Representation and Manipulation of Moving Points: An Extended Data Model for Location Estimation

Article *in* Cartography and Geographic Information Science · November 2000

DOI: 10.1559/152304099782330725 · Source: CiteSeer

4 authors, including:

José Moreira
University of Aveiro
36 PUBLICATIONS **297** CITATIONS

SEE PROFILE

Cristina Ribeiro
University of Porto
129 PUBLICATIONS **930** CITATIONS

SEE PROFILE

Jean-Marc Saglio
Institut Mines-Télécom
17 PUBLICATIONS **278** CITATIONS

SEE PROFILE

Representation and Manipulation of Moving Points: An Extended Data Model for Location Estimation

José Moreira

(jmoreira@uportu.pt)

Departamento de Informática da Universidade Portucalense

Rua Dr. António Bernardino de Almeida, 541-619

4150 Porto – PORTUGAL

(Assistant Professor and Scholarship Holder of Universidade Portucalense)

Cristina Ribeiro

(mcr@fe.up.pt)

Departamento de Engenharia Electrotécnica e de Computadores

Faculdade de Engenharia da Universidade do Porto

Rua dos Bragas

4099 Porto Cedex – PORTUGAL

Jean-Marc Saglio

(saglio@inf.enst.fr)

Département Informatique de l'Ecole Nationale Supérieure de Télécommunications

46, Rue Barrault

75013 Paris - FRANCE

Abstract

The fields of application of spatio-temporal systems, i.e., systems that must operate with time-varying spatial properties, are vast and heterogeneous. Since it would be difficult to treat such diversity as a whole, we introduce a classification for spatio-temporal systems based on the properties of the represented objects. Building on this classification, we also claim that features of some complex objects can be derived from those of simpler ones, suggesting an evolutionary approach, starting with the study of simple objects and progressing by enriching them with new features.

This paper focuses on the definition of a data model for representation of moving points. The model is based on the decomposition of the trajectory of moving points into sections. The movement within each section of a trajectory is described by a variability function. Since, for most systems, it is not possible to store the exact knowledge about the movement of a mobile, the answers to queries may be imprecise. We propose two additional approaches to deal with imprecision, the superset and the subset semantics, based on a maximum value for the variability function, and a smooth technique to integrate them in the model. Finally, we analyse some functional aspects of the implementation of the data model on a Relational Database Management System (RDBMS) and outline some directions for future research.

Keywords: Spatio-temporal systems; moving points; continuous change; Geographical Information Systems (GIS).

1 Introduction

Geographical information, commonly represented by data types such as points, lines and polygons, and the associated locations, is generally stored on proprietary GIS systems. These systems are however less effective than traditional Database Management Systems (DBMS) when it comes to non-dimensional data, often referred to as attribute data. Hence, some hybrid systems combine GIS features, taking advantage of their efficient spatial data access and graphic display capabilities, and traditional DBMS to handle attribute data. This approach typically suffers from poor integration of the two architectures and difficulties in maintaining integrity of spatial and attribute data. Spatial databases are mean to overcome this problem by storing all data, attribute and spatial, within a single data model.

None of these approaches considers another natural dimension, the temporal one. Geographical information is subject to change over time in two relevant aspects: the geometry or location of the represented objects, and the associated attributes. We are concentrating here on representing the evolution of the spatial features of objects. This contrasts with the main focus of research on temporal database systems, set on the modelling, representation and querying of temporal attribute data (Soo, 1991) (Tsotras and Kumar 1996).

There are few approaches dealing with the integration of spatial and temporal components, in spite of the vast fields of application of spatio-temporal systems (see Section 3 for a brief overview). The features and requirements of such systems depend on the kind of applications and are largely heterogeneous. For example, some systems must handle continuously changing properties while others must handle properties that change at discrete steps. Also, the properties that may change differ from one system to the other.

Till now, research on the representation of evolving data has focused on the study of discontinuous change, recording either isolated events or changes of state. In the former, the attribute values are valid only for a time instant. In the latter, attribute values are valid and constant within a time interval until an event produces a change of state. In contrast, continuously changing data is characterised by potentially having a different value at each instant; the assumption that the values of attributes within a time interval are constant is no longer acceptable. This particularity strongly influences the management of temporal data in this case.

In this paper, we introduce a classification of spatio-temporal applications based on the properties of the represented objects. A key distinction is made between continuous and discontinuous change. We specify a data model for the representation of mobile objects that can be modelled by points. As it is, in general, not possible to store exact knowledge on the movement of objects, i.e., all their possible states, we propose a semantics to cope with the uncertainty of the position of objects. We then analyse some functional aspects of the implementation of the data model on a Relational Database Management System (RDBMS).

The rest of this paper is organised as follows. Section 2 introduces a classification of spatio-temporal systems based on the properties of the represented objects. Section 3 presents an overview of relevant work in the domains of temporal and spatio-temporal systems. We do not include purely spatial systems in this overview, as the main point is the discussion of moving points, for which spatial size and shape are irrelevant. An excellent survey may be found in (Gutting 1994). Section 4 presents a data model for the representation of moving objects and an extended semantics to cope with imprecise answers in query operations. Finally, Section 5 discusses the implementation of the data model in a RDBMS pointing out some difficulties and suggesting further developments. Section 6 concludes and presents directions for future work.

2 Classification of spatio-temporal applications

A search for existing and potential applications of spatio-temporal systems shows that the application domains are vast and that systems may include objects having greatly differing properties. A classification would therefore be useful, not only to bring together applications with identical requirements, but also to identify the elementary components that must be handled by spatio-temporal systems, facilitating a modular development, and a better understanding of the extent of the issues that these systems must accomplish.

2.1 Properties of spatio-temporal objects

We propose a classification of spatio-temporal systems, based on the properties of the objects they represent, following two guidelines. On the one hand it considers the spatial properties that may change over time. On the other it considers how these changes occur.

The main time-varying properties specific to spatial objects are location, direction, size and shape. Location and direction are related with the movement of spatial objects. The location allows the system to capture the translation movement and the direction allows it to capture the rotation movement. The size and shape are relevant for the mutation of objects. The size allows the system to capture variations in scale, i.e., expansion and contraction of the objects, and the shape allows it to capture their deformations. The most complex case arises when mutation involves the aggregation or the fragmentation of complex objects, i.e., objects that are composed by other elementary objects. In this case, it is often necessary to handle functional links or relationships between the objects, allowing, for example, to answer questions about the exchange of components between objects.

In addition, spatio-temporal objects also have attributes, such as those that store descriptions or values for associated numeric measures. These may also change over time, but the analysis in this classification is restricted to spatial properties. The integration of attribute and spatial changes is analysed in (Langran 1990) and is summarised in Section 3.2.1.

The values of these spatial properties may change continuously or discontinuously over time (Yeh and Viémont 1992). When changes are continuous, the values of a property vary as a function of time and may be represented by a line with no discontinuity points (Figure 1). When changes are discontinuous the values of properties change instantaneously from one value to another. In this case, the management of changes is closely related to the management of versions studied in temporal databases. As it can be seen in section 3.1.2 the time line can be represented by time instants or time intervals. In the former, versions are associated with the recording of isolated events. Each version is valid for the time instant associated with it and between any two time instants the values of properties are undefined or null (Figure 2). In the latter, versions are associated to the recording of changes of state. Changes are caused by events and between two events, which define a time interval, it is guaranteed that the values are constant (Figure 3).

