed by NASA, and integrated in V-Analytics. It visualizes the earth in 3D, and allows for flexible navigation.

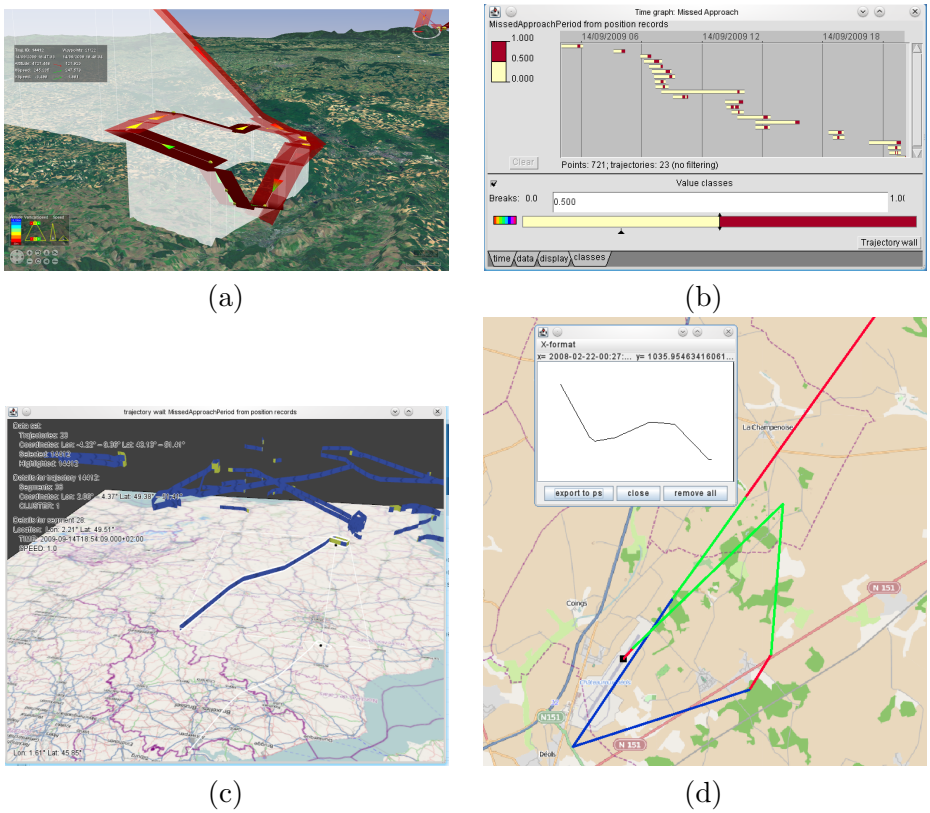

Figure 3: Missed Approach Query Results

Figure 3 (b) is a *Trajectory Time Graph* that shows the projection of the trajectory lifetimes, and highlights in red the parts where the *Missed Approach* occur. The *Trajectory Wall* in (c) is another 3D visualization tool from V-Analytics. The wall height corresponds to the trajectory lifetime. Finally the SECONDO view in (d) shows the altitude curve and the 2D trajectory projection of an aircraft that performed a *Missed Approach* procedure at the Châteauroux-Déols airport. The green color highlights the *Down1* and *Down2* events, and blue highlights the *Up* event. Note that the sharp turns in the flight trajectory are due to the relatively low sampling rate.

4.2 Stepwise-Descent Landing

A similar exploration procedure is used to explore the Stepwise-Descent Landing. The pattern is expressed using the *stpatternexextendstream* oper-

ator as follows:

```
...stpatternexextendstream[
    Dive1: .SecondAltitudeDerivative < 0.0,
    Lift: .SecondAltitudeDerivative >= 0.0,
    Dive2: .SecondAltitudeDerivative < 0.0  ;
    stconstraint("Dive1", "Lift", vec("aa.bb")),
    stconstraint("Lift", "Dive2", vec("aa.bb"));
    (end("Lift") - start("Lift")) > OneMinute ]
  filter[isdefined(.Dive1) and
    (AverageDiveAngle(.Alt atperiods .Lift) < 30.0)]...
```

It is expressed as a sequence of increasing (*Dive1*), decreasing (*Lift*), then again increasing rate of descent (*Dive2*). The *Lift* event is required to last for more than one minute, and the diving angle (i.e., the smallest angle between the altitude tangent and the horizontal) is less than 30°. That is, the aircraft is flying almost horizontal.

