Moving Objects and Their Equations of Motion

Floris Geerts

Helsinki Institute for Information Technology
Basic Research Unit, Department of Computer Science
University of Helsinki, Finland
floris.geerts@cs.helsinki.fi

Abstract. Moving objects are currently represented in databases by means of an explicit representation of their trajectory. However from a physical point of view, or more specifically according to Newton's second law of motion, a moving object is fully described by its equation of motion. We introduce a new data model for moving objects in which a trajectory is represented by a differential equation. A similar approach is taken in computer animation where this is known as physically based modeling. We give a query language for our data model and use techniques from physically based modeling to evaluate queries in this language.

1 Introduction

Q: What's a moving object?
A: An equation of motion.

Moving objects pose many challenges for data management systems. These challenges include the modeling and representation of moving objects, query language design, indexing techniques and query optimization. In this paper we focus on the modeling of moving objects and the design of query languages.

Existing models involve temporal logic, abstract data types[7, 9] and linear constraints [15, 21, 6]. In all these models the trajectories of the moving objects are explicitly stored in the database. Here, we take an alternative approach in which the sometimes complicated trajectories are stored in the form of preferably simpler equations of motions. This is somehow analogous to the constraint database approach [11, 12, 19] in which an infinite collection of points is stored in the form of geometric equations and inequalities.

Equations of motions are a natural way of describing moving objects. Many important motions have been described by equations since Newton published his Laws of Physics in 1687 [17]. The physics and mathematics of the equations of motion are by now well understood for large classes of moving objects like rigid bodies, deformable bodies, and so on [8, 2]. In this paper we will only consider moving point objects.

The approach of working with equations instead of trajectories is very similar to what is known as physically based modeling in computer graphics [4]. There,

moving graphical objects are defined in terms of their equations of motions and when needed, the trajectories are computed using numerical integration techniques.

Our data model requires a new query language and evaluation method. Fortunately, the generalized distance based query language proposed by Mokthar et al. [15] can be adapted to our setting. Using techniques from physically based modeling we show how these queries can be evaluated on moving object database in which only equations of motions are represented.

It is surprising that the approach proposed in this paper has not been considered before in the database community. As described in the recent survey of Agarwal et al. [1], the area of physically based modeling might have many more techniques which can be useful for the development of moving object databases and vice versa. They also point out other interesting cross-disciplinary aspects related to moving objects. However, in this paper we only explore physically based modeling from a database point of view and regard the exploration of other connections as future work.

Organization of the paper: In Section 2 we briefly describe concepts related to the physics of motion. The data model for moving objects is defined in Section 3. We then describe the physical validity of moving object databases in Section 4. The query language is defined in Section 5 and its evaluation is described in Section 6. We conclude the paper in Section 7.

2 Physics and Simulation

We start with a brief summary of concepts needed to describe the physics of motion. For a detailed description we refer to standard physics textbooks [8, 2] and the excellent SIGGRAPH tutorial of Baraff and Witkin [4]. In this paper we only consider point particles. Extensions to e.g. rigid body motions only require more physics and formulas and would deviate us too much from the aim of this paper.

Let $\boldsymbol{x}(t) = (x_1(t), \ldots, x_n(t))$ denote the location of point particle in $\mathbb{R}^n$ at time t. The velocity of the particle at time t is then given by $\boldsymbol{v}(t) = \dot{\boldsymbol{x}}(t) = \frac{d}{dt}\boldsymbol{x}(t)$. By Newton's Second Law, the motion is now fully described once we know the force $\boldsymbol{F}(t)$ acting on the particle (this force can be gravity, wind, spring forces, etc). Indeed, the motion is described by the unique solution of the (differential) equation of motion

$$\boldsymbol{F}(t) = m\frac{d^2}{dt^2}\boldsymbol{x}(t), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \boldsymbol{v}(t_0) = \boldsymbol{v}_0 \tag{1}$$

in which m denotes the mass of particle and $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ and $\boldsymbol{v}(t_0) = \boldsymbol{v}_0$ are initial conditions.

