Chapter 5
Trajectory Data Models

J. Macedo, C. Vangenot, W. Othman, N. Pelekis, E. Frentzos, B. Kuijpers, I. Ntoutsi, S. Spaccapietra, and Y. Theodoridis

5.1 Introduction

Trajectory databases is an important research area that has received a lot of interest in the last decade. The objective of trajectory databases is to extend database technology to support the representation and querying of moving objects and their trajectory.

Moving objects are geometries, which may be points, lines, areas or volumes, changing over time. A trajectory consists in the description of the movement of those objects. A strict definition of 'movement' relates it to change in physical position. Physical movement implies an object and a reference system within which one can assess positions. Most frequently, the reference system is geographical space and we speak about objects moving in space, therefore, about trajectories of objects in space. As geographical space per se is continuous, physical movement is described by a continuous change of position, i.e. a function from time to geographical space. Movement also implies a temporal dimension as we can only perceive movement through comparison at two different instants. Therefore, a trajectory can be equivalently defined as the record of a time-varying spatial phenomenon.

Objects may move/change at specific instants in time, without any existence or any knowledge of their existence in between. A duck suddenly disappears from your perception and reappears somewhere nearby at a later moment. In these cases, movement is perceived as neither continuous nor stepwise, but a collection of separate instants or intervals. The question *what is a moving object?* can be answered tautologically as an object that moves. An object is an identifiable real-world element that may be perceived as having an existence dissociated from that of other objects. A person and a car are obvious examples of potential objects. An object that moves is an object that is not constrained to keep the same position during its

J. Macedo
Database Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland,
e-mail: jose.macedo@epfl.ch

whole existence. Objects that move become particularly interesting when we record their trajectory. Hence, hereinafter we restrict the term moving object to denote an object to which we can associate a trajectory.

Although a trajectory can be quite simply defined as a function from time to geographical space, its description, representation and manipulation happen to be more complex. Indeed, from an application point of view, a trajectory is the record of the movement of some object i.e. the record of the positions of the object at specific moments in time. Thus, although we naturally think of a nice curve representing the trajectory of the object, in reality the trajectory has to be built from a set of sample points, i.e. the positions of the object. And the nice curve is obtained by applying interpolation methods on the set of sample points. To find the more suitable curve connecting the sample points, various interpolation methods have been proposed. However, whichever interpolation method is used, the resulting curve will only be a guess of the probable trajectory. This guess is even worse when considering the possible measurement errors that inevitably happen when recording the original points. There is thus an inherent uncertainty associated to trajectories, which, depending on the cause, is either measurement or interpolation uncertainty. To model and manage adequately uncertainty, different interpolation methods and modelling concepts have been proposed. They are presented in Sect. 5.2.

Trajectory data modelling has received a lot of attention from the research community either from researchers applying existing spatiotemporal data models to trajectory data or from researchers proposing new models specifically dedicated to moving objects and their trajectories. Indeed simply considering trajectories as a function from time to geographical space, existing spatiotemporal models can be used to model trajectories. Those models, presented in more detail in Sect. 5.3.1, usually represent trajectories as time-varying geometry.

Another trend of research has considered constraint database models to represent trajectories. Indeed since trajectories can be seen as a collection of infinite points connecting a finite number of sample points, constraint database models can be specialized to represent moving objects and their trajectory (see Sect. 5.3.2).

Starting from this constatation that neither existing spatiotemporal models nor constraint database models were perfectly adapted for trajectory modelling, a parallel line of research focussing on modelling moving objects as well as supporting location-aware queries has emerged. Those works are presented in Sect. 5.3.3.

Even if quality work on moving objects exists, there are still many open issues regarding conceptual modelling of trajectories, uncertainty, multiple representation of trajectories, continuously acquired trajectories, and query capabilities. Those open issues are described in Sect. 5.4.

5.2 Basic Concepts: From Raw Data to Trajectory

For modelling concepts to represent trajectories in databases, basic concepts of trajectory data need to be presented. This is the objective of this section, where a more formal definition of a trajectory is first proposed. Then, through the description of different interpolation methods, the process of building a trajectory from the set of positions of real-world objects will be discussed. Finally, several methods to cope with uncertainty, an inherent component of trajectories, will conclude the section.

5.2.1 *What Is a Trajectory?*

In Sect. 5.1, we presented an intuitive definition of a trajectory as the description of the movement of some object. More formally, a trajectory T is the graph of a continuous mapping from $I \subseteq \mathbb{R}$ to $\mathbb{R}^2$ (the two-dimensional plane)

$$I \subseteq \mathbb{R} \to \mathbb{R}^2 : t \mapsto \alpha(t) = (\alpha_x(t), \alpha_y(t))$$

now $T = \left\{(\alpha_x(t), \alpha_y(t), t) \mid t \in I\right\} \subset \mathbb{R}^2 \times \mathbb{R}$

5.2.2 *From Sample Points to Trajectories*

The first and foremost restriction is of course that a trajectory connected to a data sample should contain the sample points, i.e. for all points (x_i, y_i, t_i) in the sample we have $(x_i, y_i, t_i) = (\alpha_x(t_i), \alpha_y(t_i), t_i)$. It is rather trivial to remark that if our sample points are ordered in time, i.e. if $i < j$ then $t_i < t_j$, then this order will be preserved along the trajectory.

Second, given a data sample, there is an infinite number of trajectories connected to that data sample. The trajectory is by no means unique. Finding a suitable curve connecting the 'dots', the sample points, is called interpolation.

5.2.2.1 Interpolating the Sample Points

Interpolation brings along its own problems. We wish it to be fast, easily manageable, flexible and accurate. Unfortunately, improving one property does not necessarily improve another. And as we will see, more often than not, these properties counteract each other.

Linear interpolation (Fig. 5.1) is the fastest and easiest of them all. The idea is to connect the sample points with straight lines, the linearity is expressed in the fact that equal jumps in time (between the same sample points) lead to equal jumps in space. For example, the segment between the points (x_i, y_i, t_i) and $(x_{i+1}, y_{i+1}, t_{i+1})$ is given by

$$(x, y, t) = (x_i, y_i, t_i) + \frac{t - t_i}{t_{i+1} - t_i}(x_{i+1} - x_i, y_{i+1} - y_i, t_{i+1} - t_i),$$

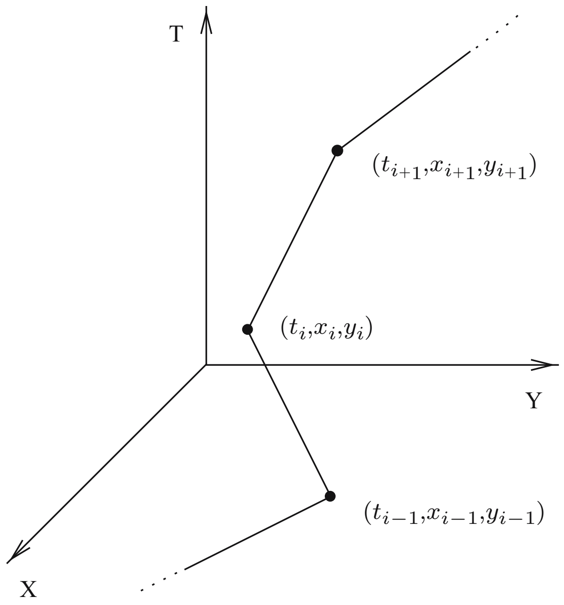

Fig. 5.1 Linear interpolation

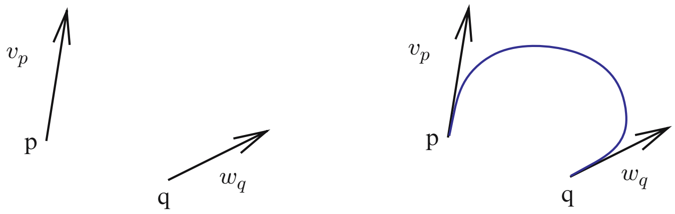

Fig. 5.2 Interpolation with Bezier curves

this is a straight line segment in $\mathbb{R}^2 \times \mathbb{R}$ parameterized by $t \in [t_i, t_{i+1}]$. The trajectory consists then of the concatenation of all these segments.

