Moving Objects: Logical Relationships and Queries

Jianwen Su*, Haiyan Xu**, and Oscar H. Ibarra*

Department of Computer Science
University of California
Santa Barbara, CA 93106
USA
{su,xu,ibarra}@cs.ucsb.edu

Abstract. In moving object databases, object locations in some multi-dimensional space depend on time. Previous work focuses mainly on moving object modeling (e.g., using ADTs, temporal logics) and ad hoc query optimization. In this paper we investigate logical properties of moving objects in connection with queries over such objects using tools from differential geometry. In an abstract model, object locations can be described as vectors of continuous functions of time. Using this conceptual model, we examine the logical relationships between moving objects, and between moving objects and (stationary) spatial objects in the database. We characterize these relationships in terms of position, velocity, and acceleration. We show that these fundamental relationships can be used to describe natural queries involving time instants and intervals. Based on this foundation, we develop a concrete data model for moving objects which is an extension of linear constraint databases. We also present a preliminary version of a logical query language for moving object databases.

1 Introduction

Existing technology has made it possible to track down movements of target objects in the air (e.g., airplanes), on the land (e.g., vehicles, wild animals, people, etc.) and ocean (e.g., ships, animals). Among the challenges novel applications involving such "moving objects" have brought to software development is the problem of data management. In a nutshell, there is a wide range of issues including modeling and representation of moving objects, query language design, indexing techniques, query optimization, and even extensions to the well known and widely adopted "data independence" principle. Prior work on spatial and/or temporal databases is relevant but insufficient for moving objects. The goal of this paper is to revisit the foundation of data models and query languages for

* Supported in part by NSF grants IRI-9700370 and IIS-9817432

** On a sabbatical leave from Department of Computer Science, Fukoka Institute of Technology, Japan

C.S. Jensen et al. (Eds.): SSTD 2001, LNCS 2121, pp. 3–19, 2001.

moving objects by taking a new perspective from a branch of mathematics—differential geometry.

Management of moving objects has been investigated in the recent years. Sistla and Wolfson et al [SWCD97,WXCJ98] developed a data model combining future temporal logic (FTL) with topological operators. Although FTL allows reasoning about temporal properties on individual moving objects, the coupling of the spatial and temporal relationships is inevitably loose hindering the reasoning process as well as eventually the query optimization. Coming from spatial data modeling, a group of researchers proposed extensions based on abstract data types (ADTs) [EGSV99,FGNS00,GBE+00]. These are practical solutions to the modeling and querying problems. Unfortunately, ADTs are too "rigid" to express sometimes intimate temporal and spatial configurations of moving objects. Indeed, once the operators are fixed, new and sophisticated relationships involving moving objects may be inexpressible since the "host" language (likely SQL) lacks the ability to "look into" the inside of an ADT. On the other hand, the focus of the modeling and querying was mostly on the spatial and temporal relationships that can be described in algebraic geometry. Güting et al [GBE+00] realized this deficiency and added four operations for computing velocity, derivative, turn, and speed as ADT operations. But the underlying reasoning for including these is unclear. They seem to deviate from the well known principles of differential geometry [MP77,Gra98]. Forlizzi et al [FGNS00] also considered moving lines and regions which is not the focus of this paper.

Constraint databases [KKR95] present an elegant conceptual model for spatial and temporal data. The primary idea is to use mathematical formulations to represent spatial and temporal data and use first order logic to express queries [KLP00]. However, the framework is based on elementary algebra and thus the relationships expressible are the ones in algebraic geometry. For example, reasoning about moving speed and direction is out of reach.

There are other related work on moving objects, but these mostly focus on indexing and algorithm issues [KGT99,AAE00] and location management and dealing with imprecision [WCDJ97,WCD+98,WJS+99].

In this paper, we consider modeling moving points in some vector space and use techniques from differential geometry [Gra98]. A focus is to conduct an exposition with well known mathematical tools. Through the analysis of logical relationships concerning moving points, we conclude that simple primitives of velocity and acceleration along with vector operations are sufficient to express not only movement-related relationships but also the well studied topological and temporal relationships. Based on this finding, we propose an extension to the linear constraint database model which allows these simple primitive operations. We also develop an extended calculus query language based on some standard linear constraint query languages (see [KLP00]). We show that this language has polynomial time complexity.

This paper is organized as follows. Section 2 gives some preliminary concepts, and Section 3 introduces the abstract model for moving points. Section 4 focuses on the relationships of the moving objects. Section 5 presents a design of a

concrete data model based on linear constraints and associated query language extended from the constraint calculus. Section 6 concludes the paper.

2 Preliminaries

In this section we briefly review some of the fundamental concepts related to curves in multi-dimensional space and some operations on vectors. These concepts will be used in the technical discussions throughout the paper. We also discuss necessary concepts of spatial databases that we will use in conjunction with moving objects.

We assume the existence of primitive data types such as *real* numbers and *time* instants. The domains of these types are standard. In our discussions, the domain of *time* has a dense total order isomorphic to the real numbers, i.e., time is continuous. For spatial data in the n-dimensional space, we assume the existence of spatial types such as $region_n$, $line_n$, $point_n$ for each n (≥ 1). The domains of these types are *semi-algebraic* regions, lines, and points (respectively). Semi-algebraic sets can be easily represented by quantifier-free first-order formulas using the constraint database approach [KKR95,KLP00].

We will model moving objects in such a way that their positions are continuous functions of time in an appropriate space. We use standard concepts including "differentiable" and "derivative" from differential calculus (cf [Gra98]). Although these concepts are abstract and mathematical, the techniques from constraint databases provide a simple conceptual paradigm and useful techniques of manipulating constraint representations of abstract functions. In this respect, we will consider function data types such as $time \rightarrow real$, which can be used to represent speed of moving objects.