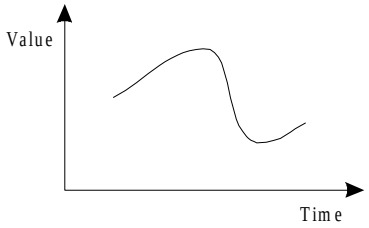

Figure 1: Continuous change

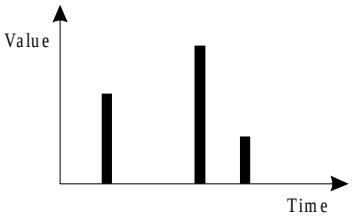

Figure 2: Event recording

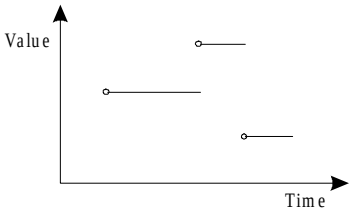

Figure 3: Change of state

Further discussion about models of time can be found in (Section 3.1).

2.2 Examples

Here after we illustrate how to classify applications into the categories proposed in the previous section. First of all, one must note that the complexity of applications may differ significantly (Table 1). For example, some applications must only deal with changes in one class of properties. This is the case of rigid moving objects, where the only properties relevant to the systems are the location and eventually the direction of the objects. The same is true for a system for monitoring the deformation, i.e., changes in shape and size, of an island due to the tides. Other applications must deal with changes in several classes of properties. For example, a system to monitor the evolution of pollution spots in the sea must be able to cope not only with the movement of the spots, but also with the changes on their extent (size and shape) and possibly with the fragmentation of one spot into several ones.

Continuous change	Rigid Moving Objects	Island	Pollution spots
Movement	√		√
Mutation:			
Simple deformation		√	√
Aggregation and fragmentation			√

Table 1: Classification of continuously changing spatio-temporal applications

The requirements of spatio-temporal representations are not found in GIS systems alone. The same need appears in other areas, and a suggestive example comes from multimedia applications. A multimedia system must be able to store and retrieve multimedia data efficiently and also to control the spatial and temporal relationships of data such as images, graphics, audio, video or animation (Vazirgiannis, Theodoridis and Sellis 1996) and (Nabil 1997). The MPEG-4 is an ISO-TEC standard to be released in November 1998 (Koenen 1997, Ed.), that allows the representation of individual units of audio and visual content called audio/visual objects (AVOs) and the composition of several of these AVOs to create audio-visual scenes. The manipulation of image sequences is envisioned with simple operations like choosing which objects will be displayed, and to specify additional parameters like their location, scale, displacement or rotation in a sequence.

A similar analysis can be carried out for systems dealing with discontinuous change (Table 2). For example, a shoreline will change over time due to the coastal erosion. To control these changes periodically it is required to record the deformations of the curves that define the shoreline. On the other hand, for an historical system studying the migration of populations it may be necessary to keep not only the movement of populations but also the changes in the territory occupied at a given historical period and even the possibility of aggregation and fragmentation of populations and territories.

Discontinuous changes	Shorelines	Migration of populations
Movement		√
Mutation:		
Simple deformation	√	√
Aggregation and fragmentation		√

Table 2: Classification of discontinuously changing spatio-temporal applications

The previous examples are merely indications, since the classes of properties that must be handled depend of the purpose of the systems. Indeed, even for the same application different levels of detail may induce different requirements. For example, a fire monitoring system for a governmental department should be able to represent the number, location and extents of burned areas, and to update this information, say, every three hours. In this case, the fire may be represented by polygons with discontinuous evolution over-time. However, a local entity responsible for the co-ordination of fire brigades, needs more detailed information to control the fire propagation. In addition to the local geography and meteorological conditions, it will need to decompose the fire on its foci and to cope with the evolution of each one individually. It will be necessary to control their movement, extent, and even the possibility of fusion and splitting of foci. This information should also be updated more frequently than the higher level one.

It should be noted that the choice between continuous and discontinuous change might be determined either by the nature of the system and the data to be modelled or by the ability or practical possibility of capturing or acquiring data with sufficient rigor. This distinction belongs to the modelling phase of the system and will not be pursued here.

3 Spatio-Temporal systems overview

Since time is a critical issue in the modelling of changes we start this section with a general outlook of recent works on temporal systems. We focus on temporal database research, which has largely studied this subject during past decade. An overview of pertinent works on spatio-temporal systems follows, emphasising the distinction between systems dealing with continuous changes and systems dealing with discontinuous changes.

3.1 Temporal databases

Almost all information systems involve the management of time-varying data. However, traditional DBMS, often referred to as non-temporal DBMS, capture a single snapshot of reality and are insufficient for those applications that require the support of past, current or even future data.

A temporal DBMS should allow the formulation of sophisticated queries over time, modifications to previous states (if an error is detected or if more information becomes available), and to future states (for planning purposes), using simple SQL-like queries. Moreover, efficient temporal access methods should be used, to improve the performance of computing intensive temporal queries.

Over the past decade several proposals for extending the database theory to incorporate temporal information have appeared. An excellent survey on temporal databases theory can be found in (Zaniolo et al. 1997) where many of the topics presented in this section are described in more detail. The definitions for the terms in *Italics* follow the Consensus Glossary of Temporal Database Concepts (Jensen et al. 1994).

3.1.1 Models of time

Temporal representations in an Information System can adopt several models of time. For example, the linear model for processes where time advances step by step from the past into the future, the cyclic model for recurrent processes like seasons, months or weeks, or the branching model to represent time dependencies between entities.

These models of time can be discrete or continuous. *Discrete models* consider the structure of points in time isomorphic to the natural numbers, hence, each point in time has a single successor. Each natural number corresponds to a *chronon*, which is the smallest non-decomposable unit of time that can be represented in the model. *Continuous models* adopt a time structure isomorphic to the real numbers and thus, contain no gaps. Each real number corresponds to a point in time.

Although time is perceived by most to be continuous, several practical arguments suggest the use of discrete models. First of all, measures of time have limited precision. Most language references to time are also compatible with the discrete model and, above all, only finite sets can be stored and manipulated in computers.

3.1.2 Temporal Data types and dimensions

The "time values associated with an object, e.g., an attribute value or a tuple" are referred to as *timestamps*. Several types of timestamps may be defined with respect to any single time dimension. The basic one, a *time instant,* is a "time point on an underlying time axis". A *time interval* is the "time between two instants" and may be represented by a set of contiguous chronons. A *temporal element* is a "finite union of n-dimensional time intervals".

Two time dimensions are of general interest in the context of databases: valid time and transaction-time. The *valid time* of a fact is the "time when the fact is true in the modelled reality; it is typically supplied by the user". The *transaction time* of a fact is the "time when a fact is current in the database and may be retrieved; transaction times are established according to the serialisation order of the transactions; transaction-time values cannot be later than the current transaction time; it is also impossible to change the past, and therefore transaction times cannot be changed". Though these two dimensions may have some application-dependent correlation they are generally considered orthogonal.

3.1.3 Temporal data models

There are now several relational and object-oriented temporal data models, with different representations of time, dimensionality and temporal structures. References to these data models can be found in (Özsoyoglu and Snodgrass 1995) and (Zaniolo et al. 1997).

Building on these two time dimensions, four different temporal models of relations can be identified. *Snapshot relations* are "relations of a conventional database system incorporating neither valid-time nor transaction-time timestamps". They capture only a single snapshot of the modelled reality and associated databases. A *valid-time relation* is a "relation with exactly one system supported valid time". It captures the past, present and possibly the future behaviour of an information system as currently known. When a correction is made the previous values are not retained. It allows the system to answer questions of the kind: “What was the salary of Sandra in January 1998?”. *A transaction-time* relation is a "relation with exactly one system supported transaction time”. It allows answering questions of the kind: “Has a change in Sandra's salary been registered in the database after January 1998?”. A *bitemporal relation* is a "relation with exactly one system supported valid time and exactly one system supported transaction time" and allows answering to questions of the kind: “What was the salary of Sandra in January 1998 as known in February 1998?”. As for valid-time relations and transaction-time

relations, there are no restrictions as to how either of these temporal dimensions may be incorporated into the tuples."