Interpolation in this manner is not so innocent, along the way some assumptions have been made. The first one is that the moving object has constant speed and direction between the sample points. Moreover, this speed is the minimal average speed needed to cover the distance between (x_i, y_i) and (x_{i+1}, y_{i+1}) in time $t_{i+1} - t_i$.

Second, changes in speed and direction at sample points are often abrupt and discontinuous, because of the sharp corners of the trajectory at the sample points. Note that the trajectory is continuous, but its speed and direction is not.

Third, it is fast. Fast to construct and to handle. Computing intersections, eliminating quantifiers (see the constraint model below) is 'easy' when we only consider objects described by linear equations (inequalities).

Interpolation with Bezier curves (Fig. 5.2) lends itself much better to create smoother curves. Given two sample points and a velocity vector, i.e. direction and speed, in each sample point, a Bezier curve is a curve where each spatial coordinate is a third-degree polynomial of the time coordinate. The beginning and end point are exactly the respective sample points. The vectors in these sample points are precisely the velocity vectors to the curve in those points.

Bezier curves are fast to construct. Transitions over sample points are nice and smooth. The trajectory is everywhere differentiable. The downside is that it is a lot harder to handle. For example, computing distance along the trajectory and computing intersections with other trajectories become much less trivial task.

Plus, you need much more information to construct this trajectory. You need to know the object's speed and direction at each sample point. If those are unknown, one can still make educated guesses. Use the average direction taken from the direction from the previous to the current sample point and from the current to the next sample point. One can make similar guesses for the object's speed using the minimal average speed needed to get from the previous to the current sample point and from the current to the next sample point.

All these interpolation methods have one thing in common. The more sample points there are, i.e. the closer they are in time, the more accurate the trajectory will be. The two methods mentioned earlier will converge to the same trajectory when you increase the frequency of the sample points.

5.2.3 Uncertainty

Interpolation will only give you a guess of a probable trajectory and that guess leaves a certain amount of uncertainty about the chosen trajectory. This first kind of uncertainty will be referred to as *interpolation uncertainty*.
In the literature, uncertainty has been defined as the measure of the difference between the actual contents of a database, and the contents that the current user or application would have created by direct and perfectly accurate observation of reality [78]. Sources of uncertainty may be one of the following:

- Imperfect observation of the real world
- Incomplete representation language
- Ignorance, laziness or inefficiency

Interpolation uncertainty may be seen as a result of the two last points. Interpolation uncertainty can be managed with beads. The bead model works under the assumption that we know an upper bound for the object's speed in between sample points, and also that the position in the sample points is an exact position, although the latter can be tackled easily. As for the upper bound on the object's speed, the maximum speed limit of the area the object covers can be used for example.

Suppose the object's maximal speed is v, and that it travels from (x_i, y_i) at time t_i and arrives at (x_{i+1}, y_{i+1}) at time t_{i+1}. At any time $t \in [t_i, t_{i+1}]$, the distance of the object, at position (x, y), to (x_i, y_i) will be at most $v(t - t_i)$. This means that at any time $t \in [t_i, t_{i+1}]$, the object is somewhere in a disc with centre (x_i, y_i) and radius

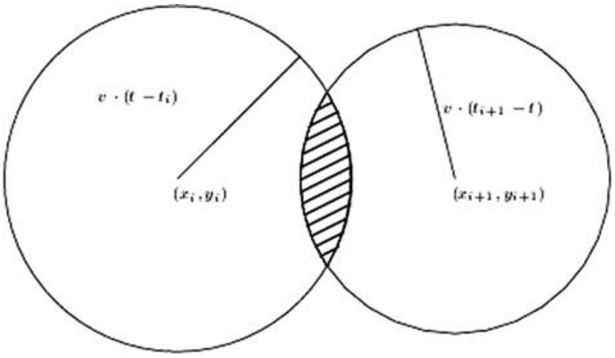

Fig. 5.3 The uncertainty area at time t

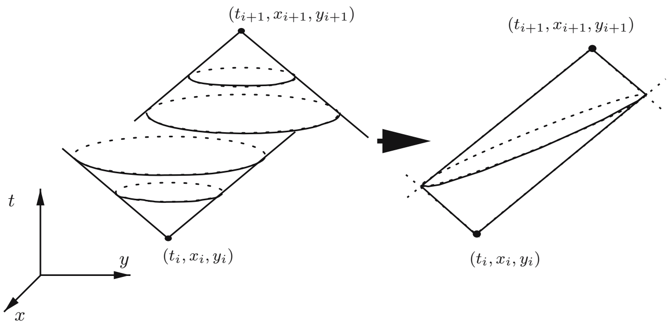

Fig. 5.4 An uncertainty bead

$v(t - t_i)$. Furthermore, in space–time this a cone, with its top in (x_i, y_i, t_i), an axis of symmetry parallel to the time axis and pointing backward in *time*.

At the same time $t \in [t_i, t_{i+1}]$, the object, at position (x, y), has to reach (x_{i+1}, y_{i+1}) in time $(t_{i+1} - t)$. That means its distance to (x_{i+1}, y_{i+1}) can be at most $v(t_{i+1} - t)$ and again the object is somewhere in a disc with centre (x_{i+1}, y_{i+1}) and radius $v(t_{i+1} - t)$. Similarly, this is a cone in space–time, but this time pointing forward in time.

So at any time $t \in [t_i, t_{i+1}]$ the object must be somewhere in the intersection of two discs as can be seen in Fig. 5.3. Or, more generally speaking, a point (x, y, t) *might* belong to a trajectory going from (x_i, y_i, t_i) to $(x_{i+1}, y_{i+1}, t_{i+1})$ if and only if it lies in the intersection of the cones as can be seen in Fig. 5.4.

The geometric object in space–time in Fig. 5.4 is called a (lifeline) bead [19]. Projecting this bead on the XY-plane yields to an ellipse with foci's in (x_i, y_i) and (x_{i+1}, y_{i+1}). This is easy to see, the distance between (x_i, y_i) and the object with coordinates (x, y) is *at most* $v(t - t_i)$, and that the distance to (x_{i+1}, y_{i+1}) is at most $v(t_{i+1} - t)$. Adding those two distances equals $v(t_{i+1} - t_i)$, which is constant and independent from t, for all $t \in [t_i, t_{i+1}]$. That means that the sum of the distances to (x_{i+1}, y_{i+1}) and (x_i, y_i) is at most a constant number $v(t_{i+1} - t_i)$ and that the geometric set of such points is, therefore, by definition the area bounded by an ellipse.