Let n be a positive integer and $\tau_1, ..., \tau_n$ be n data types. An *n-ary vector* of type $\tau_1 \times \cdots \times \tau_n$ is a tuple $(v_1, ..., v_n)$, where v_i is a value in the domain τ_i for each $1 \leq i \leq n$.

Our focus is on the moving objects that are points in the n-dimensional (real) space for some integer $n \geq 1$. For this purpose, we also view the n-ary vector type $real \times \cdots \times real$ as a data type $point_n$ for points in the n-dimensional space.

Let $\mathbf{u} = (u_1, ..., u_n)$ and $\mathbf{v} = (v_1, ..., v_n)$ be two n-ary vectors over the real numbers. We allow the following standard operations: $\mathbf{u} + \mathbf{v} = (u_1 + v_1, ..., u_n + v_n)$, and $c \cdot \mathbf{u} = (c \cdot u_1, ..., c \cdot u_n)$ where c is a real number. (Note that $\mathbf{u} - \mathbf{v}$ can be defined as $\mathbf{u} + (-1) \cdot \mathbf{v}$.) The *dot product* of two vectors $\mathbf{u} = (u_1, ..., u_n)$ and $\mathbf{v} = (v_1, ..., v_n)$ is defined as $\mathbf{u} \cdot \mathbf{v} = \Sigma_{i=1}^{n} u_i v_i$. The (Euclidean) *length* of a vector $\mathbf{u}$, denoted as $\|\mathbf{u}\|$, is defined as $\sqrt{\mathbf{u} \cdot \mathbf{u}}$. A *unit* vector has the length 1. It is clear that for each given vector $\mathbf{u}$, one can scale it down to obtain the unit vector by $\frac{1}{\|\mathbf{u}\|} \times \mathbf{u}$. Unit vectors are useful to represent moving "directions".

Unless otherwise specified, we assume that n is the dimension in which spatial objects and moving objects exist.

3 Moving Objects

In this section, we present an abstract model of moving objects using vector space and related standard methods from differential geometry [MP77,Gra98] and highlight some of the useful fundamental concepts and constructs related to query languages for moving objects. In the next section, these concepts and constructs are used to explore and formulate logical relationships of moving objects, which will help in developing query languages.

Let n be a positive integer. In our abstract model, a "moving point" in the n-dimensional space is a function from *time* to $point_n$ that is continuous and infinitely differentiable. In applications, moving objects may change directions and/or become stationary from time to time. To accommodate such changes, we allow a moving point to consist of a finite number of segments of movements, where in each segment the function remains infinitely differentiable.

Let t and t' be two time instants and $t < t'$ and f a function over time. We denote $f|_{(t,t')}$ the restriction of f over the domain (t, t').

Definition. A *moving point* is a function of type $\mathbf{p} : time \to point_n$ that can be represented as a vector of functions $\mathbf{p}(t) = (p_1(t), p_2(t), \ldots, p_n(t))$ satisfying the following conditions:

1. for each $1 \leq i \leq n$, p_i is continuous over time, and
2. There exist a positive integer m and m time instants $t_1 < t_2 < \cdots < t_m$ such that for each $1 \leq i \leq n$,
   a) $p_i|_{(-\infty,t_1)}$ and $p_i|_{(t_m,\infty)}$ are infinitely differentiable, and
   b) for each $1 \leq j \leq m-1$, $p_i|_{(t_j,t_{j+1})}$ is also infinitely differentiable.

In the remainder of the paper, we use $\mathbf{p}(t), \mathbf{q}(t), \ldots$ to represent moving objects, or just $\mathbf{p}, \mathbf{q}, \ldots$ when it is clear from the context. We also use notations such as p_i, q_j to mean the i-th and j-th components of $\mathbf{p}$ and $\mathbf{q}$, respectively.

Let $\mathbf{p} = (p_1, ..., p_n)$ be a moving point. Clearly $(p_1, ..., p_n)$ defines the *position* of $\mathbf{p}$ as a function of time. The *velocity* of $\mathbf{p}$ is defined as the derivative of $\mathbf{p}$: $vel(\mathbf{p}) = \mathbf{p}' = (p_1', ..., p_n')$. The *acceleration* of $\mathbf{p}$ is the second order derivative of $\mathbf{p}$, $acc(\mathbf{p}) = (vel(\mathbf{p}))'$.

In the following we give some example of using these primitives to express some commonly seen properties. These are in fact some basic concepts from differential geometry for curves. Let $\mathbf{p}, \mathbf{q}$ be moving points.

1. *Moving direction* of a moving point at some time instant is the tangent vector of the curve at this time instant. The moving direction can be computed as the directional derivatives, i.e., $vel(\mathbf{p})$. Sometimes, we scale it to a unit vector by $\frac{vel(\mathbf{p})}{\|vel(\mathbf{p})\|}$.
2. *Speed* of a moving point is the distance it travels through in a unit time. This is also related to velocity and can be expressed as the length of the velocity: $\|vel(\mathbf{p})\|$.
3. *Distance between two moving points* is the shortest Euclidean distance between the two points. This is easily expressible as the length of the difference

of two vectors: $\|\mathbf{p} - \mathbf{q}\|$. If $\mathbf{x}$ is a stationary point, the distance of $\mathbf{p}$ and $\mathbf{x}$ is defined in the same way: $\|\mathbf{p} - \mathbf{x}\|$. Sometimes it is useful to reason about distance between a point and a region (or line). If r is a region, the distance between $\mathbf{p}$ and r is defined as: $\min_{\mathbf{x} \in r}\{\|\mathbf{p} - \mathbf{x}\|\}$.

4. Two points are moving in the *same direction*. If $\mathbf{p}, \mathbf{q}$ are two moving points, they move in the same direction iff the unit vectors of their moving directions are exactly the same. Similar, we can use the moving directions to express *similar directions* of two moving points.

The focus of this paper is on the query languages for moving object databases. The modeling and representation problems for moving objects are studied in this context. There also are interesting issues of how to model moving objects from the application point of view. For example, linear equations may be appropriate to approximate truck movements while quadratic equations are more accurate for flight paths. These issues should be addressed for specific domains and are not in the scope of this paper.