Two different approaches have been followed for incorporating time according to these data models (Clifford, Croker and Tuzhilin 1994). The *temporally ungrouped model*, also referred to as tuple timestamping or first-normal-form (1NF) model, uses a separate tuple carrying a timestamp to represent each moment of time relevant to a fact. In the *temporally grouped model* also referred to as attribute time stamping or as non-first-normal-form (N1NF) model, the domain of each attribute is extended from simple values to complex values that incorporate the time dimension. Related events are grouped and represented in a single tuple.

3.1.4 Temporal Query Languages

Building on the diversity of data models, many temporal query languages have been proposed. A comparative overview of the major relational and object-oriented temporal query languages proposed to date can be found in (Zaniolo et al. 1997). Based on these languages a consensus temporal extension to the SQL-92 language standard designated Temporal Structured Query Language, or TSQL2 (Snodgrass and al. 1994) has been recently proposed. This proposal is currently part of the evolving SQL3 draft.

TSQL2 has a simple underlying data model, termed the Bitemporal Conceptual data model (BCDM). It is a temporally ungrouped model that timestamps tuples with bitemporal elements, i. e., sets of bitemporal chronons.

TSQL2 is a superset of SQL-92, supporting all functionality required by conventional relations. Additionally, it supports state relations, timestamped with temporal elements, which are sets of periods, and event relations timestamped with sets of instants. Several language constructs were included to facilitate the manipulation of temporal data. For example, TSQL2 performs automatic coalescing of tuples, i.e., two or more value-equivalent tuples with consecutive or overlapping timestamps are replaced by a single tuple with an interval valued timestamp which is the union of the timestamps of the original tuples. If a relation has transaction-time support, the schema is versioned and the relation may be accessed and modified through previous schemas. It supports valid-time projections and allows the specification of periods of validity.

In addition to functional aspects, performance is another important issue in the context of databases. Until now, several access methods for temporal data have been proposed, but few of them have been implemented. The difficulties in accessing temporal data and the different techniques that have been used to solve them are described in (Salzberg and Tsotras 1997). A consensus test suite of temporal queries has also been defined (Jensen 1993, Ed.).

3.2 Spatio-temporal systems

Spatio-temporal systems deal with time-varying spatial data. Applications, in domains so diverse as earth sciences, economical and social-economical studies, urban planning, traffic control, land information systems, environment protection or medical imaging, can benefit from the integration of spatial and temporal components. These two components have been mostly treated as independent domains of research and a simple merge of spatial and temporal theories is not sufficient to embrace all the new issues raised by their integration. Current research follows two main directions. The first one has to do with those applications where features values change from one state to another instantaneously. The other one concerns the applications where the values of features change continuously over time. The following sections present a brief overview of research in these areas.

3.2.1 Discontinuous changes

The representation of discontinuous changes of non-spatial data commonly referred to as attribute data, has already been studied in temporal databases domain. There are now temporal data models, efficient time-stamping techniques and query languages to manage different states, often named versions, of attribute data. However, many new problems arise when one tries to bring together spatial and temporal data. Early work on this area started with the study of individual systems and their own specific problems. Some data models and structures have been proposed, some principles have been enumerated and many problems, difficulties or obstacles of design and implementation have been identified.

The representation of spatial objects is usually a combination of geometric and attribute data. The geometric part has often a complex structure, i.e., it is composed by several primitive geometric components also referred to as elements. For example, shorelines are represented by sets of line segments. Hence, for such objects, two types of change are possible: attribute change and geometric change (Langran 1990). In the former, the whole or part of a feature undergoes attribute change. In the latter, the entire feature can be moved or deleted, a new element can be added to a feature, an existing element can be moved or deleted in a feature, or part of a feature can be reshaped or deleted. However, continual updates of geometry can cause edge effects and introduce artefacts that cause data to appear piecemeal or incorrect. If only a local area is altered, the borders of that area may no longer match the surrounding unaltered area. This problem and the representation of changes as chains of versions are discussed in (Langran 1993).

Another issue concerns the techniques for time-stamping spatial objects with complex geometry (Worboys 1994). The most intuitive and inexpensive approach is to timestamp the entire object associating a temporal value at its birth and another temporal value at its death. However, it is obvious that this solution has only a limited expressiveness for the temporal properties of the object throughout its life. To obtain more control over the granularity at which time is associated with spatial objects it is necessary to timestamp their referenced primitive geometric components. Two solutions are suggested.

One consists in merging time and space at the primitive geometric component level (point, line segment or polygon). The other consists in merging time and space at the lowest spatial level by forming a composite class of spatial point and a temporal point. Then, this class is used to construct other primitive spatio-temporal object classes of higher dimensions.

Some difficulties arise when we want to handle object identities for applications that must control the aggregation and fragmentation of spatial objects. To cope with these problems two kinds of change can be defined: constructive changes and non-constructive changes (Roshannejad and Kainz 1995). Non-constructive changes pose no problems and it is sufficient to give a unique identifier to each object along the time line. The history of constructive changes is kept using an object-oriented data model based on a class identity. It is assigned an identity number to each version of an object and maintained a double linked list with pointers to the next and previous identity numbers. Aggregation, association and multiple inheritance are built-into the object-oriented modelling.

In (Claramunt and Theriault, 1995) it is proposed a spatio-temporal framework that revisits the versioning concept of previous works. It uses an event-oriented data model defined on three time periods, past, present and future, and partitions temporal, spatial and attribute data on different structures. The model is able to handle the evolution of a single entity (appearance, disappearance, transformation and movement), the functional relationships between entities (succession and permutation) and the evolution of spatial structures involving several entities (split, union and reallocation of objects).

3.2.2 Continuous changes

Temporal databases do not provide any support for representing information about objects that continuously change over time. This feature is useful not only to some GIS applications that must handle the displacement, rotation or deformation of objects, but also to other systems dealing with continuously changing attributes, such as temperature, atmospheric pressure or salinity of sea water.

Time in continuous models is isomorphic to the real numbers, hence, involve the manipulation of infinite sets impossible to store or manipulate directly by computers. Thought the instruments to measure time have finite precision and it is therefore reasonable to define a temporal resolution that matches the precision of the measures of time, this may be insufficient. For finer resolutions this approach would still involve, in most of the cases, data sets excessively large to be managed by computers. The simple solution of decreasing the temporal resolution could lead to unacceptable losses of information.

Up to the present, as far we know, this subject has only been presented in very few and recent papers. (Sistla et al. 1997) and (Sistla et al. 1998) propose the Moving Objects Spatio-Temporal (MOST) data model and the Future Temporal Logic (FTL) language for systems dealing with attributes that change continuously over time. In this model it is assumed that attributes change over time according to a given linear function, the motion vector. This motion vector is obtained from the minimum and

maximum bounds of displacement of the moving object. It is also assumed that it is possible to automatically update the motion vector when a change in speed or direction is sensed. The FTL query language enables queries pertaining to future states of the system being modelled. The answer to these queries is tentative, i.e., it should be regarded as correct according to what is currently known about the real world, but this knowledge can change. In consequence, two kinds of semantics are provided: May and Must. Under the May semantics, the answer is the set of all objects that possibly match the query. Under the Must semantics the answer to the query is the set of objects that definitely match the predicate. This model does not store the database history anywhere.