Beads are not easy to handle. To reduce the complexity of evaluating certain queries, minimal bounding circular and elliptic cylinders and even minimal bounding rectangles are used. Note that to manage uncertainty in this manner, we need a way to determine the maximal speed of the object.

Another kind of uncertainty needs to be considered, namely the kind introduced by measurement errors. Sensors introduce errors, e.g. the measurement error with GPS. We will refer to this kind of uncertainty as *measurement uncertainty.* A model that captures both kinds of uncertainty is described by [71].

In this model an *uncertainty threshold* is introduced. This threshold denotes the maximal distance of the object to the assumed location on the trajectory. After linear interpolation, this model assigns to each point on the trajectory a disc, parallel to the XY-plane, of radius equal to the threshold. Taking all those discs together in three-dimensional space–time results in a tube around the polyline connecting the sample points.

This threshold incorporates interpolation uncertainty and measurement errors all at once. It does not discriminate sample points from interpolated points, though we only need to take the measurement error into account when it comes to the sample points.

Now assume that the threshold is chosen so that the trajectory volume is a minimal bounding volume for the beads of the same sample points, then beads reduce uncertainty roughly by a factor 3, since a cone has one-third the volume of its minimal bounding circular cylinder.

However, considering the structure of the trajectory volumes, i.e. circles parallel to the XY-plane, these structures are much easier to handle computationally. For example, the alibi-query is child's play in this model, and the alibi-query determines whether two trajectories have a possible intersection. It is merely necessary to determine whether there exists a time instant in which the two trajectories are less than twice the threshold apart. Evaluating this query in the bead model is much less trivial [33], since it involves solving four quadratic equations.

In case of network-constrained movements, like cars in a highway or trains in railroads, the uncertainty between two consequent sampled positions could be further reduced by exploiting the network topology. Such an idea is depicted in [3] where authors provide equations that describe the geometry of the uncertainty area.

5.3 Modelling Approaches for Trajectories

Approaches for modelling trajectories fall in three categories: the first two categories, *spatiotemporal data models* and *constraint data models*, do not propose specific concepts for trajectories but can be used to represent trajectories. The last category, *moving objects data models* regroups attempts specifically developed for the modelling and querying of moving objects and thus the modelling and querying of trajectories.

5.3.1 Off-the-Shelf Spatiotemporal Data Models

Many spatiotemporal models have been proposed in the literature, stemming from either the entity-relationship approach (e.g. ST USM [40, 62], STER [72, 73]), the object-oriented approach (e.g. Perceptory [7, 8, 12, 43], Extended spatiotemporal UML [60,61], OMT-G [11] STOQL [34], spatiotemporal ODMG [13], Tripod [28]), or a logic-based approach based on constraints (e.g. [29, 42, 63]). A framework for characterizing spatiotemporal data models is given in [21, 50].

A common characteristic of these models is the use of data types as basic building blocks for developing spatiotemporal data management. The definition of standard two-dimensional spatial data types has reached a good level of consensus in the GIS community. Although temporal data types have been standardized in the GIS community [36], no such agreement exists in the database community: Proposed solutions [35, 65, 66] have not reached the acceptance status by the SQL committees [67] [16], and an alternative approach has been proposed in [17]. As for spatiotemporal data types, the work by [30,31] is foundational for building a general approach that is applicable to any modelling dimension.

In this section, we have chosen among the rich literature those models that can be considered as spatiotemporal data model and that are able to model moving objects and trajectories. Although these conceptual models do not attempt to describe the internal structure of trajectories, they may be used to describe time-varying geometry [1] that is useful for modeling trajectories.

5.3.1.1 ISO TC 211

The ISO TC 211 Geographic information/Geomatics is the ISO technical committee responsible for defining international standards related to geographic information. These standards specify methods, tools and services for acquiring, processing, analyzing, accessing, presenting and transferring geographic information between different users, systems and locations. In this section, we use two of the ISO TC 211 standards:

1. ISO 19107 Geographic information – Spatial schema [37] defining a set of spatial data types and operations for geometric and topological spaces. It only covers vector data.
2. ISO 19108 Geographic information – Temporal schema [36] defining a set of temporal data types and functions needed to describe spatial events that occur in time.

5.3.1.2 STER

The spatiotemporal ER (STER) model [72,73] is an extension of the entity- relationship model with constructs for modelling spatiotemporal information. The structural

[1] More specifically a time-varying point, which store the successive positions, (x,y) pairs, of the object over time.

concepts provided by STER are those of the basic entity-relationship model: entity type, relationship type, attributes, and is-a (generalization/specialization) link.

5.3.1.3 Perceptory

The approach proposed by [7, 8, 12, 43] is to define spatial, temporal, (and multimedia) plug-ins for visual languages (PVL) that can be added to any existing database design tool. This visual plug-ins consists in a set of elementary concepts with their graphical symbols and an associated grammar defining how the symbols can be combined to express more complex concepts. Perceptory provides for two-dimensional and three-dimensional spatial types but in the following we restrict ourselves to two-dimensional types. A temporal PVL has also been defined. It provides two symbols representing the basic types Instant and Period. As for the spatial PVL, combinations of these symbols allow to represent alternative temporalities or multiple temporalities.

5.3.1.4 MADS

MADS [79] is an object + relationship spatiotemporal conceptual data model. In this model, it is assumed that the real world of interest that is to be represented in the database is composed of complex objects and relationships between them; both are characterized by properties (attributes and methods) and both are possibly involved in a generalization hierarchy (is-a links). Spatiality and temporality may be associated at the different structural levels: object, attribute and relationship. The spatiality of an object conveys information about its location and its extent; its temporality describes its life cycle.

5.3.1.5 Comparison of the Models for Trajectory Modelling

In terms of data representation, trajectories may be modelled as a time-varying geometry. Except ISO TC 211, all the aforementioned spatiotemporal data models allow to represent time-varying geometries. For instance, in Fig. 5.5 is shown two alternative ways to model car trajectories using the MADS data model. The design (Fig. 5.5a) defines a car as a spatial object type having a time-varying geometry i.e. its spatiality is a set of pairs (point, instant). This is shown by the point icon at the top right corner followed by a function symbol including the instant icon. Alternatively, the time-varying spatial attribute may be kept in the trajectory attribute representing the movement of car objects as in Fig. 5.5b. An equivalent schema can be defined using the STER and the Perceptory models.

In case the model does not have the concept of time-varying spatial attribute, an approach for representing trajectories is to represent trajectories as objects on their own, independently of the object that generates them. Figure 5.5c illustrates a design of car trajectory using ISO TC 211 standard spatial and temporal hierarchies. In this

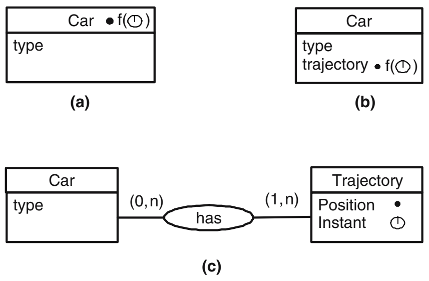

Fig. 5.5 Alternatives to trajectory modelling

design, an entity called trajectory is created with two attributes: position and time instant, which are ISO 19107 point (i.e. GM_Point class) and ISO 19108 instant (i.e. Instant class) type, respectively. For each entity that has a trajectory, we must associate it to the trajectory entity using a 1:N relationship.