4 Logical Relationships of Moving Objects

In this section, we study logical relationships of moving objects of interest to queries over such objects. Previously investigated relationships for moving objects can be roughly grouped into two categories: relationships based on time instants and on time intervals [SWCD97,WXCJ98,EGSV99,GBE+00,FGNS00]. Clearly, spatial relationships such as topological relations [EF91] and temporal relationships are relevant to moving objects. However, these relationships can be described in algebraic geometry. Movements of the moving objects bring new perspectives of the relationships. Indeed, relationships such as moving speed and direction are only natural but also they need differential geometry methods [MP77,Gra98] that are beyond algebraic geometry. We argue that primitives including velocity and acceleration from differential geometry [Gra98] are not only necessary for expressing relationship concerning movements, but also sufficient to express spatial and temporal relationships. This provides a foundation to develop a data model and a query language for moving objects, which will be discussed in the next section.

To facilitate the discussions, we consider a database with the following classes:

FLIGHTS (airline: *string*, number: *string*, position: $moving\_point_3$)
AIRPORTS (code: *string*, state: *string*, location: $region_2$)
RUNWAYS (airport: *string*, number: *string*, landing-direction: $real \times real$)
COUNTIES (name: *string*, state: *string*, area: $region_2$)

Here the relation FLIGHTS contains the flight information: the airline name and flight number, and the current position of the aircraft. The position data is a moving point in the 3-dimensional space. The AIRPORTS and COUNTIES relations contain only stationary spatial data for airport locations and county regions which are regions in the 2-dimensional space. The relation RUNWAYS include runway orientations specified as unit vectors in the 2-dimensional space.

We consider the following queries.

q_1: Locate all United flights over Ventura at 4pm, January 29, 2000.
q_2: List all pairs of flights within 300 miles of the San Francisco Airport right now that are moving in opposite directions and both are increasing speed.
q_3: Report all flights approaching LAX within 45 degrees of the orientation of runway 3 that are below 20,000 feet. (This may be needed for rescheduling flight landings due to a runway change.)
q_4: Find all United flights that entered Santa Barbara (SB) county during time t_1 to t_2.
q_5: Show all flights that have been flying above LA county for the last 15 minutes.

Although the queries and the databases seem slightly artificial, it is not difficult to imagine that such temporal-spatial relationships could easily occur in queries in various applications. Our focus is to develop appropriate model and query languages through the study of the logical relationships needed in querying moving object databases.

If we take a close look at the above queries, q_1 is a spatial selection (range intersection) along the longitude and latitude for a given a time instant (snapshot). This indicates that the topological, spatial, and temporal relationships in the traditional temporal-spatial databases remain relevant to querying moving object databases. However, the movements of the objects introduce new types of relationships. For example, it should allow computation of moving speed from which distance and time information can be derived. Also, moving direction is important and useful. They are exhibited in queries q_2 and q_3. Query q_4 includes a property "enter" which is different from the intersection predicate in the traditional spatial relationships. In fact, the predicate does not make much sense for static objects but is very useful for moving objects. For query q_5, one way to express the property is to first extract the trajectories of all flights in the specified time duration and followed by a spatial containment query. An alternative is to state directly that these flights indeed stayed in the area during the last 15 minutes. The latter is easier to understand and perhaps more desirable.

The functionality of a DBMS goes far beyond query processing. Similarly, a conceptual data model should not only include data representation tools but more importantly support data manipulation languages which are easy to use and whose queries can be optimized. To reach this goal, we have to understand logical relationships for moving objects and the goal to develop simple and familiar yet expressive primitives that can facilitate query optimization.

The 9-intersection model for topological relations [EF91] simplifies the topological properties. It would be desirable to develop similar simple models for moving object relationships. For example, we can classify binary moving object relationships in the following three dimensions: between moving objects or moving and stationary objects, involving time instant or time interval, and related to position, velocity, or acceleration (Fig. 1 below).

Fig. 1 shows some relationships in some of the categories. For example, the predicate "in" refers to a moving object is inside a region at a time instant. This can be used for query q_1. "Enter" states that the object intersects the boundary

moving vs.	*position*		*velocity*		*acceleration*	
moving	dist-lessthan, collision	catching-up	opp-dir, closer	collision-course	meet	ap-proaching
stationary	in, on-line, dist-lessthan	stays-in	enter, aim-at, closer	toward	land	ap-proaching
time	*instant*	*interval*	*instant*	*interval*	*instant*	*interval*

Fig. 1. Three dimensional view of moving object relationships

of a region in the direction from outside to inside; the latter part is related to the velocity. Query q_4 will need this. "Opposite direction" ("aim-at") means that two objects are moving toward each other (respectively the object is moving toward a region) at an instant, which can be used for q_2. "Collision" means that two objects are in the same position at an instant. "Collision course" refers to objects on a collision course for a period of time. "Meet" (or "land") is a refined version that requires the objects' speed to be the same (or 0, resp.) and acceleration to be either positive for the slower object or negative for the faster object, or both. "Approaching" specifies the changes in speed and direction toward collision for a time period.

Clearly, there are many more relationships that one can formulate among moving objects. It is thus interesting to identify "fundamental" or "primitive" ones.

Mathematics provides tools for conceptualization and modeling in computer science and especially in databases. Indeed, much of the development and study of the relational languages has been based on first order logic (see [AHV95]). In spatial databases, elementary geometry techniques were the basis for characterizing topological relationships (e.g., [EF91,Ege91]). The introduction of constraint databases brought algebraic geometry into the database context [KLP00], where more complex topological properties can be analyzed using real variables and elementary algebra. Representative work includes [PSV96,KPdB97,KdB99]. It is little surprising that this long tradition would continue.