(Yeh and Cambray 1994) proposes a data model to represent highly variable numerical data, i.e., data characterised by a potentially different value at each quantum of time (chronon). The evolution of data is represented by a limited number of values of the type *(t, v, vv)*, where *t* is a time interval, *v* is the value associated to the initial state of an attribute on that time interval and *vv* is a behavioural function that describe how the value evolves. A behavioural function is either a punctual function, a step function, a linear function or an interpolation function. (Yeh and Cambray 1995) introduces an extension of the previous model to manage highly variable spatial data in a GIS.

4 Moving Points Data Model

4.1 An evolutionary approach

Building on the classification and examples of (Section 2) we argue that the features of complex objects can be derived from those of simpler ones. For example, it is possible to compose a representation for moving objects with continuous changing shape, like the pollution spots, by combining the movement and simple deformation properties. Having this in mind, we are following an evolutionary approach, starting with the study of simple objects and progressing by enriching them with new features.

We start with the study of moving points on an n-dimensional space. They constitute the simplest class of moving objects and their field of application is vast. There are many systems, like those dealing with the position of cars, ships or planes, which only need to keep the position of the objects. Their shape and size are not relevant in the scale of the maps that are used.

Yet, we expect that the extension from moving points to rigid moving objects, i.e., moving objects with fixed shape and size, will not constitute a hard problem. The movement is still the main feature; geometric operations will be a little harder to compute, since the trajectories are represented by a polygon rather than a line (indeed the polygon can be seen as a line with the same width as the object). We are trusting that the semantics of operations used here for moving points will require little changes to handle the extent of the objects. These objects can be enriched later with new features to cope with

movement and deformation simultaneously and finally with features to handle the aggregation and fragmentation of moving objects.

4.2 Characterisation of moving points

Before advancing to the definition of a data model for moving points, it is interesting to take a closer look to their features. Starting with the most important, the movement, two types of moving objects may be identified: those having a free trajectory and those with a constrained trajectory. *Free trajectory* means that there are few or no restrictions on the movement of a point in an n-dimensional space. This is, for example, the case of a ship moving on the ocean. *Constrained trajectory* means that the movement of an object in space is strongly restricted. A prime example is the trajectory of trains. In this case, the complexity of the problem can be reduced to the representation of a point moving on a one-dimensional space. Thus, constrained trajectory can be regarded as a specialisation of the general case, the free trajectory, where some simplifications or optimisations can be accomplished.

Another issue concerns the methods for acquisition of data. On the one hand, there are *event driven systems*, where one can assume that it is possible to automatically detect changes in the speed or direction of a mobile. On the other hand there are sensor systems, such as the Global Positioning System (GPS), that capture the location data in an ordered sequence at regular intervals of time. We designate these systems by *observation driven systems*. These observations can be synchronised, if all mobiles are observed at the same instant, or independent. It should also be considered the possibility of having several sources of information. In this case, the database system must be able to control unordered sequences of observations.

In the following section we present a data model allowing the representation of the history of moving objects. This model is able to cope with points moving on an n-dimensional space. The observations can be independent or synchronised for all objects, but, for each object, the observations must be accomplished at regular time intervals and be entered as ordered sequences.

The transaction time, i.e. the time when observation reports are received and stored, is not taken into account. We only assume that transaction timestamps are in the same order as valid timestamps, i.e. observation timestamps.

4.3 The model

The output of a sensor system is an ordered sequence of states represented as a set of triplets *(Pid, t, ov)*, that we designate by Observations Base, where *Pid* is the identification of the object represented by a point, *t* is a timestamp and *ov* is an n-dimensional value corresponding to the location of the object.

It is important to point out that this information is captured whether or not the position of an object has changed between consecutive observations. Also, one may expect that most real objects have a

"smooth" movement and therefore, the speed in a sequence of consecutive observations will often be approximately constant.

These observations are then translated into movement analytical formulas and stored in a Movement Knowledge Base structure similar to the one defined on (Yeh and Cambray 1994). This structure is described by a sequence of values *(Pid, t_0, t_1, v, vv)*, where *Pid* is the identification of the object, *[t_0, t_1]* is the valid time interval for the variability function *vv*, and *v* is an n-dimensional value corresponding to the initial state, i.e., the location of the object, at t_0. In this structure, each tuple represents a section of a trajectory, i.e., contains the complete information not only about the location of an object, but also about its movement within a valid time interval. Moreover, it allows the aggregation of consecutive time intervals with the same or very similar functions of variability. Consequently, the number of tuples required for representation of the movement of objects decreases, reducing the storage space requirements as well.

$$\tilde{v} = \bar{v} + vv(t - \bar{t})$$

(Equation 1): Regression line of *v* on *t*

$$vv = \frac{\sum_{i=1}^{n}(v_i - \bar{v})(t_i - \bar{t})}{\sum_{i=1}^{n}(v_i - \bar{v})^2}$$

(Equation 2): Variability factor

We use a linear approximation function to describe the movement within a valid time interval (Equation 1). This equation represents the regression line of *v* on *t* that minimises the sum of square deviations of the observed values from the line. The variable $\tilde{v}$ is called the predicted or estimated value of *ov* for any corresponding value of *t*, and $(ov - \tilde{v})$ is the error of prediction or estimation (Figure 4). The variables $\bar{v}$ and $\bar{t}$ are respectively the averages of *ov* and *t* of *n* observed values. The variability factor is given by *vv* (Equation 2).

Any other value *v'* with valid time *t'* being in the valid time interval *[t_0, t_1]* can be calculated using the function $v' = v + vv(t' - t_0)$.

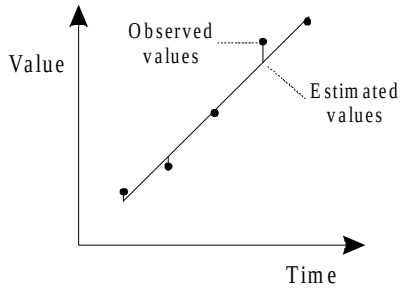

Figure 4: Linear interpolation

For each observation *i* in a valid time interval the approximation must obey the following formula: $\Delta v_i = |(ov_i - \tilde{v}_i)| <= \xi$, where ξ is an user defined precision. So, when a new observation is added, the formulas are recalculated and, if there is a $\Delta v_i > \xi$, the formerly valid time interval is closed and a new

tuple is created. ξ represents the quality of estimation and should be dictated according to the spatial resolution requirements of the application . The aggregation rate strongly depends on ξ.

It is important to point out that the precision of the whole system depends on two factors. One is the precision of the measurement instruments (sensor systems) and the other is the precision of the model. No practical reason justifies having a model with a precision greater than the precision of the sensor systems. As an example, considering a sensor system that yield spatial information in units of miles, and using this unit to store spatial data in the model, it is sufficient to set ξ to *0.5* miles to obtain answers in exactly the same unit as if no interpolation were used. Setting ξ to greater values, for example one mile, allows increasing the probability of aggregation of intervals, but the user should be aware that some discrepancies might arise (Figure 5) and (Figure 6).