Example 1. Consider a point particle moving only along the x-axis according to the equation of motion $-\omega^2 x(t) = \frac{d^2}{dt^2}x(t)$ where $\omega \in \mathbb{R}$. Solving this equation results in $x(t) = A\sin(\omega t + \varphi)$, where A and φ are real constants determined by the initial conditions. This motion is known as the harmonic oscillator. □

In the above example, the equation of motion can be solved analytically. However, this is not always the case and one often has to rely on numerical integration techniques to obtain some information about the solution [10]. Commonly used methods include Euler's method and higher-order (adaptive) Runge-Kutta methods. Most methods take first-order differential equations as input, meaning that only partial derivatives of the first order may occur in the equation. However, this forms no restriction since higher-order differential equations can always be transformed into first-order ones by adding variables: E.g., the second-order equation (1) is equivalent to the first-order equation

$$\frac{d}{dt}\boldsymbol{X}(t) = \frac{d}{dt}\begin{pmatrix}\boldsymbol{x}(t)\\ \boldsymbol{v}(t)\end{pmatrix} = \begin{pmatrix}\boldsymbol{v}(t)\\ \boldsymbol{F}(t)/m\end{pmatrix}. \tag{2}$$

When equation (2) together with an initial condition and an initial and final point in time t_0 and t_1 is fed to these integration methods, the output is a continuous piecewise linear curve $[t_0, t_1] \to \mathbb{R}^{2n}$ which approximates the real solution to the differential equations. The real solution of a differential equation is also called an *integral curve* and is uniquely determined by the equation and the initial conditions. Moreover, by Picard's existence theorem this integral curve always exists under some mild condition on the formulas defining the equations (the so-called Lipschitz condition). We will assume that these conditions are always satisfied.

3 A Model for Moving Objects

In this section we introduce a data model for moving objects. Unlike earlier models for moving objects that use ADTs [7, 9] or linear constraints [15, 21, 6], we do not store the complete trajectories of the moving objects, but store the equations of motion to which the trajectories are the solution.

We will represent geometric objects by means of linear constraints. A *linear constraint* over variables $x_1, \ldots, x_n$ has the following general form: $\sum_{i=1}^{n} a_i x_i \theta a_o$, where $a_0, a_1, \ldots, a_n$ are integer numbers and θ is an order predicate ($=, <, \leq, >, \geq$). Constraints are interpreted over the real numbers. We use a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ to denote a point in space. On top of the ordinary variables we also have the time variable t.

The position of a moving point can be modeled by a function of time, represented by the real line $\mathbb{R}$, to the n-dimensional space $\mathbb{R}^n$. A function from $\mathbb{R}$ to $\mathbb{R}^n$ is *linear* if it has the form $\boldsymbol{x} = \boldsymbol{a}t + \boldsymbol{b}$ where $\boldsymbol{a}, \boldsymbol{b}$ are vectors in $\mathbb{R}^n$; A function is *piecewise linear* if it consists of a finite number of linear pieces, i.e., if it has the form

$$\boldsymbol{x} = \begin{cases} \boldsymbol{a}_1 t + \boldsymbol{b}_1 & \text{if } t_0^{(1)} \leq t \leq t_1^{(1)} \\ \quad\vdots & \qquad\vdots \\ \boldsymbol{a}_k t + \boldsymbol{b}_k & \text{if } t_0^{(k)} \leq t \leq t_1^{(k)}, \end{cases}$$

where $t_1^{(i)} \leq t_0^{(i+1)}$ for all $i = 1, \ldots, k$.