Differential geometry combines (differential) calculus with geometry (see [MP77,Gra98]). The use of calculus allows the capture of much finer geometric properties. Take curves as an example, velocity of a curve (i.e., moving point) is simply its (directional) derivative and the derivative of the velocity gives the acceleration. Other important properties such as curvature and torsion can then be expressed. In fact, the *Fundamental Theorem of Curves* states that each curve is uniquely determined by its curvature and torsion (modulo position) [MP77].

Once a time instant is fixed, moving objects are simply points in the space. Therefore, all spatial relationships are relevant and needed. In addition, the time dimension explicitly exists for moving objects and temporal relationships should also be included. Although the combination of temporal and spatial relationships can help to express "entering a region", they are nevertheless insufficient for

reasoning about the moving directions and speed as well as their changes. It is the weakness inherited from the underlying mathematical model of elementary geometry. The weakness was also observed in [GBE+00]. Although the model of [GBE+00] for moving object uses ADTs with operation including derivatives, their approach of simply adding missing and needed operations makes their language cumbersome and very difficult to use. The consequences are that the language is difficult to analyze and query optimization is hard or even impossible. We believe that the difficulty comes really from their ADT approach, and not from the moving objects themselves. A necessary step towards an elegant simple query language is to analyze the fundamental relationships of moving objects.

For the focus of our exposition on logical relationships of moving objects, we consider velocity and acceleration of moving points represented in the vector space, along with the usual operations on vectors. Thus in our model, we provide:

1. Vectors (vector space) for representing moving points and vector additions and multiplications by real numbers.
2. If $\mathbf{p}$ is a vector of a moving point, then the velocity $vel(\mathbf{p})$, acceleration $acc(\mathbf{p})$, length $\|\mathbf{p}\|$, and moving direction $dir(\mathbf{p}) = \frac{vel(\mathbf{p})}{\|vel(\mathbf{p})\|}$.

Although the operations listed above are very simple, we show that many interesting logical relationships including those we listed in Fig. 1 are expressible in terms of these. Intuitively, distance can be obtained from vector length. With distance one can easily express topological relationships in the sense of [EF91]. Moving direction information is derivable from velocity. Moving direction can help to express some temporal relationships between moving objects. Our model also allows the primitives to be combined using arithmetic constraints (in the spirit of differential calculus) and logical connectives.

Although according to the Fundamental Theorem of Curves, curvature and torsion are key properties in characterizing curves, we do not include them as primitives. This is primarily due to our focus on queries in the present paper. We believe that curvature and torsion are important and can be useful in things such as comparing and querying trajectories. This will be left for future work since our focus of this paper is mainly on the motions.

In the remainder of this section, we give a few example relationships and show how they can be expressed. Although these are mostly standard in differential calculus [Gra98], we include them here for exhibiting the intimate relationships between moving object modeling/querying and differential geometry techniques.

We first look at some distance relationships. Given two moving points $\mathbf{p}$ and $\mathbf{q}$, their distance at time t can be expressed as $\|\mathbf{p}(t) - \mathbf{q}(t)\|$, which we denote as $dist(\mathbf{p}, \mathbf{q}, t)$. The following are some relationships concerning distance:

- $distance\_lessthan(\mathbf{p}, \mathbf{q}, t, d) \equiv (dist(\mathbf{p}, \mathbf{q}, t) \leq d)$.
- $in(\mathbf{p}, r, t) \equiv (\mathbf{p}(t)$ inside $r)$, where r is a region and "inside" is a spatial predicate.
- $collision(\mathbf{p}, \mathbf{q}, t) \equiv (\mathbf{p}(t) = \mathbf{q}(t))$.
- $catching\_up(\mathbf{p}, \mathbf{q}, t_1, t_2) \equiv (\forall tt'(t_1<t<t'<t_2) \rightarrow dist(\mathbf{p}, \mathbf{q}, t)>dist(\mathbf{p}, \mathbf{q}, t'))$. Similarly "stays_in" can be expressed.

For moving direction related relationships, velocity is needed.

- $opposite\_direction(\mathbf{p}, \mathbf{q}, t) \equiv (dir(\mathbf{p})(t) + dir(\mathbf{q})(t) = \mathbf{0})$ and also $same\_direction(\mathbf{p}, \mathbf{q}, t) \equiv (dir(\mathbf{p})(t) = dir(\mathbf{q})(t))$.
- $on\_collision\_course(\mathbf{p}, \mathbf{q}, t_1, t_2) \equiv$ $(\forall t(t_1 < t < t_2 \rightarrow opposite\_direction(\mathbf{p}, \mathbf{q}, t)) \wedge \exists t(t_2 < t \wedge collision(\mathbf{p}, \mathbf{q}, t)))$.
- $aim\_at(\mathbf{p}, r, t) \equiv (\exists \mathbf{x}(\mathbf{x} \text{ inside } r \wedge (unit(\mathbf{x} - \mathbf{p}(t)) = dir(\mathbf{p})(t))))$, where $unit(\mathbf{x})$ converts $\mathbf{x}$ to a unit vector.
- $enter(\mathbf{p}, r, t) \equiv (on\_line(\mathbf{p}, \text{boundary}(r), t) \wedge \exists t' (t' < t \wedge \forall t'' (t' < t'' < t \rightarrow \neg(\mathbf{p}(t) \text{ inside } r))))$ where "boundary" returns the boundary lines of a region r. The formula states that just before $\mathbf{p}$ moves across the boundary of r, $\mathbf{p}$ is always outside of r. Note that this expression uses interval property and not velocity. It can also be expressed with the moving direction from the velocity of $\mathbf{p}$.

Finally, in some cases acceleration is needed. For example, $land(\mathbf{p}, \mathbf{x}, t) \equiv dist(\mathbf{p}, \mathbf{x}, t) = 0 \wedge \|vel(\mathbf{p})\| = 0 \wedge acc(\mathbf{p}) < \mathbf{0}$, where the last condition states that the velocity slows down along all coordinates.