To conclude we note that the representation of trajectories using the spatiotemporal data models presented above have the following limitations:

- The semantic of the above spatiotemporal concepts cannot express the exact semantic of what is a trajectory. For example, defining the car entity using a time-varying spatial attribute allows to represent the geometry of the trajectory adequately but does not encompass all the constraints and operations that are specific to trajectories. An example of such a constraint can state that a trajectory must contain at least two different instants, some specific operations can be the duration or the direction of the trajectory. A model for trajectory should provide for specific data structure for trajectories. The lack of specific structures for describing trajectories demands additional work to the designer in correcting the impedance between spatiotemporal constructor's semantics and correct trajectory representation.
- Although trajectories may be expressed as a spatiotemporal entity attribute, in some spatiotemporal data model, such as in STER, spatial attributes are inherited (or, obtained) from space, meaning that the entity attribute is defined whether or not the entity exists at specific position in space.
- None of the models propose any specific operations to analyse trajectories.
- None of the models propose any specific query language operators to query trajectories.

5.3.2 *The Constraint Database Approach to Trajectories*

During the past ten years, an acclaimed method for effectively representing infinite geometrical figures in databases is provided by the *constraint database model.* This model was introduced by Kanellakis, Kuper and Revesz in 1990 [39] and deeply studied during the second half of the 1990s (an overview of the area of constraint databases can be found in [49]). In the constraint model, a two-dimensional geometrical figure, for instance, is finitely represented by means of a Boolean combination of polynomial equalities and inequalities. These involve polynomials with two real variables that represent the spatial coordinates of a point in the plane. The set of points on the upper half of the unit circle, for instance, is in this context given by

$$\{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 = 1 \wedge y \geq 0\}$$

(in mathematical terminology, these figures are called *semi-algebraic sets* and for an overview of their properties we refer to [9]).

This way of representing fixed figures can easily be adapted to describe figures that change. Indeed, we can add a time dimension and consider geometrical objects in three-dimensional space–time that are described by polynomial equalities and inequalities that also have a time variable t. The set

$$\{(t,x,y) \in \mathbb{R}^{1+2} \mid x^2 + y^2 = (1-t)^2 \wedge y \geq 0 \wedge 0 \leq t \leq 1\}$$

models a shrinking half circle, whereas the set

$$\{(t,x,y) \in \mathbb{R}^{1+2} \mid (x-t)^2 + y^2 = 1 \wedge y \geq 0 \wedge 0 \leq t \leq 1\}$$

models a half circle that moves along the x-axis. In [41], an SQL-like query language was discussed for this kind of data that focusses on exploiting topological changes in moving or changing geometric figures.

Since, trajectories can be seen as a collection of infinite points connecting a finite number of sample points, the constraint model can be specialized to model moving points and trajectory data.

In this section, we discuss some specific attempts that can be classified under the constraint model, to deal with trajectories. In particular, we look at the *linear constraint model*, an approach based on *differential geometry*, and an approach using *equations of motion*.

5.3.2.1 The Linear Constraint Model

When the polynomials used to model data are restricted to be linear, we have the *linear constraint model*. Given a finite set of sample points, in space–time, the linear constraint model basically assumes that the moving object moves with constant speed along a straight line connecting two succeeding sample points. This speed is the minimal average speed needed to reach the destination. The graph of such a trajectory is a piecewise linear curve.

Suppose we have m time instants $t_1 < t_2 < \ldots < t_m$, and a function p that maps time t to a point

$$p(t) = (p_1(t), p_2(t), \ldots, p_n(t))$$

in $\mathbb{R}^n$, describing the trajectory of a moving object. In the linear constraint model each p_i is represented by the constraint $p_i(t) = b_{ij}t + c_{ij}$ for $t_j \leq t \leq t_{j+1}$ $(j = 1,\ldots,m-1)$.

Obvious drawbacks are the discontinuities of the speed (and moving direction) of the moving object in the sample points. This makes the trajectory unsmooth and thus seemingly unnatural. On the other hand, a big argument for this approach is the ease of computation it allows.

5.3.2.2 The Differential Geometry Model

One approach by Su, Xu and Ibarra [69] is through the use of differential geometry. They use first- and second-order derivatives to describe direction, velocity and acceleration. On the other hand, they allow vector arithmetic on moving points to compute distances and speeds. A moving object is defined as a piecewise C^∞- curve in $\mathbb{R}^n$. Let $t_1 < t_2 < \ldots < t_m$ be m time instants and p a function that maps time t to a point $p(t) = (p_1(t), p_2(t), \ldots, p_n(t))$ in $\mathbb{R}^n$ where each p_i is a real continuous (for all t) function that is infinitely differentiable on $]-\infty, t_1[$, each $]t_{j-1}, t_j[$ and $]t_m, +\infty[$. The nature of the functions $p_i(t)$ suitable for this depend on the specific application. It is clear that this model generalizes the linear constraint model.

In this model, a query language is provided based on the following operations. First, typical operations in a metric vectors space are allowed (sum of two vectors and scalar multiplication with a real number; and the standard dot product and thereof derived norm/length of a vector). Second, differential geometry operations (taking the derivative) are allowed:

- The velocity of a moving object $\mathrm{vel}(p) = p'(t) = (p'_1(t), p'_2(t), \ldots, p'_n(t))$
- The acceleration of a moving object $\mathrm{vel}'(p) = p''(t) = (p''_1(t), p''_2(t), \ldots, p''_n(t))$
- The speed of a moving object $\|\mathrm{vel}(p)\|$
- The moving direction of a moving object $\frac{\mathrm{vel}(p)}{\|\mathrm{vel}(p)\|}$
- The distance between two moving objects p and $q : \|p - q\|$

A database system is constructed now as follows. First, a logical vector type is introduced. If τ is a function of type $time \to \mathbb{R}$, then the LV (logical vector) type is of the type $\tau^n = \tau \times \tau \times \cdots \times \tau$:

- A relation schema R is a finite set of pairs (A, T), where A is an attribute name and T of the LV-type.
- A database schema D is a finite set of relation schemas.
- A tuple is a total mapping from attribute names to the domains of the LV-types and a relation instance is a finite set of tuples over R.
- 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.

A first-order formula $\varphi(x)$ with free variables defines a query Q as follows: if I is a constraint database, then the answer to Q on I is

$$Q(I) = \{a \mid I \models \varphi(a)\}.$$

- Variables and values of associated types or LV-types are terms
- Field operations on variables of type real or time are again terms

– Vector space operations on LV-types over the real type field are again terms
– If x is a vector and p is a term of the LV-type, then $\mathrm{unit}(x)$, $\mathrm{vel}(p)$, $\mathrm{dir}(p)$ are terms of the LV-type and $\mathrm{len}(p)$ is a term of the real type.