Definition 1. A *trajectory* is a piecewise linear function from $\mathbb{R}$ to $\mathbb{R}^n$. Let $\mathcal{T}$ be the set of all trajectories. □

We also introduce the *derivative variables* $\dot{x}_1, \ldots, \dot{x}_n$. Although we will consider them as an ordinary variable, the semantics of a derivative variable $\dot{x}$ is the derivative of x (as a function of time) with respect to the time variable. A *differential constraint* over the variables $x_1, \ldots, x_n, \dot{x}_1, \ldots, \dot{x}_n$ has the following general form:

$$\dot{x}_i = f_i(x_1, \ldots, x_n, t), \quad i = 1, \ldots, n$$

where f_i is a multi-variate polynomial with integer coefficients in the variables $x_1, \ldots, x_n$ and t.

Example 2. A differential constraint corresponding to the harmonic oscillator of Example 1 is

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2, t) = x_2 \\ \dot{x}_2 &= f_2(x_1, x_2, t) = -\omega^2 x_1. \end{aligned}$$

□

As mentioned in Section 2, a differential constraint does not completely specify an integral curve. Also *initial constraints* on variables x_i and t are needed. We only consider initial constraints of the form

$$I(c_1, \ldots, c_n, t_0) \equiv \bigwedge_{i=1}^{n} x_i = c_i,$$

where the c_is and t_0 are real numbers. Hence, an initial constraint specifies a point in $\mathbb{R}^{n+1}$.

Example 3. Consider the differential constraint given in Example 2. We know already from Example 1 that every integral curve is of the form $(x_1(t), x_2(t))$ where $x_1(t) = A\sin(\omega t + \varphi)$ and $x_2(t) = \dot{x}_1(t)$. Let $c_1, c_2 \in \mathbb{R}$ and consider the initial constraint $I(c_1, c_2, 0)$. The integral curve corresponding to this initial constraint is then uniquely defined by taking $A = \sqrt{(\frac{c_2}{\omega})^2 - c_1^2}$ and $\varphi = \arcsin(\frac{c_1}{\omega})$. □

Definition 2. An n-dimensional *equation of motion* consists of a finite number of triples $(I^{(i)}, DE^{(i)}, \tau^{(i)})$, where for each $i = 1, \ldots, k$,

- $I^{(i)}$ is an initial constraint. I.e., $I^{(i)} = I^{(i)}\left(c_1^{(i)}, \ldots, c_n^{(i)}, t_0^{(i)}\right)$, where $t_0^{(i)} < \tau^{(i)} \leqslant t_0^{(i+1)}$;
- $DE^{(i)}$ is a differential constraint. I.e., $DE^{(i)}$ is equal to $\dot{x}_j = f_j^{(i)}(t, x_1, \ldots, x_n)$, for $j = 1, \ldots, n$; and
- $\tau^{(i)}$ is the final time for which the equation of motion $DE^{(i)}$ is valid.

We denote the set of all n-dimensional equations of motion by $\mathcal{E}$. □

An equation of motion consisting of k initial and differential constraints, corresponds naturally to k integral curves which describe the motion of point during disjoint or adjacent intervals in time.

Example 4. Let DE be the differential constraint given in Example 2. Consider the equation of motion consisting of

$$\{(I(c_1, c_2, 0), DE, 1), (I(d_1, d_2, 1), DE, 2), (I(e_1, e_2, 3), DE, \infty)\}.$$

This equation of motion then corresponds to the description of a moving point using the integral curves in $(x, \dot{x})$-space:

$$t : [0, 2] \cup [3, \infty) \rightarrow \begin{cases} (A_1 \sin(\omega t + \varphi_1), A_1 \omega \cos(\omega t + \varphi_1)) & \text{if } 0 \leq t \leq 1 \\ (A_2 \sin(\omega t + \varphi_2), A_2 \omega \cos(\omega t + \varphi_2)) & \text{if } 1 \leq t \leq 2 \\ (A_3 \sin(\omega t + \varphi_3), A_3 \omega \cos(\omega t + \varphi_3)) & \text{if } 3 \leq t, \end{cases}$$

where the constants A_i and φ_i are determined by the initial constraints as shown in Example 3. We have depicted an instantiation of this example in Figure 1. □