5 A Linear Model for Moving Objects

In this section, we introduce a new data model and a query language for moving objects. Unlike the earlier models based on ADTs [EGSV99,FGNS00,GBE+00] and temporal logic [SWCD97], the new model combines the linear constraint techniques from constraint databases [KLP00] with the operations and relationships we studied in the previous section. We show that queries in the language can be evaluated efficiently in polynomial time.

Constraint databases [KKR95] are particularly suitable for conceptual modeling and manipulation of multi-dimensional data [KLP00]. Central to constraint databases are mathematical formulas with a prescribed semantics (a.k.a. constraints).[1] Constraint database techniques provide a fundamentally sound approach to spatial and temporal database applications. The elegance of constraint models lies in that they allow spatio-temporal data to be viewed conceptually as mathematical objects in algebraic geometry. The important data independence principle can be very naturally supported.

Constraints can be integrated into relational or object-oriented structures to suit applications. For example, the data model using conjunctions of linear constraints as values that can then be used as values in a classical relations was adopted in the DEDALE system [GRS98], the constraint query and update languages of [BBC98], and the COSMOS project [KRSS98]. With this model, one can represent information such as follows. Consider the relation COUNTIES which contains the names, states, and regions of counties. Here region data are spatial, representing the administrative

[1] This is not to be confused with *integrity constraints.*

COUNTIES

name	state	area
Santa Barbara	California	$region_{\text{sb}}$
Ventura	California	$region_{\text{v}}$
Los Angeles	California	$region_{\text{la}}$
...	...	...

areas. We can represent the information by a relation COUNTIES shown above. The attribute *area* has values that are 2-dimensional regions in the (real) plane. In a linear constraint database such regions are represented by mathematical objects using boolean combinations of linear equations and inequalities. For example, $region_{\text{v}}$ is represented as a formula $\varphi(x_1, x_2) = 68 \leq x_1 \leq 70 \wedge x_2 \leq 2 \wedge 70 \leq x_1 + 8x_2$, where a point (x_1, x_2) is in $region_{\text{v}}$ iff $\varphi(x_1, x_2)$ is true.

Linear constraints over the real numbers may involve equality and order predicates $(=, <, \leq, >, \geq)$ and addition $(+)$. The following are examples of linear constraints: $\Sigma_{i=1}^{p} a_i\, x_i = a_0$, $\Sigma_{i=1}^{p} a_i\, x_i \leq a_0$, where x_i's are variables and a_i's are real (or rational for the linear case[2]) numbers.

We now define our linear constraint model for moving objects. Intuitively, we use linear constraints to represent the functions from time to points in space. Since time and space are over real numbers, this means that the linear constraints are over a time variable t and n dimension variables, $x_1, ..., x_n$.

Definition. A moving point $\mathbf{p}$ has a *linear constraint representation* if there exist a positive integer m and real numbers $a_1, ..., a_m$ satisfying the following conditions.

1. $a_1 < a_2 < \cdots < a_m$, and
2. let a_0 be a symbol representing $-\infty$ and a_{m+1} representing $+\infty$ (for technical convenience), and for each $1 \leq i \leq n$, the function $p_i(t)$ is represented by the following constraints:

$$p_i(t) = \bigvee_{j=0}^{m} (x_i = b_{ij}t + c_{ij} \wedge a_j \leq t \leq a_{j+1}) \tag{1}$$

where b_{ij}'s and c_{ij}'s are real numbers and for each $1 \leq j \leq m$ and each $1 \leq i \leq n$,

$$b_{ij-1}a_j + c_{ij-1} = b_{ij}a_j + c_{ij} \tag{2}$$

In the above definition of a linear constraint representation of a moving point $\mathbf{p}$, each coordinate p_i of $\mathbf{p}$ is a linear function of the time variable t (equation 1), and $\mathbf{p}$ may change speed and direction at time instants $a_1, ..., a_m$. The condition (2) in the definition is to ensure that the coordinate functions are continuous at these time instants. Clearly linear functions are infinitely differentiable. In fact, their derivatives are constants and second or higher order derivatives are all zeros.

[2] The first order theory of rational numbers with multiplication is undecidable.

Example 5.1 Consider a tuple in the relation FLIGHTS where the moving point values of positions are in the 3-dimensional space. A part of the position value of the tuple is shown as follows:

$$x_1{=}2t{-}40 \wedge 0{\leq}t{\leq}21 \bigvee x_1{=}2 \wedge 21{\leq}t{\leq}22 \bigvee x_1{=}\tfrac{1}{2}t{-}9 \wedge 22{\leq}t{\leq}47$$
$$x_2{=}-t{+}23 \wedge 0{\leq}t{\leq}21 \bigvee x_2{=}-t{+}23 \wedge 21{\leq}t{\leq}22 \bigvee x_2{=}1 \wedge 22{\leq}t{\leq}47$$
$$x_3{=}30 \wedge 0{\leq}t{\leq}21 \bigvee x_3{=}-5t{+}135 \wedge 21{\leq}t{\leq}22 \bigvee x_3{=}-t{+}47 \wedge 22{\leq}t{\leq}47$$

The airplane described above moved towards southeast first, turned at time 21 (and position $(2, 2, 30)$) and also started to descend, and made another turn at time 22 (and position $(2, 1, 25)$) and continued to descend until landed at time 47 (and position $(14.5, 1, 0)$). ∎

Example 5.2 Continue with Example 5.1. The velocities of the airplane were $(2, -1, 0)$ from time 0 to 21, $(0, -1, -5)$ during the interval $(21, 22)$, $(\frac{1}{2}, 0, -1)$ during the final landing phase. The speeds were $\sqrt{5}, \sqrt{26}, \frac{\sqrt{5}}{2}$, respectively. ∎

In the linear representation case, clearly the velocity is always constant and the acceleration remains 0. Therefore the acceleration primitive is unnecessary. In our model, we will have vectors and the usual operations, *vel*, *dir*, and $\|\cdot\|$.