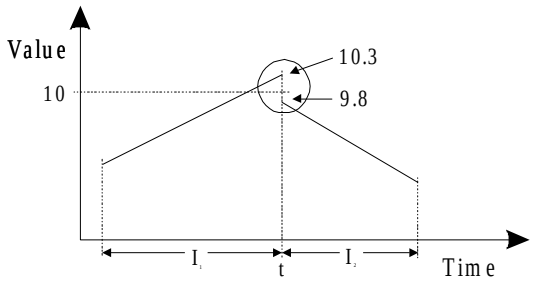

Figure 5: Discontinuity between two consecutive intervals for $\xi = 0.5$

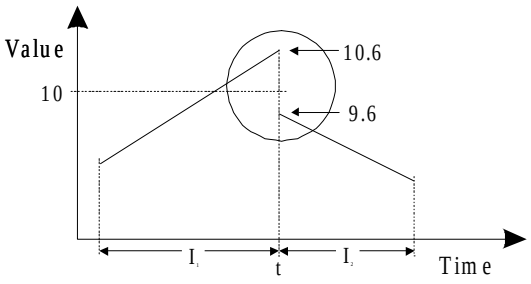

Figure 6: Discontinuity between two consecutive intervals for $\xi = 1$

In the first case, where ξ is equal to *0.5*, the answer to a query about the position of the object at time instant *t*, using either I_1 or I_2 is exactly the same value obtained by the sensor system, since the values are rounded to the unit, in this case *10*. However, in the second case, where ξ is equal to *1*, it may happen that rounding the values obtained for the position of a mobile at time instant *t* using I_1 gives the value *11* and using I_2 gives the value *10*. Nevertheless, these two values are within the precision defined by the user.

The number of decimal places that must be used to store the estimated values can be calculated using the formula $Ceiling(Log_{10}(\xi/2))$, where *Ceiling* is a function that rounds a number up to the nearest integer; and $Ceiling(Log_{10}(2*\Delta t_{max}/\xi))$ to calculate the number of decimal places for the values of the variability function, where Δt_{max} is the maximum amplitude for time intervals. Δt_{max} is a user-defined parameter and the **aggregation rule** can be at last defined as:

A new observation *(Pid, t, ov)* is aggregated to the last section of the trajectory of a moving point *(Pid, t_0, t_1, v, vv)*, if and only if, $(t-t_0) < \Delta t_{max}$ and for any observation of *Pid* within $[t_0,t_1]$, $|ov_i - \tilde{v}_i| < \xi$.

Till now the discussion has been focussed on values defined on a one-dimensional space. However, the previous model can be extended to cope with values defined on two- or higher- dimensional spaces. The new Movement Knowledge base becomes $\{(Pid, t_0, t_1, v, vv_1, vv_2, \ldots, vv_n)\}$. In this structure, *v* in an n-dimensional value $(x_1, x_2, \ldots, x_n)$. The variables $x_1, x_2, \ldots, x_n$ are independent and consequently each one has its own function of variability $vv_1, vv_2, \ldots, vv_n$ respectively. It is a temporally ungrouped data

model (Clifford, Croker and Tuzhilin 1994) and therefore, for any new observation if any of the variables $x_1, x_2, \ldots, x_n$ does not fulfil the aggregation rule a new tuple is created. As for the user defined precision value, it may be a single ξ to all variables or there may be an individual $\xi = \xi_1, \xi_2, \ldots, \xi_n$ for each of $x_1, x_2, \ldots, x_n$.

4.4 Semantics

Sensor systems provide data about the location of objects at regular time intervals and thus the exact behaviour of objects between two observations is unknown. The data model defined in the previous section assumes that their movement between two observations is linear and uniform. However, there are some cases where this constraint cannot be applied. For example, when there is a spread of toxic wastes in the sea, the authorities would like to query a nautical surveillance system to know which ships have crossed the polluted zone for a specified time interval. Let's consider that the polluter has followed the trajectory represented in (Figure 7). The black dots represent the four observations made during that period of time, the shaded rectangle represents the polluted area and the hatched line is the linear approximation of the movement of the ship stored in the Movement Knowledge Base.

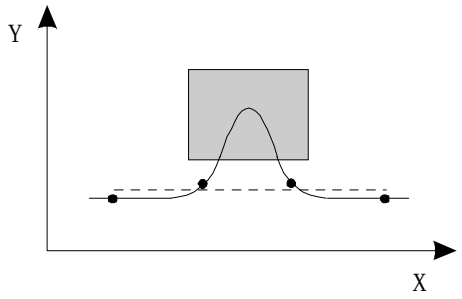

Figure 7: Indeterminacy of the behaviour of an object between consecutive observations

It can be seen that the hatched line does not cross the shaded region and thus, the answer to the query would not include the real responsible for the spread of waste. The opposite may also occur, since it is possible that the answer to the previous query includes some false candidates for which the approximated trajectory stored in the Movement Knowledge Base crosses the area but they actually did not.

So, the result of a query using the previous model is a set of possible candidates, that may include some false tuples corresponding to objects that do not actually conform the intended predicate and may omit some tuples corresponding to some objects that do conform to it. To deal with this problem it is possible to define two new kinds of semantics for query operations: Superset and Subset. Under the Superset semantics the result of a query is the set of all tuples that possibly conform to a given predicate. This is a superset of the real result. Under the Subset semantics the result of a query is the set of tuples that definitely conform the predicate. This is a subset of the intended result.

These two semantics are grounded in a maximum value of the variability function per unit of time. To start with, let's consider an object, defined on a two-dimensional space, moving from point *P1* to *P2*

during a given time interval and say that the maximum speed of the object is 20 units of space per unit of time. Using the previous model, it is assumed that the object follows the trajectory defined by the line connecting *P1* and *P2* with a linear uniform movement. If the distance between *P1* and *P2* is also 20 then it is clear that the object has moved at the maximum speed and following a straight line between the two points (Figure 8). On the other hand, if the two points are coincident it is probable that the object stayed immobile. However, this is not always the case because it is also possible that the object has followed a trajectory like the curve in (Figure 10). In the worst case, it is possible that the object moved at maximum speed from *P1* to a point *P'* during a period of half of the time interval and then went back in the opposite direction at maximum speed as well. Since it could have followed any direction, the minimal region comprising all the possible trajectories is defined by a circle with centre on *P1* and radius is equal to $vv_{max}/2$.

For the intermediate cases, let's consider that the distance between *P1* and *P2* is 10 (Figure 9). The line between *P1* and *P2* defines the probable trajectory of the object. The worst case, i.e., the case where the real trajectory can diverge as much as possible from the probable trajectory, is when the object moves at maximum speed. The most distant points that can be reached by the object are those defined by the ellipsis given by the equation $x^2/a^2+y^2/b^2=1$, being $a=vv_{max}/2$, $c=(P2-P1)/2$ and $b^2=a^2-c^2$.

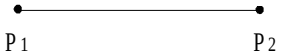

Figure 8: When the distance between the points is maximal

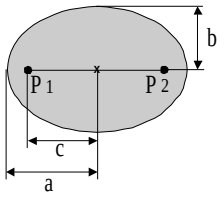

Figure 9: General case

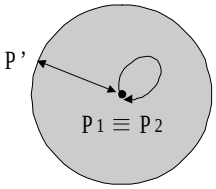

Figure 10: When the two points are coincident

Coverage of all possible trajectories

These three possibilities can be reduced to a single one, the ellipsis approximation. In the first case, where the speed of the object is maximum, a line segment is obtained, because $b = 0$ and $a = c$. In the second case, where *P1* and *P2* coincide, a circle is obtained because $c = 0$ and $a = b$. Therefore, it is possible to state that the uncertainty about the trajectory depends on the ratio of its velocity and the maximum velocity. Also, the imprecision is given by an ellipsis ranging from a circle (maximal imprecision) into a line segment (no imprecision at all).