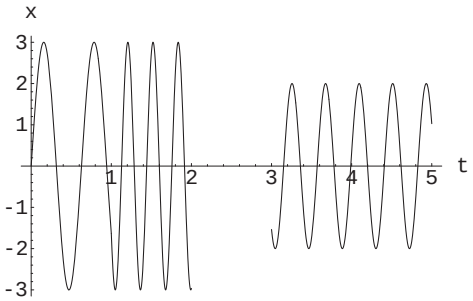

Fig. 1. The integral curves corresponding to the equation of motion given in Example 4 (Only the (x, t)-coordinates are shown).

The definition of equation of motion does not rule out discontinuities in the transition between different integral curves. E.g., in Figure 1 the integral curve disappears between $t = 2$ and $t = 3$. In some applications it might be desirable that integral curves smoothly glue together at transition points. We remark that Definition 2 can easily be restricted to this setting.

We now define a mapping PL from the set of equations of motions $\mathcal{E}$ to the set of trajectories $\mathcal{T}$. With each $e \in \mathcal{E}$ we associate the piecewise linear trajectory $PL(e) \in \mathcal{T}$ obtained by applying a numerical integration method. This mapping is clearly dependent on the choice of numerical integration technique. In this section we will use *Euler's method* [18]. This might not be the best method currently available [10], but suffices for explaining the ideas in this section. We

define the mapping PL in more detail now. Consider first the case that the equation of motion consists of a single initial and differential constraint.

Applied to an initial and a differential constraint I and DE, Euler's method takes a fixed small step size h, sets $\boldsymbol{x}(t_0) = \boldsymbol{c}$, as demanded by the initial constraint and then defines

$$\boldsymbol{x}(t_{i+1}) = \boldsymbol{x}(t_i) + \boldsymbol{f}(\boldsymbol{x}(t_i), t_i)h,$$

where $t_i = t_0 + ih$ for $i = 1, 2, \ldots$ and $\boldsymbol{f} = (f_1, \ldots, f_n)$ are the functions occurring in the differential constraint DE.

One then obtains a trajectory in $\mathcal{T}$ by linearly interpolating between consecutive points. More specifically, if τ is the final time for the differential constraint to be valid, then let K be the maximum integer such that $t_0 + Kh < \tau$ and we define the trajectory $PL((I, DE, \tau))$ as

$$t : (t_0, \tau) \to \begin{cases} \boldsymbol{x}(t_0) + \boldsymbol{f}(\boldsymbol{x}(t_0), t_0)t & \text{if } t_0 \le t \le t_1 \\ \vdots & \vdots \\ \boldsymbol{x}(t_K) + \boldsymbol{f}(\boldsymbol{x}(t_K), t_K)t & \text{if } t_K \le t \le \tau, \end{cases} \tag{3}$$

where again $t_i = t_0 + ih$ for $i = 0, 1, \ldots, K$.

In general, when an equation of motion consists of several triples (I, Eq, τ), the mapping PL is the union of the mappings PL on the individual triples.

We would like to have that if the integral curve is already a trajectory, then the PL mapping will return the same trajectory as well.

Example 5. Consider a point moving along the x-direction between initial time $t_0 = 0$ and final time $t_2 = 1$ such that

$$x(t) = \begin{cases} 2t & \text{if } 0 \le t \le \frac{1}{2} \\ 1 - 2t & \text{if } \frac{1}{2} \le t \le 1. \end{cases} \tag{4}$$

Let $I^{(1)} \equiv x = 0 \wedge t = 0$ and $I^{(2)} \equiv x = 1 \wedge t = \frac{1}{2}$. Furthermore, consider the differential constraints

$$DE^{(1)} \equiv \dot{x} = 2, \qquad DE^{(2)} \equiv \dot{x} = -2.$$

Let e be the equation of motion consisting of the initial constraints $I^{(1)}$ and $I^{(2)}$, together with the corresponding differential constraints $DE^{(1)}$ and $DE^{(2)}$, and final time points $\tau^{(1)} = \frac{1}{2}$ and $\tau^{(2)} = 1$. The trajectory $PL(e)$ is the union of $PL(I^{(i)}, DE^{(i)}, \tau^{(i)})$, $i = 1, 2$ each of which defined by (3). Euler's method on e will now return the same trajectory as represented by (4). □

It is easy to prove that the observation made in the last example holds in general.