We attach a map (Figure 4 (a)) and a space-time cube (Figure 4 (b)) showing median positions of the horizontal (*Lift*) segments of the landing trajectories and 388 trajectories that own these events. The map shows interesting pattern: such landings occur frequently for flights to Paris from SW, W and NW, and sometimes from SE. The space time cube shows that such patterns occur in the morning and late afternoon around Paris, and during the day in Lyon and Nice. Figure 4 (c), (d) show individual results in the Trajectory Globe View, where one can clearly identify the sequence dive-lift-dive described in the query.

5 Conclusions

In this paper, we have demonstrated a novel work of integrating a visual analytics system and a moving objects database system. Now we have an integrated system that allows flexible query processing and is open for extensions. It is scalable and can be applied, in principle, to very large data sets. Experiments with larger data sets, and other kinds of operations (e.g., kNN and spatial join) are in preparation.

References

[1] SECONDO web site. http://dna.fernuni-hagen.de/secondo.html/.

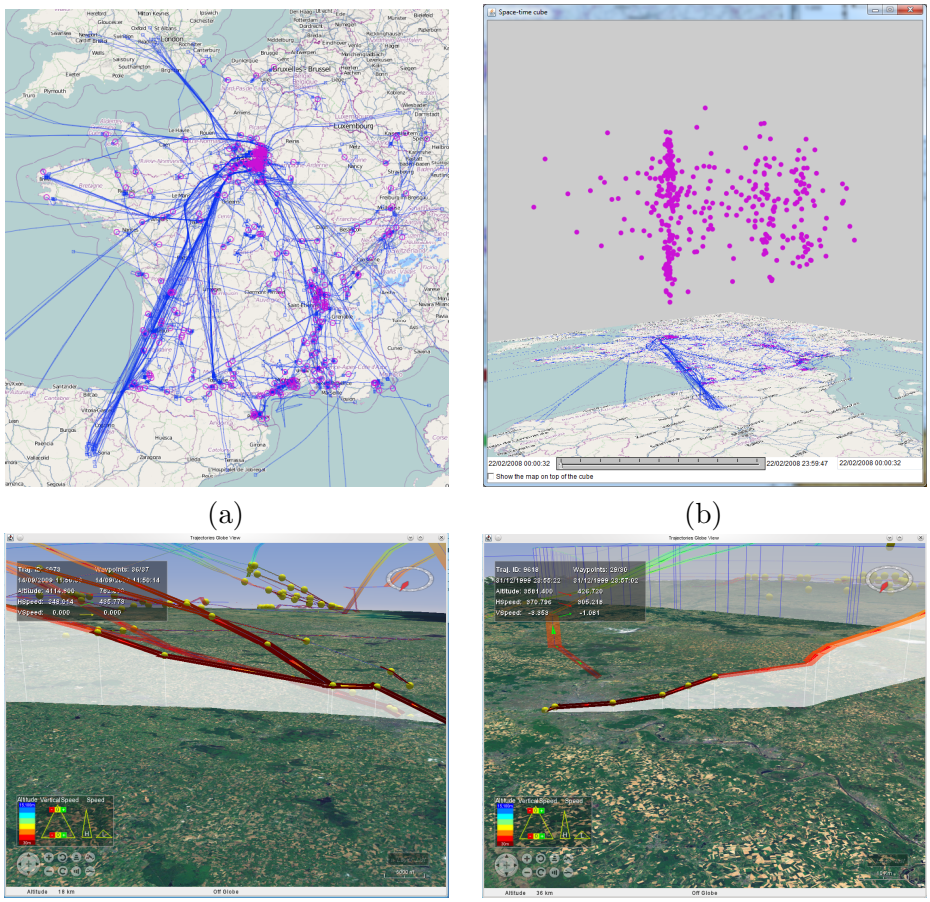

Figure 4: Stepwise-descent Query Results

[2] Adrienko, N., and Adrienko, G. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization and Computer Graphics 17* (February 2011), 205–219.

[3] Andrienko, G., Andrienko, N., and Wrobel, S. Visual analytics tools for analysis of movement data. *SIGKDD Explor. Newsl. 9* (December 2007), 38–46.

[4] Andrienko, G. L., Andrienko, N. V., Rinzivillo, S., Nanni, M., Pedreschi, D., and Giannotti, F. Interactive visual clustering of large collections of trajectories. In *IEEE VAST* (2009), pp. 3–10.

[5] Forlizzi, L., Güting, R. H., Nardelli, E., and Schneider, M. A data model and data structures for moving objects databases. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2000), ACM, pp. 319–330.

[6] Güting, R. H., Böhlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., and Vazirgiannis, M. A foundation for representing and querying moving objects. *ACM Trans. Database Syst. 25*, 1 (2000), 1–42.

[7] Sakr, M., and Güting, R. Spatiotemporal pattern queries. *GeoInformatica 15* (2011), 497–540.