To ensure that an object has surely crossed a region it is sufficient that at least one of the observations has been performed when the object was within that region. This can be done easily with the model since it is known the time instant of all observations and there is a function allowing the calculation the position of an object at a given moment. Moreover, even if the object was not within a region at the moment of any observation there are certain cases that one can ensure that an object crossed a region. This is the case of the object in (Figure 11) where though none of the observations took place when the

object was within the region, there is not any possible trajectory completely outside that region. In conclusion, to ensure that a moving object has crossed a region during a period of time it is sufficient that, one of the observations took place when the object was within that region or that all possible curves joining the two points and contained in the ellipsis intersect the region.

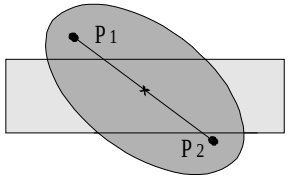

Figure 11: All possible trajectories between the two points cross the rectangle.

The transposition of these semantics to other dimensional spaces is straightforward. In one-dimensional spaces the ellipsis is converted into a line segment and in three-dimensional spaces the ellipsis is converted into an ellipsoid.

These Superset and Subset semantics allow different kinds of answers to queries. The Subset semantics allows identifying the set of tuples associated to the objects that surely match a predicate. The superset semantics allows identifying the set of tuples associated to all objects that possibly match a query predicate, and its complement identifies the set of objects that surely do not match that query predicate. The difference between the two sets gives the sets of objects for which one can not assert if they match the query or not.

4.5 Extended model

For simplicity, this discussion will be presented for objects represented on a two-dimensional space.

In the data model, a valid time interval may include several observations, hence the definition of a region according to the Superset and Subset semantics engages several ellipsis (Figure 12). Since the elapsed time between consecutive observations is constant, all ellipsis are equal.

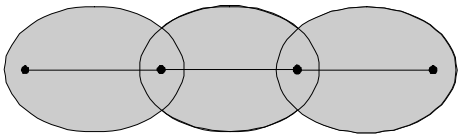

Figure 12

It is however not easy in practice to work with a set of such geometry to answer queries. To avoid this one can use as approximation a Minimum Bounding Rectangle (MBR) enclosing the set of ellipsis (Figure 13). It is easier to represent and use, and the enlargement caused by the extra approximation in the area covered by the rectangle is not significant. The Superset and Subset semantics remain valid using MBR as approximation.

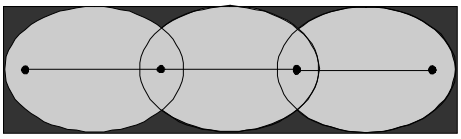

Figure 13

To cope with the Superset and Subset semantics and the use of approximations it is necessary to change the data model. The new Movement Knowledge Base is composed by tuples of the type *(Pid, t_0, t_1, v, n, vv_1, vv_2, b_1, b_2)*. Like the previous model, *Pid*, *[t_0, t_1]* and *v*, are respectively the identification of the object, the valid time interval, and a two-dimensional value *(x_1, x_2)* representing the position at the beginning of the time interval; vv_1 and vv_2 are the variability functions associated to each of the dimensions in that time interval. The additional attributes are *n*, representing the number of observations in the time interval, and b_1 and b_2 representing, respectively, distances (*a-c)* and *b* in (Figure 10).

The attribute *n* could be omitted since the interval between observations is regular and can therefore be calculated. It is included only for simplicity and verification purposes; it allows, for example, the detection of missing observations due to unavailability of the sensor system.

The general structure of the Movement Base for objects moving on n-dimensional spaces is of the kind *(Pid, t_0, t_1, v, n, vv_1, vv_2, …, vv_n, b_1, b_2, …, b_n)*.

5 Implementation on a RDBMS

This section presents a functional analysis on the implementation of the data model defined in (Section 4.5) and considers objects moving on a two-dimensional space. It focuses on the implementation of an algorithm for insertion of new observations into the knowledge base, as well as in the identification and specification of spatial, temporal and spatio-temporal query operations.

5.1 Data structures

The previous data model can be directly implemented on an RDBMS but some practical reasons justify some slight changes. The structure of the Observations Base is exactly the same in the model *(pid, t, x, y)*. However, the structure of the Movement Base becomes *(pid, t_0, t_1, t_m, n, x_m, y_m, vx, vy, bx, by)*. In this structure t_0 and t_1 are respectively the timestamps of the first and last observations in the time interval; x_m and y_m, the location of the object at t_m; finally the attribute t_m, represents the average of the timestamps of the observations in the interval. With these new attributes the values $\bar{x}$, $\bar{y}$ and $\bar{t}$ (Equation 2) can be easily calculated using the formulas $\bar{x} = (x_m * n + ox)/(n+1)$, $\bar{y} = (y_m * n + oy)/(n+1)$ and $\bar{t} = (t_m * n + ot)/(n+1)$ respectively, where *ox*, *oy* and *ot* are the values of a new observation. The pseudo-code for inserting new observations follows:

```
InsertObservation(opid, ot, ox, oy, precision, vmax)
FETCH last tuple FROM MvtTable WHERE pid=opid INTO mvt;
IF mvt.t1 < ot THEN                                          /* TimeSt. of new obs. is valid?*/
    INSERT INTO ObsTable VALUES (opid, ot, ox, oy);          /* Update the observations table */
    IF mvt is Null THEN                                      /* No previous obs. for opid */
        INSERT INTO MvtTable VALUES (opid, ot, ot, ot, 1, ox, oy, null, null, null, null);
    ELSE
        lx = Mvt.xm + Mvt.vx*(Mvt.t1-Mvt.tm);                /* Last x position */
        ly = Mvt.ym + Mvt.vy*(Mvt.t1-Mvt.tm);                /* Last y position */
        Mvt.tm = (Mvt.n * Mvt.tm + ot) / (Mvt.n + 1);        /* Compute new tm */
        Mvt.xm = (Mvt.n * Mvt.xm + ox) / (Mvt.n + 1);        /* Compute new xm */
        Mvt.ym = (Mvt.n * Mvt.ym + oy) / (Mvt.n + 1);        /* Compute new ym */
        SELECT SUM((t-Mvt.tm)*(x-Mvt.xm))/SUM((t-Mvt.tm)**2),   /* (Equation 2) */
               SUM((t-Mvt.tm)*(y-Mvt.ym))/SUM((t-Mvt.tm)**2)
            FROM ObsTable WHERE pid=opid INTO newvx, newvy;
        SELECT MAX(ABS((x-Mvt.xm)-bx*(t-Mvt.tm))),           /* Maximum error prediction in x,y */
               MAX(ABS((y-Mvt.ym)-by*(t-Mvt.tm)))
            FROM ObsTable WHERE pid=opid INTO devx, devy;
        IF devx > precision OR devy > precision THEN         /* New tuple */
            newbx = (vmax-(ox-lx)/(ot-Mvt.t1))/2;
            newby = SQRT(vmax**2-(ox-lx)/(ot-Mvt.t1)**2)/2;
            DELETE FROM ObsTable WHERE pid = opid AND t<Mvt.t1   /* Discard useless observations */
            INSERT INTO MvtTable VALUES (opid, Mvt.t1, ot, (Mvt.t1+ot)/2, 1,(Mvt.xm+ox)/2,
                (Mvt.ym+oy)/2, (ox-lx)/(ot-Mvt.t1), (oy-ly)/(ot-Mvt.t1), newbx, newby));
        ELSE                                                 /* Expand time interval */
            newbx = (vmax – newvx)/2;
            newby = SQRT(vmax**2-newvx**2)/2;
            UPDATE MvtTable SET t1=ot, tm=Mvt.tm, n=n+1, xm=Mvt.xm, ym=Mvt.ym, vx=newvx,
                vy=newvy, bx=newbx, by=newby WHERE pid=opid and t0=mvt.t0;
        END IF;
    END IF;
END IF;
```

Vmax represents the maximum speed of the object. The *ObsTable* refers to the Observations Base and holds the observations included in the last time interval of each object stored in the Movement Knowledge Base identified in the algorithm by *MvtTable*. Whenever a new observation cannot be aggregated to the last tuple of *Pid* in the *MvtTable*, all the tuples, except the last one, referring to *Pid* in *ObsTable* are deleted. For simplicity, we do not consider an upper bound for the amplitude of time intervals; it is assumed that the precision of the stored values is sufficient to cope with the requirements of the application.