Proposition 1. *Let $\boldsymbol{\gamma} : [t_0, t_1] \to \mathbb{R}^n$ be a piecewise linear curve. Then there exists an equation of motion $Eq \in \mathcal{E}_n$ such that $PL(Eq) = \boldsymbol{\gamma}$.* □

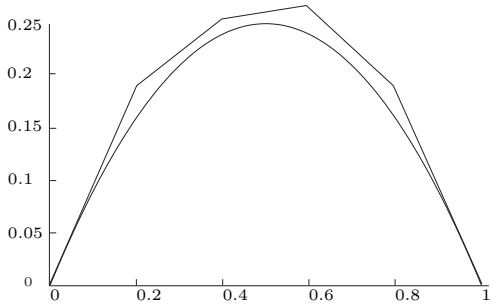

Fig. 2. Integral curve (parabola) of the moving object in Example 6 and corresponding trajectory (linear curve) obtained using a rather large stepsize $h = 0.2$. (Only the (x, y)-coordinates are shown.)

As mentioned above, Euler's method is not the most desirable integration method since it does not provide guarantees on the accuracy. This is common to most step-wise classical numerical integration techniques [20]. In contrast, interval methods produce guaranteed error bounds [16]. They use interval arithmetic to calculate each approximation step, explicitly keeping the error bounds within safe interval bounds. We leave it to future work to integrate interval methods in our framework.

We now have all the ingredients for defining a moving object database. Let $\mathcal{O}$ denote an infinite set of object identifiers.

Definition 3. A n-dimensional *moving object database* is a triple (O, Eq, τ) where O is a finite subset of $\mathcal{O}$, Eq is a mapping from O to $\mathcal{E}$, and τ is a time instance such that each interval in $Eq(o)$ for every object $o \in O$ ends earlier than or at time τ. □

Proposition 1 says that this definition of moving object databases generalizes earlier definitions in which trajectories are explicitly represented by piecewise linear functions. [15, 21]. However, the equation of motion in Proposition 1 requires in general the same number of initial and differential constraints as the number of linear segments of the trajectory and our data model is therefore not necessarily more compact. This changes however when the integral curves are not linear anymore. In these cases a single initial and differential constraint can represent a trajectory consisting of many linear segments.

Example 6. Let e be the equation of motion consisting of an initial constraint $x = 0 \wedge y = 0 \wedge \dot{v} = 1 \wedge \dot{w} = 1 \wedge t = 0$, together with the differential constraint

$$\dot{x} = v, \quad \dot{y} = w, \quad \dot{v} = 0, \quad \dot{w} = -g,$$

and a final point in time $\tau = 1$. A simple computation shows that the solution of this equation of motion is

$$t \mapsto (t, -gt^2 + t, 1, -2gt + 1),$$

which restricted to the (x, y)-plane results in a parabola. By definition PL will map this equation of motion to a linear approximation of the parabola. We have depicted this situation in Figure 2. We used a large stepsize in Euler's method in order to distinguish the integral curve from its approximating trajectory. □

We remark that our definition of differential constraints often demands the introduction of new variables. It might be therefore desirable to make explicit in which dimension of the integral curve one actually is interested in (e.g., three spatial dimensions and time). This can be easily added to the definition of equations of motion.

4 Physical Correctness

Updates for moving objects are of particular importance due to their dynamic character. We therefore need to be able to update moving object databases and the following pair of updates adapted from [15] clearly suffices.