Let a be a time instant, p a value in the LV-type domain, and $x_1,\ldots,x_n$ real values, then $p(a;x_1,\ldots,x_n)$ is *true* if p is at $(x_1,\ldots,x_n)$ at the time instant a.

5.3.2.3 The Equations of Motion Model

Another way of approaching moving objects is storing objects through their *equation of motion* [27]. The idea behind this is to use Newton's Second Law, which states that once you know the total force acting on an object, you know its motion. Newton's Second Law connects force with acceleration through the object's mass. Let $x : \mathbb{R} \to \mathbb{R}^n : t \mapsto x(t) = (x_1(t), x_2(t), \ldots, x_n(t))$ describe the coordinate to a moving object at *time* t, let m be its mass, and let $F : \mathbb{R} \to \mathbb{R}^n : t \mapsto F(t) = (F_1(t), F_2(t), \ldots, F_n(t))$ describe the total force $F(t)$ acting on the object. The Second Law then states that

$$F(t) = m\frac{\mathrm{d}^2 x(t)}{\mathrm{d}t^2}.$$

Given two initial conditions $x_0 = x(t_0)$ and $v_0 = \mathrm{d}x(t_0)/\mathrm{d}t$, initial position and initial speed of the object, a single solution to this second-order differential equation can be found, not always analytically.

One can reduce the order of this equation by adding variables

$$\frac{\mathrm{d}}{\mathrm{d}t}X(t) = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ \frac{F(t)}{m} \end{pmatrix}.$$

The space in which the image of $X(t)$ lies is also referred to as the phase space.

A differential constraint over the variables is of the form

$$\dot{x_i} = f_i(x_1,\ldots,x_n,t),$$

where f_i is a multivariate polynomial in its variables. The author then describes an equation of motion as a finite set of triples where every triple contains a set of initial constraints, a set of differential constraints and an end time for which the triple is valid.

A trajectory is constructed from such a triple but first-order approximation, i.e. approximation with a linear piecewise curve. In the article, the author demonstrates this using Euler's method. The moving object database is then a triple consisting of a finite set of object identifiers, a mapping from this set to the set of all equations of motion and finally a time instant, which is an upper bound for existence of all the moving objects.

5.3.3 Modelling and Querying Moving Objects Databases

About a decade efforts attempt to achieve an appropriate kind of interaction between temporal and spatial database research. Spatiotemporal databases are the outcome of the aggregation of time and space into a single framework [1,53,56,68]. Substantial research work has been carried out focussing in modelling spatiotemporal databases, while recently new needs have been imposed by a series of ubiquitous applications as location-based services. This section presents a parallel line of research focussing on modelling trajectories as spatiotemporal objects (the so-called *moving objects*). Researchers in the field of Moving Objects Databases (MOD) have been studying the representation issues of trajectories into computer systems aiming at keeping track of object locations, as well as supporting location-aware queries. If, only time-dependent locations need to be managed (e.g. mobile phone users, cars, ships, etc.), then *moving point* is the basic abstraction; while, if the time-dependent shape or extent is also of interest (e.g. group of people, armies, spread of vegetation), then we are talking about *moving regions*.

A straightforward approach widely used in industry is to model a moving point by generating periodically a location-time point of the form (l,t), indicating that the object is at location l at time t, where l may be a coordinate pair (x,y). Points are stored in a database, and a database query language is used to retrieve the location information. This method is called *point-location management*, and it has several critical drawbacks, such as (1) does not enable interpolation or extrapolation, (2) leads to a critical precision/resource trade-off and (3) leads to cumbersome and inefficient software development.

In the literature of the MOD field, there are two main approaches to model trajectory data: one for querying current and future positions of the moving objects in [64, 76, 77] and the second for querying past trajectories of the moving objects in [22,25,30,44].

Querying current and future positions must consider the problem of managing the positions of a set of entities in a database. However, to keep the location information up-to-date, we encounter an unpleasant trade-off. If updates are sent and applied to the database very often, the error in location information in the database is kept small, yet the update load becomes very high. Indeed, keeping track of a large set of entities is not feasible. Conversely, if updates are sent less frequently, the errors in the recorded positions relative to the actual positions become large. This problem was explored by Wolfson et al. [64,76,77], who have developed a model, called MOST, that allows one to store the motion vector rather than the objects' positions in the database, avoiding a very high volume of updates. In Wolfson and colleagues' work, the location of a moving object is simply given as a linear function of time, which is specified by two parameters: the position and velocity vector of the object. Thus, without frequent update messages, the location server can compute the location of a moving object at a given time t through linear interpolation: $y(t) = y_0 + v(t - t_0)$ with time $t > t_0$. An update message is only issued when the parameter of the linear function, i.e. v, is changed. This update approach is called *dead-reckoning*. It offers a great performance benefit in linear mobility patterns, but performance suffers when the randomness of the mobility pattern increases. In addition, Wolfson et al. group incorporates a new concept of dynamic attributes, which change over time; hence the results of queries also change over time, leading to a notion of continuous queries. The related query language, called future temporal logic (FTL), allows one to specify temporal relationships between objects in queries. This approach is restricted to moving point objects, and is dealing with current and expected near future movement. FTL has the following SQL/OQL type syntax:

RETRIEVE target-list
WHERE condition-list

The condition part is specified as a FTL formula. FTL formulas are interpreted over future histories specifying the object locations. Static attributes remain unchanged, while dynamic attributes change according to their functions. FTL employs a variety of spatial, temporal predicates and operators. Later we present three representative examples illustrating the FTL query language:

Q1: Retrieve names of red colour objects that will be inside the region P within 10 units of time.

RETRIEVE O.name
WHERE O.colour = red AND Eventually-within-10 (INSIDE(O,P))

Q2: Retrieve names of objects that will be within a distance of 10 from a truck for the next five units of time.

RETRIEVE O.name
WHERE Always-for-5 (DISTANCE(O,O')= 10) AND O'.type = truck

Q3: Retrieve all objects that enter a tunnel in the next 5 units of time and stay inside it for the subsequent 10 time units.

RETRIEVE O.type
WHERE Not Inside(O,P) AND Eventually-within-5(Always-for-10 (Inside(O,P)) AND P.type = tunnel

The need for capturing complete histories of objects' movement has promoted the investigation of continuously moving objects. Clearly as location data may change over time, the database must describe not only the current state of the spatial data but also the whole history of this development. Thus it should allow to go back in time at any particular instant, to retrieve the state of the database at that time, to understand the evolution, to analyze when certain relationships were fulfilled and so forth. This approach was developed by Guting and colleagues [22, 25, 30, 44]. They described a new approach where moving points and moving regions are viewed as three-dimensional (2D space + time) or higher-dimensional entities whose structure and behaviour is captured by modelling them as abstract data types. Such types and their operations for spatial values changing over time can be integrated as base (attribute) data types into object-relational, object-oriented or other extensible database management system. More specifically, they introduced a type constructor τ, which transforms any given atomic data type a into a type $\tau(a)$ with semantics $\tau(a) = time \rightarrow a$. In this way, the two basic types defined, namely *mpoint* and *mregion*, may also be represented as $\tau(point)$ and $\tau(region)$, respectively. They provided an algebra with data types such as moving point, moving line and moving region together with a comprehensive set of operations. All the types that are produced by application of the type constructor τ on other data types are functions over an infinite domain; hence each value is an infinite set.