The changes in the structure of *MvtTable* simplify the insertion of new observations. Nevertheless, though this structure could be directly used in query operations no advantages would be achieved. Therefore, since the structure defined in the model is easier to understand, we create the following view:

```
CREATE VIEW Mvt (pid, t0, t1, n, x, y, vx, vy, bx, by) AS
        SELECT  pid, t0, t1, n, xm+vx*(t0-tm), ym+vy*(t0-tm), vx, vy, bx, by
        FROM    MvtTable;
```

Likewise, depending on the requirements of each application, it is possible to create other views to facilitate the query operations.

5.2 Query operations

The following examples illustrate how to implement some query operations using the defined model. They are not intended as an exhaustive exploration of its features but rather as an introduction to the subject.

The data model is represented with a single table, corresponding to the Movement Base. In this table or its derived views like *Mvt*, each tuple contains the information required to describe the behaviour of a moving object within a valid time interval, i.e., each tuple corresponds to a section of a trajectory. Hence, spatial, temporal and spatio-temporal queries can be expressed in SQL using the general format:

```
SELECT  Pid_Function, ValidTime_Function, Spatial_Function, Velocity_Function
FROM    Mvt M1 [, Mvt M2]
WHERE   pid
  AND   ValidTime Condition
  AND   Spatial Condition
  AND   Velocity Condition
 [AND   M1.attribut = M2.attribut];          /* Spatio-temporal auto-equi-join */

                        {attribut = [pid, Valid Time, Spatial, Velocity]}
```

This format allows expressing any combination of projections of pid, valid time, spatial and velocity functions. Each of these projections allows answering different questions. The pid projection allows to answer questions of the kind “which”; valid time interval is used in answers to “when”; spatial window is used in answers to “where”; and velocity in answers to “how”. Some examples follow.

(Query A) Snapshot: Where was the object *pid* at time *t*?

```
SELECT  x+vx*(:t-t0), y+vy*(:t-t0)
FROM    Mvt
WHERE   pid =:pid
AND     :t BETWEEN t0 AND t1;
```

(Query B) Spatio-temporal restriction: Which objects were in region defined by *(x0, x1, y0, y1)* at time *t*?

```
SELECT  pid
FROM    Mvt
WHERE   :t BETWEEN t0 AND t1
AND     (x+vx*(:t-t0)) BETWEEN :x0 AND :x1
AND     (y+vy*(:t-t0)) BETWEEN :y0 AND :y1;
```

(Query C) Static line crossing: When (intervals of time) did the object *pid* cross the line defined by polynomial constraint *p(v)=0*?

```
SELECT   t0, t1
FROM     Mvt
WHERE    pid = :pid
AND      p(v)=0;
```

(Query D) Average speed: Which sections in trajectory of object *pid* were traversed with an average speed lesser than *v*.

```
SELECT   t0, t1, x, y, x+vx*(t1-t0), y+vy*(t1-t0)
FROM     Mvt
WHERE    pid = :pid
AND      sqrt(vx*vx+vy*vy)<=:v
ORDER BY t0;
```

It is worth noticing that the speeds represented in the *Mvt* view are, on the one hand, averages values since it is postulated that the movement between two observations is uniform, and on the other hand, minimal values because it is also postulated that movement is linear. Therefore, the results of queries like (Query D) are based on minimum values of speed.

The previous queries are examples of some common questions in a moving points system. They are simple questions but some remarks should be done. First, it would be useful to have analytical geometry functions like position at a time instant (Query A), spatial window restriction at a time instant or interval (Query B), intersection of trajectories with open and closed polynomial lines (Query C) or calculation of speeds (Query D). Queries would be easier to specify and errors would be prevented.

It is interesting to analyse (Query C) carefully. The result of the query is the time interval in which the mobile crossed the line defined by *p(v)=0*. However, if one is interested in the time instant at which the mobile probably crossed the line, the expression *SELECT t0, t1* should be replaced by *SELECT t*. In this expression, *t* is a free variable obtained from the evaluation of the conditional statement *p(v)=0* and thus, it cannot be used in the projection of a SELECT statement. Although, it is possible, depending on the function *p(v)*, to solve some of these cases, the solutions can be very difficult to express. The problem becomes even more complex when using polylines representing, for example, a frontier. These questions arise because the data may continuously change over time and consequently, the values of some attributes are no longer constant within a time interval or tuple. Further research must be done on this topic, looking for solutions namely in the domains of constraint databases (Chomicki and Revesz 1997) and abstract data types (Scholl et al. 1996).

Frequently, queries will be about trajectories and paths. An example follows.

(Query E) Path length: What the length of the path traversed by object *pid* during the time interval $[t_a,t_b]$?

```
SELECT   SUM(SQRT(vx*(t1-t0)*vx*(t1-t0)+ vy*(t1-t0)*vy*(t1-t0)))
FROM     Mvt
WHERE    pid = :pid
OR       t0 BETWEEN ta AND tb
OR       t1 BETWEEN ta AND tb
OR       ta BETWEEN t0 and t1
```

The previous query is solved to some extent, but some difficulties arise. The model does not deal with fractions of time intervals, though it is unlikely that t_0 of the first time interval and t_1 of the last one, have an exact match with *ta* and *tb* respectively. There is no easy solution to handle this problem in standard SQL. However, this and other issues concerning time intervals are known and have already been in studied the context of temporal databases. Particularly, TSQL2 (Zaniolo et al. 1997) has already proposed operations like the partitioning and coalescing of tuples, the calculation of aggregates or the selection of valid time periods.

Join operations between moving points lead to auto-join operations since they involve two instances of the same structure (in this case the view Mvt).

(Query F) Auto-join: Which are the objects that have crossed the same location at the same speed as the mobile *pid* at time instant *t*.

```
SELECT  M2.pid
FROM    Mvt M1, Mvt M2
WHERE   M1.pid = :pid
AND     M1.vx = M2.vx     AND   M1.vy = M2.vy
AND     M2.x = M1.x       AND   M2.y = M1.y
```

The implementation of queries using the Superset and Subset semantics is not discussed in this paper.

6 Conclusion and future research

The focus of this paper is twofold: the first part analyses the requirements for a system where information about spatial and temporal situation of objects must be kept, while the second proposes a data model for the representation of moving points in GIS suited to represent other kinds of continuous changing data as well.

First, we propose a classification of applications based on the properties of the represented spatio-temporal objects. The main properties are movement and mutation. Movement is captured by detecting changes in the location or direction of the objects. Mutation is caused by changes in the size and shape, which, in complex cases, may involve the fragmentation or aggregation of objects. Building on this classification, we claim that features of complex objects can be derived from those of simpler ones, suggesting an evolutionary approach that starts with the study of simple ones, like moving points, and progresses by enriching them with new features.