Definition 4. Let τ be a time point, $o \in \mathcal{O}$, $e \in \mathcal{E}_n$. An *update* on a moving object database (O, Eq, τ) is one of the following:

- Create a new object: $\mathsf{new}(o, \tau', e)$ results in $(O \cup \{o\}, Eq', \max\{\tau, \tau'\})$ where Eq is identical to Eq except for $Eq(o) = e$.
- Terminate an existing object: $\mathsf{terminate}(o, \tau')$ results in (O, Eq', τ) where Eq' is identical to Eq except for $Eq'(o) = Eq(o) \wedge t \leq \tau'$. □

In addition to the usual demands for updates, the physical meaning of the equations of motions also demands the presence of update operators on the moving object databases. Indeed, consider two moving objects o_1 and o_2 whose integral curves intersect at time t_c. We say that o_1 and o_2 are in contact at time t_c. At this contact time t_c the initial and differential constraints should change according to physical laws in order to represent a physical reality. We call a moving object database *physically consistent* if at times of contact the equations of motions have been updated correctly. We will not give explicitly the formulas needed for the updates but refer instead to the tutorial of Baraff for derivation and explanation of the updates [4].

One distinguishes between two types of contact. When o_1 and o_2 are in contact at time t_c and they have a velocity towards each other, we speak of a *colliding contact*. Colliding contacts require an instantaneous change in velocity of the objects. This implies that at time t_c the differential constraint still holds but the initial constraint needs to be updated.

When o_1 and o_2 are in contact at time t_c and o_2 is resting or cannot move, e.g. in case o_2 is a wall, then we speak of a *resting contact*. In this case, one needs to compute a contact force and has to change the differential constraint accordingly in the equation of motion of o_1 for $t \geq t_c$.

Note that we assumed that the contact times t_c are known. Collision detection algorithms provide a way of computing (or approximating) these intersection points. It has been extensively studied in computational geometry and

robotics [13] and currently fast and robust algorithms exists. However, most of these algorithms are practical for a small number of moving objects. Also it is difficult when moving objects are geometric objects other than points and are deformable. We refer to the survey on modeling of motion for the current state of research [1].

5 Query Language

Now that we have defined the data model, it is time to look for a query language. We first give the kind of queries which we would like to ask and answer. A very natural example of a moving database is an interplanetary database containing moving planets, galaxies, space shuttles, satellites and so forth. We are now interested in the following kind of queries.

Q_1 Locate all planets in the earth's North hemisphere at 2pm.
Q_2 Find all comets entering our solar system today.
Q_3 List all pairs of satellites flying in opposite direction.
Q_4 List all moons of planets in our solar system which are visible at least 10 minutes today.
Q_5 Give all pairs of planets and asteroids which are in collision course.

In order to ask and answer these queries we will use the query languages FO(f) based on a generalized distance f as defined by Mokthar et al. [15].

Let Γ denote the set of all continuous functions from $\mathbb{R}$ to $\mathbb{R}^n$.

Definition 5. A *generalized distance* is a mapping from the set Γ of continuous curves to continuous functions from $\mathbb{R}$ to $\mathbb{R}$. Let a (O, Eq, τ) be a moving database and f a generalized distance. For each $o \in O$ we set $\mathsf{f}_o = \mathsf{f}(\boldsymbol{\gamma})$, where $\boldsymbol{\gamma}$ is the integral curve corresponding to $Eq(o)$. Moreover, we define $\hat{\mathsf{f}}_o = \mathsf{f}(PL(Eq(o)))$. □

So, a generalized distance describes some property of curves which continuously changes in time.

The language FO(f) consists of a many-sorted first-order logic with real numbers, time instants, and objects. The language uses a single time variable t and many objects variables. We do not allow real variables; these are embedded in the generalized functions.