It is important to note that in MOD modelling, the trajectory of a moving point can be described either as a curve or as a polygonal line in three-dimensional space. In the first case, a curve is defined as a certain kind of infinite set of points without fixing any finite representation. In the second case, the definition uses a finite representation of a polyline, which in turn defines the infinite points set making up the trajectory of the moving point. In Erwig et al. [22], the difference between these two levels of modelling is discussed at some depth, and the terms abstract and discrete modelling are introduced for them. As an extension to the abstract model in [24,30] introduced the concept of *spatiotemporal predicates*. The goal was to investigate temporal changes of topological relationships induced by temporal changes of spatial objects. A corresponding query language incorporating these concepts was presented in [23]. In [25], the authors presented the definition of the discrete representation of the above-discussed abstract data types. They introduced the concept of sliced representation, the basic idea of which is to decompose the temporal development of a moving value into fragments called *slices* such that within a slice this development can be described by some kind of simple function. Algorithms implementing the rather large set of operations defined in [30] are studied in [25,44].

The final outcome of this work was a system that implements the above-described moving objects data model and query language completely integrated into a DBMS environment [4]. More specifically, the prototype has been developed as an algebra module in SECONDO's extensible environment [18]. Further we provide three representative queries exemplifying the resulted SQL-like query language [4]:

Q1: Where exactly were the trains during period P?

SELECT Id, Line, trajectory(Trip atperiods P) AS Stretch
FROM Trains
WHERE Trip present P;

Q2: At what times have trains passed through (underground) the region R?

SELECT Id, Line, deftime(Trip at R) AS Times
FROM Trains
WHERE Trip passes R;

Q3: Where have the trains passing through station S been at time T (as far as they are moving at this time).

SELECT Id, Line, val(Trip atinstant T) AS Pos
FROM Trains, Stations
WHERE Trip passes Loc AND SName contains S AND Trip present T

Another approach following the paradigm of moving objects was presented in [52]. This research focussed on the representation and querying of continuously as well as discretely moving objects similar to those presented in [30]. From a theoretical point of view, a data type-oriented model (STAU) that supports the representation of objects both under object-oriented and object-relational platforms was introduced. From a technical point of view, two data cartridges under ORACLE object-relational DBMS were developed. The first cartridge provides pure temporal functionality implementing TAU temporal types [38]. The second cartridge supports a palette of moving object data types, which has been implemented by merging the temporal cartridge with Oracle's spatial cartridge. The resulted system supports a wide set of object methods that extends the Oracle PL/SQL query language with spatiotemporal semantics. Indicative examples of the aforementioned query language include:

Q1: When did John leave the rectangular area defined by (x_1, y_1) *lower left and* (x_2, y_2) *upper right co-ordinates?*

SELECT h.route.f_leave(SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1003,3), SDO_ORDINATE_ARRAY(x_1, y_1, x_2, y_2)))
FROM humans h
WHERE h.id = John

Q2: What is John's speed at 24/11/2007-10:45:30?

SELECT h.route.f_speed(tau_tll.d_timepoint_sec(2007,11,24,10,45,30))
FROM humans h
WHERE h.id = John

Q3: Find John's friends that are located within 1,000 m distance from his current location.

SELECT f.id
FROM humans h, friends f
WHERE h.id = John AND h.route.f_within_distance(1000, f.route, tolerance, NOW))

By assuming that a trajectory is modelled in its finest spatial granularity (exact location), all the previously mentioned data models provide support at two levels.

First, they provide a mechanism to split a trajectory into sub-paths, according to some variables such as the sampling rate and the most appropriate update time. Second, MOD models usually imply a linear interpolation between the exact locations. Considering the first issue, sliced representation [25] is adopted as the solution for the model proposed in [30]. In [54], the authors utilize sliced representation and develop a moving type that associates a period of time with the description of a simple function that models the behaviour of the moving object in that specific time period. Considering the second issue, linear interpolation is considered sufficient for the querying purposes of a MOD. However, other types of interpolations could be as well important either for making motions more realistic or for sub-serving the tasks of privacy and/or modelling in various granularities. The model in [54] provides an extensible mechanism to support different kinds of interpolation, currently implementing linear and arc sub-motions. This model was recently extended [10, 55] to support not only historical queries but also dynamic ones.

Following the modelling primitives described earlier, several solutions have been proposed to address specific query processing issues in MODs. Research in the field is driven by related work performed in the domain of (stationery) spatial databases. For instance, queries of the form '*find all objects located within a given area during a certain time interval*' generalize the spatial range query of the form '*find all objects within a given area*'. Many different types of the so-called *coordinate-based queries* [59] have been proved to be useful for MODs: Queries of the form '*find all objects' locations within a given area at a certain time instance*', called *timeslice* queries, constitute a special type of range queries where the temporal extent is set to zero. Another straightforward extension of pure spatial queries includes the *nearest neighbour* queries of the form '*find the nearest moving object to a query object at a certain time instance (or during a certain time interval)*'. As discussed in [26], in the case of time-interval nearest neighbour queries, the query object can be either a two-dimensional point or another trajectory, while the query may return either the nearest to the query object in any time during a time interval or in every time instance of the query time interval (historical continuous queries). The last extension of spatial queries already discussed in the spatiotemporal trajectory literature deals with trajectory join queries [5, 6], which are categorized in the so-called *distance join* and *k-closest pairs* queries. The former is defined as follows: Given two sets of trajectory data sets $Q = q_1,..,q_n$ and $T = t_1,..,t_m$ compute all pairs (q_i,t_j), where q_i and t_j have distance no more than a particular threshold at a given time stamp. In correspondence to the classic closest-pair problem of computational geometry, the latter finds the k closest pairs of trajectories between the two data sets at the given time stamp, i.e., the pairs of (q_i,t_j) that have the k smallest in-between distances at the given time stamp. Both queries can be generalized in the *time-relaxed* context where the temporal dimension is of no interest; as such the latter query type is transformed to '*find the k closest pairs of trajectories between the two data sets at any time stamp*'. Another useful co-ordinate-based query (Fig. 5.6) in MOD derives from the so-called *trajectory similarity problem* and aims to find 'similar' trajectories of moving objects. To handle such queries efficiently, MOD systems

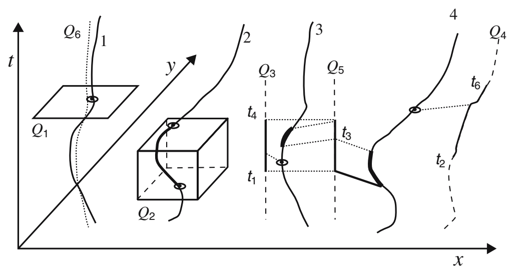

Fig. 5.6 Coordinate-based queries: timeslice (Q1), range (Q2), point nearest neighbor (Q3) trajectory nearest neighbor (Q4), historical continuous point nearest neighbor (Q5), most similar trajectory (Q6)

should incorporate query processing methods for supporting *most similar trajectory (MST)* search also discussed in [15, 70, 74].