The definition of a data model for representing moving points follows. The model adopts the concepts defined in (Yeh and Cambray 1994) for the representation of highly variable data. Each tuple represents a section of a trajectory by a valid time interval, a value corresponding to the location of the mobile at the beginning of the time interval and a variability function of the value. The variability function used is a linear approximation function based on the method of minimum squared deviation. The model allows the aggregation of consecutive observations into a single tuple. The factor of aggregation depends on the precision value used by the approximation function. The precision is a

user-defined value that depends on the requirements of applications. Since, in most systems, it is not possible to store the exact knowledge about the movement of a mobile, the answers to queries are often approximated values. To deal with this imprecision, two additional semantics for the estimation of the position of objects are proposed. Under the Superset semantics the result of a query are guaranteed to include all the tuples matching a given predicate. Under the Subset semantics it is guaranteed that the result of a query only includes tuples that match the given predicate.

Finally, we discuss some issues on the implementation of the data model in a RDBMS. Two main difficulties were identified when implementing some query operations in standard SQL. One is related to the manipulation of continuous changing values and the other concerns the manipulation of time intervals.

This paper identifies several issues that require further work. We intend to study the implementation of the proposed data model on DBMS supporting Abstract Data Types (ADT), namely exploring specific spatial and temporal extensions. On the other hand, the data model is based on a linear regression method. In the future, we intend to study the impact of other interpolation methods, like n^{th}-order polynomials and splines under heterogeneous sets of trajectories (smooth, irregular, etc.), first in the storage requirements and later in the performance of query operations.

This preliminary work has identified several problems that justify the exploration of alternative data models. In this respect, the approaches from constraint databases and abstract data types seem quite promising.

General performance is a crucial issue since, in general, the amounts of information that must be managed on these systems are large. Therefore, particular attention must be given to the development of efficient algorithms and access methods for computing intensive query operations.

References

(Chomicki and Revesz 1997) Jan Chomicki and Peter Revesz. Constraint-Based Interoperability of Spatio-Temporal databases. Proceedings of the 5th International Symposium on Advances in Spatial Databases, SSD'97. Lecture Notes in Computer Science, Springer Verlag, 1997.

(Claramunt and Theriault, 1995) C. Claramunt and M. Theriault. Managing Time in {GIS}: An Event-Oriented Approach. Recent Advances in Temporal Databases, September 1995. Proceedings of the International Workshop on Temporal Databases, Springer Verlag.

(Clifford, Croker and Tuzhilin 1994) James Clifford, Albert Croker and Alexander Tuzhilin. On completeness of historical relational query languages. ACM Transactions on Database Systems, 19(1), March 1994.

(Gutting 1994) Ralf Gutting. An introduction to spatial database systems. VLDB Journal, 3(4), October 1994.

(Jensen 1993, Ed.) C. Jensen (Ed.). A Consensus Test Suit of Temporal Database Queries. Technical Report, Department of Mathematics and Computer, Institute for Electronic Systems, Number R 93-2035, November 1993.

(Jensen et al. 1994) Christian Jensen, J. Clifford, R. Elmasri, S. K. Gadia, P. Hayes, S. Jajodia, C. Dyreson, F. Grandi, W. Kafer, N. Kline, N. Lorentzos, Y. Mitsopoulos, A. Montanari, D. Nonen, E. Peressi, B. Pernici, J. F. Roddick, N. L. Sarda, M. R. Scalas, A. Segev, R. T. Snodgrass, M. D. Soo, A. Tansel, P. Tiberio and G. Wiederhold. A Consensus Glossary of Temporal Database Concepts. SIGMOD Record (ACM Special Interest Group on Management of Data), 23(1), March 1994.

(Koenen 1997, Ed.) Rob Koenen. MPEG-4 Overview. ISO/IEC JTC1/SC29/WG11, Coding of moving pictures and audio, N1730, July 1997.

(Langran 1990) Gail Langran. Tracing temporal information in an automated nautical charting system. Cartography and Geographic Information Systems, 17(4), 1990.

(Langran 1993) Gail Langran. Issues of implementing a spatio-temporal system. International Journal of Geographical Information Systems, 7(4), 1993.

(Nabil 1997) Mohammad Nabil. Modelling and retrieving Multimedia Data Using Spatial and Temporal Information. PhD thesis, University of New South Wales, Sydney, Australia, July 1997.

(Özsoyoglu and Snodgrass 1995) G. Özsoyoglu and R. Snodgrass. Temporal and real-time databases: A survey. Tkde, 7(4), August 1995.

(Roshannejad and Kainz 1995) A. A. Roshannejad and W. Kainz. Handling identities in spatio-temporal databases. Twelfth International Symposium on Computer- Assisted Cartography, Volume 4, 1995.

(Salzberg and Tsotras 1997) Betty Salzberg and Vassilis Tsotras. A Comparision of Access Methods for Temporal Data. Technical Report TR-18, Time Center: Aalborg University – Denmark and University o Arizona – USA, June 1997.

(Scholl et al. 1996) Michel Scholl, Agnés Voisard, Jean-Paul Peloux, Laurent Raynal, and Philippe Rigaux. SGBD Géographiques – Spécifités. International Thompson Publishing, Paris – France, 1996.

(Sistla et al. 1997) A. Prasad Sistla, Ouri Wolfson, Sam Chamberlain and Son Dao. Modelling and Querying Moving Objects. Proceedings of the Thirteenth International Conference on Data Engineering, April 1997. IEEE Computer Society Press.

(Sistla et al. 1998) A. Prasad Sistla, Ouri Wolfson, Sam Chamberlain and Son Dao. Querying the uncertain position of moving objects. Temporal Databases: Research and Practice. Springer Verlag, 1998.

(Snodgrass and al. 1994) Richard Snodgrass, I. Ahn, G. Ariav, D. Batory, J. Clifford, C. E. Dyreson, R. Elmasri, F. Grandi, C. S. Jensen, W. Kafer, N. Kline, K. Kulkarni, T. Y. C. Leung, N. Lorentzos, J. F. Roddick, A. Segev, M. D. Soo and S. A. Sripada. TSQL2 language specification. SIGMOD Record (ACM Special Interest Group on Management of Data), 23(1), March 1994.

(Soo, 1991) M. D. Soo. Bibliography on Temporal Databases. SIGMOD Record (ACM Special Interest Group on Management of Data), 20(1), March 1991.

(Tsotras and Kumar 1996) V. J. Tsotras and A. Kumar. Temporal Database Bibliography Update. SIGMOD Record (ACM Special Interest Group on Management of Data), 25(1), 1996.

(Vazirgiannis, Theodoridis and Sellis 1996) Michael Vazirgiannis, Yannis Theodoridis and Timos Sellis. Spatio-temporal Composition in Multimedia Applications. Proceedings of IEEE-ICSE 96, Workshop on Multimedia Software Development, March 1996.

(Worboys 1994) Michael Worboys. Unifying the spatial and temporal components of geographical information. Sixth International Symposium on Spatial Data Handling, Volume 1, 1994.

(Yeh and Cambray 1994) T.-S. Yeh and B. de Cambray. Managing highly Variable Data in a Temporal Database. A Medical Application. Second International Symposium on Applied Corporate Computing (ISACC'94), October 1994.

(Yeh and Cambray 1995) T.-S. Yeh and B. de Cambray. Modelling Highly Variable Spatio-Temporal Data. Sixth Australian Database Conference (ADC'95), January 1995.

(Yeh and Viémont 1992) T.-S. Yeh and Y. H. Viémont. Temporal Aspects of Geographical Databases. Proceedings of EGIS, 1992.

(Zaniolo et al. 1997) Carlo Zaniolo, Stefano Ceri, Christos Faloutsos, Richard Snodgrass, V. Subrahmanian, and Roberto Zicari. Advanced Database Systems. Morgan Kaufmann Publishers, San Francisco, California, 1997.