- *time terms* are polynomials over the time variable t with integer coefficients.
- *real terms* include real variables, and $\mathsf{f}(y, t)$ where y is an object variable and t is a time term.

Atomic formulas are formed by equality and order predicates over terms of the same sort. Formulas are then constructed by propositional connectives and universal/existential quantifiers over object variables.

Definition 6. A query is a quadruple (y, t, I, φ) where y is an object variable, t a time variable, I a time interval and φ a formula with only y and t free. □

Let $D = (O, Eq, \tau_0)$ be a moving object database. Then for each time τ, we define

$$Q[D]_\tau = \{(o) \mid o \in O \wedge \varphi(o, \tau)\}.$$

The answer to Q can then be of an *existential* nature, $Q^\exists(D) = \{(o) \mid \exists t(t \in I \wedge o \in Q[D]_t)\}$, or *universal*, $Q^\forall(D) = \{(o) \mid \forall t(t \in I \to o \in Q[D]_t)\}$. It is clear that both $Q^\exists$ and $Q^\forall$ can be obtained from Q_τ. It is easy to see that queries $Q_1, \ldots, Q_5$ can all be expressed in FO(f).

6 Evaluating the Queries

The evaluation procedure of FO(f) queries in our database model is an adaptation of the procedure given by Mokthar et al. [15]. Let $Q = (y, t, I\varphi)$ be a query and D a moving object database. Mokhtar et al. showed that in order to evaluate $Q[D]_\tau$ it is sufficient to know at each time $t \in I$ the precedence relation $\leq_t$ defined by

Definition 7. Let $D = (O, Eq, \tau')$ be a moving object database, τ a time instant, $o, o' \in O$. The object o *precedes* o' at time τ, denoted by $o \leq_\tau o'$, if $\mathsf{f}_o(\tau) \leq \mathsf{f}_{o'}(\tau)$. □

If the moving objects are represented by trajectories, then the precedence relation can be easily obtained as shown in [15].

In our data model, we have a larger class of trajectories consisting of the integral curves of the equations of motion. Since we do not have the integral curves at our disposal we will replace them for the time period I by the trajectories obtained by applying the PL map defined in Section 3.

Definition 8. Let $D = (O, Eq, \tau')$ be a moving object database, τ a time instant, $o, o' \in O$. The object o *approximately precedes* o' at time τ, denoted by $o \preceq_\tau o'$, if $\hat{\mathsf{f}}_o(\tau) \leq \hat{\mathsf{f}}_{o'}(\tau)$. □

Instead of answering Q based on the precedence relation $\leq_\tau$ we will answer it based on the approximate precedence relation $\preceq_\tau$. In order to guarantee a good approximation we only have to be sure that the intersection points of the curves

$$\gamma : t \in I \to \mathsf{f}_o(t), \quad o \in O$$

are accurately approximated by the intersection points of

$$\gamma : t \in I \to \hat{\mathsf{f}}_o(t), \quad o \in O,$$

since these determine possible changes in the precedence relations.

If we simply take the PL map defined in Section 3 based on Euler's method then this will not be the case since this method has fixed time step size h and will therefore make wrong conclusion concerning the intersection of the above curves. Using standard approaches from physically based modeling, we can however change step size in the neighborhood of intersection points of the above curves.

Two common approaches exists. In retroactive detection [3] one checks after each integration step for intersection points. If an intersection is detected, one traces back the exact moment of intersection using e.g., bisection methods or regular falsa, and the integration starts again from the intersection moment. On the other hand, conservative advancement methods [5] creep up to intersection methods, taking smaller steps as it gets closer. For our purposes it does not really matter which method is used. It only matters when there are a vast number of moving objects in the database. In this case one can rely on the Timewarp algorithm [14].

The above methods are assumed to be part of the PL mapping so that the approximate precedence relation captures the exact precedence relations as good as possible. Using this approximate precedence relation, a query in FO(f) can be evaluated as described by Mokthar et al. [15].