According to the classification in [57, 59], apart from co-ordinate-based queries, the so-called *trajectory-based queries* are also of great interest. In contrast to co-ordinate-based, trajectory-based queries require the knowledge of the complete – or at least a subset of the – object's trajectory to be processed. Such queries consider topological relations (e.g. enter, leave, etc.) and may provide derived information about an object's navigation (e.g. average speed, travelled distance, etc.). Furthermore, the combination of a range with topological queries produces another class of queries called *combined queries*. As an example [59], consider the following query *'What were the trajectories of objects after they left Tucson street between 7 and 8 a.m. today, in the next hour'*, which first locates the trajectories contained in an inner range query window (*Tucson street, between 7 and 8 a.m. today*) and then retrieve those parts of objects' trajectories contained in an outer query window (*in the next hour*).

5.4 Open Issues

As presented in Sect. 5.3.3, consequent quality work has been done regarding trajectory modelling and querying. However, open issues remain particularly regarding conceptual modelling of trajectories, uncertainty, multiple representation of trajectories, continuously acquired trajectories and query capabilities. They are described further.

5.4.1 Conceptual Modelling of Trajectories

Most of the works on moving objects have paid very little attention on the conceptual description of the trajectory. In those models, the trajectory of a given object is a by-product, so to speak, of capturing the object's mobility. Indeed describing an object type as a moving point allows representing the position of the object all along its lifespan. It does not allow the system to be aware of the semantic segmentation of the object's paths into semantically meaningful trajectories. To be able to associate a trajectory or a list of trajectories to the object we need more than moving points. Trajectories should be promoted as a modelling construct i.e. be first-class data, rather than computable derived data. Moreover, the specification of the trajectory construct should be done at the conceptual level to fix a purely conceptual view of the concept and to ensure its maximal flexibility. Indeed, when looking for a conceptual model for trajectories, we have to focus on trajectory characteristics that are generic, i.e. independent of any specific application domain, while being relevant to the application realm and not driven by considerations pertaining to their implementation in a computer-based system. To propose a conceptual solution for the trajectory concept, the following issues need to be tackled.

5.4.1.1 Conceptual Description of Trajectories

From a conceptual point of view, the concept of trajectory denotes the evolving position of some object in some space, from an initial position to a final position. A trajectory has two facets:

– A spatiotemporal facet: it allows to record the positions of the moving object.
– A semantic facet: it allows to associate application-dependant information or characteristics to the whole trajectory as well as to any of its subparts. For example, a point-based trajectory for a person could store the activity the person is doing between two defining points not necessarily consecutive (visiting, walking to work, etc.) and the transport means. Obviously, the trajectory of a car will bear completely different data: a car trajectory could store for each defining point its road distance from the previous point, the duration of *en route* stops, the amount of highway fees paid and the gasoline consumption over the last segment.

5.4.1.2 Constraints of Trajectories with Their Environment

In the same way as we need to associate semantic data to trajectories and/or to their subparts, we want to be able to describe the constraints holding between the trajectories or their subparts and their environment [75]. As trajectories are spatiotemporal object types, semantic as well as both topological and temporal constraints should be considered. For example, a topological constraint might describe the fact that the whole trajectory of this person is included in the area delimitated by the old city part of Lausanne but also that this particular part of the trajectory is equal to the geometry of the stairs going to the cathedral.

5.4.1.3 Spatial, Temporal or Spatiotemporal Operators

To manipulate trajectories, a set of operators should be identified. As trajectories are spatiotemporal objects, the data model must offer traditional temporal and spatial operators like the ones proposed by Allen [2] in the temporal dimension and

Egenhofer [20] in the spatial dimension, respectively, as well as trajectory-specific operators (such as, for instance, how to 'sum-up' two trajectories). Indeed, promoting trajectories as first class modelling constructs has direct consequences on the operators applying to trajectories. It implies evolving from operations on moving objects as defined by e.g. [30] to manipulation of trajectories as a whole. Thus investigations should be done to define an underlying algebra for trajectories.

5.4.2 Uncertainty

The representation of moving objects within a MOD is inherently imprecise due to errors introduced from different sources like measurement, sampling and preprocessing.

So far, the related research [3, 58, 71] emphasizes on sampling errors introduced by the interpolation process, which is utilized to 'predict' the moving objects positions within the sampled points. The usually adopted interpolation method is linear interpolation, which is the simplest one. More advanced techniques, like polynomial splines, should be also considered so as to better approximate the actual movements.

Interpolation uncertainty, although important, is only one source of uncertainty. Further uncertainty sources should be considered and their impact on the represented trajectory with respect to the actual trajectory should be investigated. Examples of such sources are compression, simplification and matching of trajectories to the network (in case of a network-constrained movement).

The uncertainty so far refers to the spatial dimension, i.e. how well the stored positions of the moving object represent its actual positions. Spatial information, however, is not the only aspect of a movement. Time and speed, for example, constitute other aspects that might contain uncertainty in their values.

The idea of restricting uncertainty by exploiting the underlying network has started to be addressed in the literature [3]. However, a further uncertainty factor is imposed here that of finding the position of the moving object within the network, i.e. network matching. Investigating how matching affects the quality of trajectories, and how the uncertainty it imposes interacts with the inherent uncertainty of trajectories seems a promising research line. Furthermore, the uncertainty area of an object within a network depends on the geometry of the network, which, in the general case, does not solely consists of straight lines. More complex geometries, e.g. circles or spirals, should be investigated and the shape of uncertainty areas should be allocated.

To conclude, uncertainty in the representation of a moving object is an important issue within a moving object database, since the adopted representation is the basis for other DB operations like querying and indexing. Two critical questions arise: what is an effective representation schema for trajectories under uncertainty and how the uncertainty is propagated into other MOD operations. Since uncertainty is a reality in MOD, the efforts should be towards its limitation so as to provide the end user with reliable results.

5.4.3 Streaming Models

The streaming model is completely suitable for moving object data since they encounter frequent updates, their volume is unexpectedly varied, and they are being processed under real-time conditions with several continuous spatiotemporal queries. Thus, MODs perfectly fit with the stream concept, and it sounds reasonable to go towards a streaming procedure that feeds a trajectory database or a trajectory data warehouse. Streaming spatiotemporal data has addressed a considerable research attention in the past few years [32, 45, 46, 51]. Existing work in streaming models include [51], which attempts to model the management of moving objects under the assumption that the trajectories are continuous, time-varying and possibly unbounded data streams, proposing a basic framework for managing trajectory streams with the introduction of constructs for advanced query capabilities in an SQL-like language. However, modelling of moving objects must be further studied, in addition to the introduction of algebraic constructs for windows and to the proposal of syntax rules for query language.

5.4.4 Multiple Representation

Multiple representation is an important issue regarding the modelling of trajectories. Multiple representation means that we want to store or to be able to retrieve several representations for the same phenomenon in the database. This is an important requirement as each application has its own perception of the real world and its own data processing tasks leading to specific requirements both in terms of what information is to be kept and in terms of how information is to be represented. Spatial and temporal applications show additional requirements in terms of multiple representation as they need flexibility also in the perception and representation of spatial and temporal features.