Using the above linear constraint moving points as a primitive type, we can build a data model for moving points in a systematic way. We use the relational model to demonstrate the approach, although it is as easy to use an object oriented model.

An ADT encapsulates the internal structure of a data type and thus provides a clear interface by a set of operations on the data type. This is appropriate for some applications in databases such as to deal with binary large objects. Although the ADT approach may be appropriate in some applications, we believe that it is not an ideal approach for spatial and temporal databases. This is due to the need to distinguish different spatial data types, and the intimate relationships which exist between them. On one hand, modeling all temporal-spatial types using a single ADT over-simplifies the data structure and loses the flexibility of doing spatial and temporal reasoning. On the other hand, if we treat different temporal-spatial types as different ADTs we are forcing another "rigid" design and spatial and temporal reasoning is equally hard.

In any case, ADTs are limited by the operations they provide. If the queries and update operations are completely predictable, ADTs can be developed. For example, it would be impossible to express some logical relationships between the elevations of two objects had elevation not been an ADT operation. Therefore, in cases ad hoc queries are to be supported, one has to provide a logic language that is capable of combining ADT operations beyond sequential compositions. An interesting question here is then what is the minimum set of *basic* ADT operations needed since some operations can be expressed by combining the basic operations.

We believe the temporal-spatial data types should encapsulate implementation details as ADTs do but they should expose a "conceptual structure" which

can be used for expressing temporal spatial relationships. The latter differs from the spirit of ADTs.[3] We propose the notion of a "logical data type" (LDT). Intuitively, an LDT in our model provides a relation schema for representing the data inside the values and the data can be accessed directly outside of the LDT and be reasoned about. We give the following technical notion.

Definition. Let $\tau = \mathit{time} \to \mathit{real}$ be a function of type from time to real numbers and n a positive integer. The n-ary *logical vector* type (LV type), denoted as $\boldsymbol{P}_n$, is the type $\tau^n = \tau \times \cdots \times \tau$. The *domain* of $\boldsymbol{P}_n$ is the set of all moving points having linear constraint representations.

Although $\boldsymbol{P}_n$ resembles syntactically a product type of n function types from time instants to real numbers, the semantics (domain) is different. We use $\boldsymbol{P}_n$ to represent moving points in n-dimensional space in this paper, where each function gives the mapping from time to a coordinate value.

A *relation schema* is a finite set of pairs (A, T), where A is an attribute name and T a type or LV type such that the attribute names are distinct. A *tuple* over a relation schema R is a total mapping from attribute names in R to elements in the domains of the respective types or LV types. A *relation instance* of a relation schema R is a finite set of tuples over R. A *database schema* is a finite set D of relation schemas and a *database instance* of D is a total mapping I from D such that for each R in D, $I(R)$ is a relation instance of R.

We now consider query languages for our model of moving objects. We start with a discussion on constraint query languages for constraint databases.

The relational calculus was extended to a constraint query language [KKR95]. In the traditional constraint database context, a first-order formula $\varphi(\overline{x})$ with free variables $\overline{x}$ defines a *query* Q in the following sense: if I is a constraint database, the answer to Q on I is defined as

$$Q(I) = \{\overline{a} \mid \text{ the database } I \text{ satisfies } \varphi(\overline{a})\}.$$

A key point here is that the answer $Q(I)$ may be an infinite relation but *should* be definable as a constraint relation.

It is not obvious that the first-order logic defines a query language for constraint databases. The key "ingredient" is the quantifier elimination property of the first-order theory of real closed fields. A logical theory admits *quantifier elimination* if every formula is equivalent to a quantifier-free formula. The fact that the theory of real closed fields admits quantifier elimination was first discovered by Tarski [Tar51]. There are tractable algorithms to perform quantifier elimination for a fixed number of distinct variables [BKR86,Col75,Ren92]. This property was used in [KKR95] to design the constraint database framework and is necessary [GS97].

In our model, a relation has a finite set of tuples. Tuples provide some descriptive and discrete information, while the spatial regions are represented as

[3] In theory, one could argue that ADTs could provide conceptual structures but such ADTs can't really support optimization of sequences of operation on the ADTs.

the same way as in constraint relations (e.g., [KRSS98]), and moving points are represented as values in LV types.

We can extend the relational calculus (or a constraint query language) for moving objects in our model. Specifically, we allow real variables (representing time and spatial coordinates) to access the data inside values of LV types, linear arithmetic constraints on these real variables, and first order constructions ($\vee, \wedge, \neg, \forall, \exists$) on the constraints. The language design needs to be carefully crafted to incorporate LV types and associated operations and, more importantly, to ensure query results to always be relations (i.e., finite sets of tuples with values in the domains of types and LV types). Although this is not technically difficult, it is not a trivial task.

The *terms* in our calculus include variables (with associated types or LV types), values in the domain of types or LV types, addition of two terms of type *real* (or *time*), multiplication of a *real* (or *time*) term by a real number, and also the following.

- If $\mathbf{p}, \mathbf{q}$ are terms of type $\boldsymbol{P}_n$ and c a term of type *real*, then $\mathbf{p} + \mathbf{q}$, $c \cdot \mathbf{p}$ are also terms of type $\boldsymbol{P}_n$.
- If $\mathbf{x}$ is a vector, $\mathit{unit}(\mathbf{x})$ is to convert it into a unit vector with the same direction.
- If $\mathbf{p}$ is a term of type $\boldsymbol{P}_n$, $\mathit{vel}(\mathbf{p}), \mathit{dir}(\mathbf{p})$ are terms of type $\boldsymbol{P}_n$ and $\mathit{len}(\mathbf{p})$ is a term of type *real*.

Since a value of type an n-ary vector of *real* types can also be viewed as a value of type $\boldsymbol{P}_n$ (with all constant functions). We naturally extend the operations on LV types to include vectors of reals.