7 Conclusions

We proposed a new data model for moving object based on equations of motions and showed that the query language proposed by Mokthar et al. [15] can be used also in our setting. The evaluation of queries is however different since trajectories are not stored in the database but are generated in real-time when a query is asked. These trajectories can be generated with an adjustable degree of accuracy depending on the database instance. Using techniques from physically based modeling we're able to guarantee a successful answer to these queries. Many aspects still need to be investigated, including

- The complexity of the computation of the approximate precedence relation, and more general the complexity of the query evaluation described in Section 6.
- The generalization of the moving point objects to general moving objects using results from [4]. This involves a more complex definition of differential constraints and update operators. A challenge is to extend the generalized distance-base query language FO(f) so that it can query the spatial extent of the moving object as well.
- Data integration of our data model. It is important that the proposed data model integrates well with other spatio-temporal models.
- The automatic derivation of simple equations of motion from observed trajectories.

Acknowledgement

The author would like to thank Taneli Mielikäinen for his encouraging remarks.

References

1. P. K. Agarwal, L.J. Guibas, H. Edelsbrunner, J. Erickson, M. Isard, S. Har-Peled, J. Hershberger, C. Jensen, L. Kavraki, P. Koehl, M. Lin, D. Manocha, D. Metaxas,

B. Mirtich, D. Mount, S. Muthukrishnan, D. Pai, E. Sacks, J. Snoeyink, S. Suri, and O. Wolefson. Algorithmic issues in modeling motion. *ACM Comput. Surv.*, 34(4):550–572, 2002.

2. V. I. Arnold, A. Weinstein, and K. Vogtmann. *Mathematical Methods of Classical Mechanics*, volume 60 of *Graduate Texts in Mathematics*. Springer-Verlag, 1997.
3. D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics*, 24(4):19–28, 1990.
4. D. Baraff and A. Witkin. *Physically Based Modelling*. Pixar Animation studios, 2001. SIGGRAPH 2001 Course Notes.
5. J. Basch, L.J. Guibas, and J. Hershberger. Data structures for mobile data. *Journal of Algorithms*, 31(3):1–28, 1999.
6. J. Chomicki and P. Revesz. Constraint-based interoperability of spatiotemporal databases. *Geoinformatica*, 3(3):211–2423, 1999.
7. M. Erwig, R.H. Güting, 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.
8. R.P. Feynman, R.B. Leighton, and M Sands. *The Feynman Lectures on Physics*. Addison-Wesley, 1989.
9. R. H. Güting, M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Trans. Database Syst.*, 25(1):1–42, 2000.
10. A. Iserles, editor. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press, 1996.
11. P. C. Kanellakis, G. M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computing and Systems Sciences*, 51(1):26–52, 1995.
12. G.M. Kuper, J. Paredaens, and L. Libkin, editors. *Constraint Databases*. Springer-Verlag, 2000.
13. M. Lin and S. Gottschalk. Collision detection between geometric models. In *Proceedings of the IMA Conference on Mathematics of Surfaces*, 1998.
14. B. Mirtich. Timewarp rigid body simulation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 193–200, 2000.
15. H. Mokthar, J. Su, and O. Ibarra. On moving object queries. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 188–198. ACM Press, 2002.
16. N.S. Nedialkov. *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation*. PhD thesis, Computer Science Dept., University of Toronto, 1999.
17. I. Newton. *Philosophiae Naturalis Principia Mathematica*. 1687.
18. W.H. Press, Flannery B.P., Teukolsky S.A., and Vetterling W.T. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 1992.
19. P. Revesz. *Introduction to Constraint Databases*. Springer-Verlag, 2001.
20. L.F. Shampine. *Numerical Solution of Ordinary Differential Equations*. Chapman & Hall, 1994.
21. J. Su, H. Xu, and O.H. Ibarra. Moving objects: Logical relationships and queries. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, volume 2121 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2001.