Let $\mathbf{p}$ be a value in the domain of $\boldsymbol{P}_n$, a be a time instant, and $b_1, ..., b_n$ be n real numbers. The expression "$\mathbf{p}(a; b_1, ..., b_n)$" is true if the moving point $\mathbf{p}$ is at the position $(b_1, ..., b_n)$ at the time instant a.

The formulas in our query language include the following.

- $x = y$ if x, y are terms of the same type, or $x \leq y$ if x, y are *real* or *time* terms,
- $\mathbf{p}(t; x_1, ..., x_n)$ if $\mathbf{p}$ is a term of type $\boldsymbol{P}_n$, t a *time* term, and $x_1, ..., x_n$ are *real* terms.
- $R(x_1, ..., x_k)$ if R is a relation schema in the database and x_i's are terms of the respective types.
- $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \exists x\varphi, \forall x\varphi$ are formulas if φ, ψ are formulas and x is a variable.

A *query* is an expression of form "$\{(x_1, ..., x_k) \mid \varphi\}$" where φ is a formula with all free variables in $x_1, ..., x_k$.

We illustrate our query language through the example queries listed in Section 4. We use the first two letters to abbreviate relation names.

1. The query q_1 can be expressed as follows.

$$\left\{ (n, x_1, x_2, x_3) \;\middle|\; \exists \mathbf{p} \exists r \left(\begin{array}{l} \textsc{Fl}(\mathrm{UA}, n, \mathbf{p}) \wedge \mathbf{p}(t_0; x_1, x_2, x_3) \wedge \\ \textsc{Co}(\mathrm{Ventura}, \mathrm{CA}, r) \wedge r(x_1, x_2) \end{array} \right) \right\}$$

Here t_0 is the value for the time instant "4pm on January 19, 2000". Also, r is a 2-dimensional region and the expression "$r(x_1, x_2)$" is true if the point (x_1, x_2) is in r. Note that this is a conjunctive query and directly corresponds to the English version.

2. Query q_2, since all moving points in linear representation have no acceleration, can be expressed as follows:

$$\{(n_1, n_2) \mid \exists \mathbf{pq} x_1 x_2 r p\, \textsc{Fl}(x_1, n_1, \mathbf{p}) \wedge \textsc{Fl}(x_2, n_2, \mathbf{q}) \wedge \textsc{Ai}(\text{SFO}, \text{CA}, r) \wedge dir(\mathbf{p})(t_{\text{now}}) + dir(\mathbf{q})(t_{\text{now}}) = 0 \wedge r(p) \wedge \exists y_1 y_2 y_3 \mathbf{p}(t_{\text{now}}; y_1, y_2, y_3) \wedge len(p - (y_1, y_2)) \leq 300 \wedge \exists z_1 z_2 z_3 \mathbf{q}(t_{\text{now}}; z_1, z_2, z_3) \wedge len(p - (z_1, z_2)) \leq 300\}$$

Here t_{now} stands for the current time. The first line of the expression finds the locations of two flights and the airport region. The second line ensures the moving directions to be opposite to each other. The third and fourth lines are testing the distances of the flights to the airport.

3. For query q_3, the condition on the approaching angle is expressed by examining the direction vector of the flight projecting to the plane and the orientation vector of the runway. After the former is converted into a unit vector, the angle condition can be expressed as an equivalent condition on the distance. The query expression is shown below.

$$\{(x, n) \mid \exists \mathbf{p} l y_1 y_2 y_3\, \textsc{Fl}(x, n, \mathbf{p}) \wedge \textsc{Ru}(\text{LAX}, 3, l) \wedge dir(\mathbf{p})(t_{\text{now}}; y_1, y_2, y_3) \wedge len(unit(y_1, y_2) - l) \leq \sqrt{2 - \sqrt{2}} \wedge y_3 < 20000\}$$

4. In query q_4, the interesting condition is "enter". Note that it is expressed naturally as in differential calculus. The flight enters the county at time t if the flight is not above the county at every time instant just before t:

$$\{(n) \mid \exists \mathbf{p} t r\, \textsc{Fl}(\text{UA}, n, \mathbf{p}) \wedge \textsc{Co}(\text{SB}, \text{CA}, r) \wedge t_1 \leq t \leq t_2 \wedge \psi(r, \mathbf{p}(t)) \wedge \exists t'\, (t' < t \wedge \forall t''\, (t' < t'' < t \rightarrow \neg \psi(r, \mathbf{p}(t''))))\}$$

where $\psi(r, \mathbf{p}(t))$ is a formula which test if at time t the moving point $\mathbf{p}$ is in the (2-dimensional) region r. The formula ψ is similar to the corresponding part of the formula in query q_1.

5. The query q_5 expresses the condition that the flight stays in a region by explicitly stating that for every time instant it is in the region.

$$\{(x, n) \mid \exists \mathbf{p} r\, \textsc{Fl}(x, n, \mathbf{p}) \wedge \textsc{Co}(\text{LA}, \text{CA}, r) \wedge \forall t\, (t_{\text{now}} - 15 < t < t_{\text{now}} \rightarrow \exists x_1 x_2 x_3 (\mathbf{p}(t; x_1, x_2, x_3) \wedge r(x_1, x_2)))\}$$

Note that this query states a property which should be held for a period of time. It can also be expressed with temporal logic [Eme90] but the temporal operators are almost disjoint from expressions for spatial relationships and the expressions are less intuitive.

We briefly discuss evaluation of queries in our query language. Basically, queries in our language can be translated into the constraint query language of

[KKR95] augmented with relational attributes. Since the language of [KKR95] contains multiplications, it is easy to see that it gives an effective evaluation algorithm. Thus we have:

Theorem 5.3 Each calculus query can be evaluated in polynomial time (in terms of the database size).

There are several issues concerning the language and its semantics:

1. Consider query q_1 above. It pulls coordinate values out from inside a moving point and returns as results. Such a use may cause some query results to be infinite, i.e., "unsafe". This is not a problem if we extend the definition of relations to be finitely representable relations [GS97]. Another possibility is to return the entire moving point value as a value of LV type.
2. Length function computes the length of a vector. Clearly length is not linear. Uncontrolled use of the length operator would cause results to contain non-linear equations. One way to overcome this, is to treat length as an aggregate function and to limit the syntax to only allow "staged" evaluation of aggregate functions. This will guarantee the output to be representable in linear constraints, similar to [GS96].

6 Conclusions

In this paper, we present a model for moving objects using techniques from differential geometry. By studying the relationships related to moving objects and queries over them, we show that primitives of velocity and acceleration from differential geometry are expressive enough for topological, spatial, and temporal relationships, as well as for relationships for movements. We also developed a concrete data model based on linear constraint databases and a query language for moving objects.

Although our work is very preliminary, the differential geometry tools certainly yield a fresh look at the modeling and language issues for moving objects. There are many issues remaining. One issue is to fine tune the syntax of constraint query formulas to guarantee the closure property. It is also interesting to study the relationships between trajectories and in this case, it appears that curvature and torsion are useful. Introducing aggregate functions is yet another issue. An important aspect is to deal with integrals that is left out from our current model. At a more fundamental level, efficient algorithms and query optimization are the key issues to the eventual success of any models and languages.

Acknowledgements. The authors thank Yuan-Fang Wang for pointing out the relevance to differential geometry and Xianzhe Dai for giving the pointers to the differential geometry literature. The authors also benefited from discussions with Hongjun Zhu and Fang Yu.

References

[AAE00] P. K. Agarwal, L. Arge, and J. Erickson. Indexing moving points. In *Proc. ACM Symp. on Principles of Database Systems*, 2000.

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[BBC98] A. Belussi, E. Bertino, and B. Catania. An extended algebra for constraint databases. *IEEE Trans. on Knowledge and Data Engineering*, 10(5):686–705, 1998.

[BKR86] M. Ben-Or, D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. *Journal of Computer and System Sciences*, 32(2):251–264, April 1986.

[Col75] G. E. Collins. Quantifier elimination for real closed fields by cylindric decompositions. In *Proc. 2nd GI Conf. Automata Theory and Formal Languages*, volume 35 of *Lecture Notes in Computer Science*, pages 134–83. Springer-Verlag, 1975.

[EF91] M. J. Egenhofer and R. Franzosa. Point-set topological spatial relations. *Int. Journal of Geo. Info. Systems*, 5(2):161–174, 1991.

[Ege91] M. J. Egenhofer. Reasonning about binary topological relations. In *Proc. Symp. on Large Spatial Databases*, 1991.

[EGSV99] M. Erwig, R. H. Guting, M. Schneider, and M. Vazirgiannis. Spatio-temporal data types: an approach to modeling and querying moving objects in databases. *GeoInformatica*, 3(3):269–296, 1999.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 7, pages 995–1072. North Holland, 1990.

[FGNS00] L. Forlizzi, R. H. Guting, E. Nardelli, and M. Schneider. A data model and data structures for moving objects databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2000.

[GBE+00] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Varirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25(1), 2000. to appear.

[Gra98] A. Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica (Second Edition)*. CRC Press, 1998.

[GRS98] S. Grumbach, P. Rigaux, and L. Segoufin. The DEDALE system for complex spatial queries. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, June 1998.

[GS96] S. Grumbach and J. Su. Towards practical constraint databases. In *Proc. ACM Symp. on Principles of Database Systems*, 1996.

[GS97] S. Grumbach and J. Su. Finitely representable databases. *Journal of Computer and System Sciences*, 55(2):273–298, October 1997.

[KdB99] B. Kuijpers and J. Van den Bussche. On capturing first-order topological properties of planar spatial databases. In *Proc. Int. Conf. on Database Theory*, 1999.

[KGT99] G. Kollios, D. Gunopulos, and V. J. Tsotras. On indexing mobile objects. In *Proc. ACM Symp. on Principles of Database Systems*, pages 261–272, 1999.

[KKR95] P. Kanellakis, G. Kuper, and P. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, 1995.

[KLP00] G. Kuper, L. Libkin, and J. Paredarns, editors. *Constraint Databases.* Springer Verlag, 2000.

[KPdB97] B. Kuijpers, J. Paredaens, and J. Van den Bussche. On topological elementary equivalence of spatial databases. In *Proc. Int. Conf. on Database Theory*, 1997.

[KRSS98] G. Kuper, S. Ramaswamy, K. Shim, and J. Su. A constraint-based spatial extension to SQL. In *Proc. ACM Symp. Geographical Information Systems*, 1998.

[MP77] R. S. Millman and G. D. Parker. *Elements of Differential Geometry.* Prentice-Hall, Edgewood Cliffs, NJ, 1977.

[PSV96] C. H. Papadimitriou, D. Suciu, and V. Vianu. Topological queries in spatial databases. In *Proc. ACM Symp. on Principles of Database Systems*, 1996.

[Ren92] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation*, 13:255–352, 1992.

[SWCD97] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *Proc. Int. Conf. on Data Engineering*, 1997.

[Tar51] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California Press, Berkeley, California, 1951.

[WCD+98] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez. Cost and imprecision in modeling the position of moving objects. In *Proc. Int. Conf. on Data Engineering*, Orlando, FL, 1998.

[WCDJ97] O. Wolfson, S. Chamberlain, S. Dao, and L. Jiang. Location management in moving objects databases. In *Proc. the Second International Workshop on Satellite-Based Information Services (WOSBIS'97)*, Budapest, Hungary, October 1997.

[WJS+99] O. Wolfson, L. Jiang, P. Sistla, S. Chamberlain, N. Rishe, and M. Deng. Databases for tracking mobile units in real time. In *Proc. Int. Conf. on Database Theory*, pages 169–186, Jerusalem, Israel, 1999.

[WXCJ98] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: issues and solutions. In *Proc. Int. Conf. on Statistical and Scientific Database Management*, 1998.