In the context of trajectories, multiple representation may result from the description of the same trajectory according to several viewpoints but more importantly according to different spatial and temporal granularity. Viewpoint, here, should be understood as the expression, by a group of users, of their specific interests in data management. Granularity refers to the notion that the world is perceived at different level of details i.e. in the temporal dimension using more or less time steps like in [38] and in the spatial dimension considering a smaller or bigger spatial resolution. Complex applications naturally include the need for multiple representations of trajectories. Indeed different tasks often require different granularities: for instance, considering the trajectory of a person travelling from home in Lausanne to work in Geneva, some tasks might be only interested to analyse the trajectory from the starting point in Lausanne to the arrival point in Geneva and then use a coarse spatial and temporal granularity. On the contrary, another task might need a more detailed description: at 7.40 a.m. the person has left home to walk to the bus station for 10 min, then has taken the bus to the train station where the person

has been waiting for 5 min, then travelled for 30 min, etc. In this example, the same trajectory is modelled at different levels of spatial and temporal granularity. Moreover, the same trajectory may show parts in different granularity, e.g. more detailed data for critical sections of the trajectory: for instance, a detailed description of the trajectory between the person's home and the train station will be kept while from Lausanne train station to Geneva train station no specific detail is necessary.
Although multiple representation has received a lot of attention in the spatiotemporal database community, multiple representation applied to the description of moving objects and their trajectories is still an open issue on which few propositions exists [14, 33]. In this area, research work has to be done to provide for a model describing multiple representations of the same trajectory including conversion operators to shift among granularities and an adapted query language. Proposed operators for granularity change exist, but how to maintain multiple representations with maximal flexibility has rarely been addressed.

5.4.5 Query Capabilities

Open issues regarding the query types supported by MODs include novel query types, meaningful only in the spatiotemporal domain, as well as the expansion of query types from other domains. In particular, there exist a significant number of spatial queries not yet adopted in the spatiotemporal framework, and particularly in trajectory databases. For example, the recently proposed group nearest neighbour query [48] can be also applied in the spatiotemporal domain; nevertheless, its employment is not straightforward at all, since a trajectory GNN query would had to clarify several issues concerning its definition (a) the type of the query objects (static or continuously moving), (b) the time interval during which the GNN is requested, (c) the way of determining the distance of GNN from the query objects, since distances from all query objects could be calculated on exactly the same time stamp or considering the entire query period. In addition, such type of queries involving the calculation of the nearest distance of trajectories could be extended by employing the notion of network distance discussed in the domain of spatial network databases (SNDB) [47].

Regarding the trajectory similarity topic, the majority of the existing approaches, being mainly inspired by the time series analysis domain, propose generic similarity metrics for two-dimensional data [15, 74] in order to answer queries requesting about the most similar to a given trajectory. However, the notion of 'textitmost similar' in the trajectory domain can be considered through several aspects, since trajectories have spatial, temporal, spatiotemporal and other derived features. Consider for example the following queries:

- Query 1 (*spatial similarity*): Find objects whose route (i.e. the projection of trajectory on two-dimensional plane) is quite similar to that of object id= 132 (irrespective of time).

- Query 2 (*spatiotemporal similarity*): Find objects that follow a route similar to that of object id= 132 during the same time interval, e.g. 3–6 PM.
- Query 2a (*speed pattern-based spatiotemporal similarity*): Find objects that follow a route similar to that of object id= 132 and, additionally, move with a similar speed pattern.
- Query 3 (directional similarity): Find objects that follow a direction similar to a given direction pattern, e.g. NE during the first half of the route and subsequently W.

To the best of our knowledge, there is no work dealing with the different similarity types that can be defined based on these underlying parameters of the trajectories. However, such queries are expected to be at least as popular as those already examined, which mainly deal with the spatial similarity between trajectories.

5.4.6 Conclusion

The objective of this chapter was to provide an extensive discussion of the state-of-the-art on data modelling of trajectories. We have initiated our discussion by describing trajectory as the record of time-varying phenomenon. In terms of data representation, trajectories are described as a collection of sample points that need to be linked. To find the suitable curve connecting the sample points, interpolation methods are used. As shown, there are two kinds of methods that may apply to this problem, and the trajectory accuracy depends basically on the number of sample points and the method used. Interpolation only gives a guess of a probable trajectory, leaving a certain amount of uncertainty that needs to be taken care of. Two approaches have been presented to treat uncertainty of trajectories.

In terms of data modelling, many research efforts have been done in modelling spatiotemporal applications, among them some are specifically defined for moving objects and their trajectories, some are not. Indeed, since trajectory applications are a sub-domain of spatiotemporal applications, we have analysed these off-the-shelf spatiotemporal models to deal with trajectory representation. Similarly, we have analysed constraint database modelling that provides a method to represent infinite geometrical entities in databases and thus could be specialized to model moving points and trajectory data. Finally, we have presented important researches in the field of MOD that are studying representation issues of trajectories as well as their querying. From a theoretical point of view, constraint and moving objects models approaches are less conceptual and more implementation-oriented than spatiotemporal data models are. Former approaches focus more on definition of mathematical models, abstract data types, algorithms for set of operations and query answering approaches.

For completeness of the discussion, we have shown in Sect. 5.4 open issues in trajectory modelling that we found relevant. The first open issue concerns the definition of an adequate conceptual representation of trajectories not as a by-product of capturing objects' mobility but as a first class-construct. Effective representation of trajectories taking into account uncertainty and its propagation to operations are also important issues that need to be addressed. Besides, the high-data volume and frequent updates in the context of streaming spatiotemporal data model has not received so far the attention it requires. Another important open issue regarding the modelling of trajectories is to deal with several representations for the same trajectory in the database (i.e. multiple representation). Last but not least, queries capabilities must be improved in the context of trajectory databases, for instance queries based on: derived trajectory information (e.g. speed), nearest neighbour or trajectory similarity.

References

1. T. Abraham and J.F. Roddick. Survey of spatio-temporal databases. *Geoinformatica*, 3(1):61–99, 1999.
2. J.F. Allen. Maintaining knowledge about temporal intervals. *Communications ACM*, 26(11):832–843, 1983.
3. V.T. de Almeida and R.H. Güting. Supporting uncertainty in moving objects in network databases. In *Proceedings of the 13th International Symposium on Geographic Information Systems (GIS'05)*, pp. 31–40, ACM, 2005.
4. V.T. Almeida, R.H. Güting, and T. Behr. Querying moving objects in secondo. In *Proceedings of Mobile Data Management (MDM'06)*, p. 47, 2006.
5. P. Bakalov, M. Hadjieleftheriou, E.J. Keogh, and V.J. Tsotras. Efficient trajectory joins using symbolic representations. In *Proceedings of Mobile Data Management (MDM'05)*, pp. 86–93, 2005.
6. P. Bakalov, M. Hadjieleftheriou, and V.J. Tsotras. Time Relaxed Spatiotemporal Trajectory Joins. In *Proceedings of the 13th Annual International Workshop on Geographic Information Systems (GIS'05)*, pp. 182–191, 2005.
7. Y. Bédard. Visual modeling of spatial databases: Towards spatial pvl and uml. *Geomatica*, 53:169–185, 1999.
8. Y. Bédard, S. Larrivée, M.-J. Proulx, and M. Nadeau. Modeling Geospatial Databases with Plug-Ins for Visual Languages: A Pragmatic Approach and the Impacts of 16 years of Research and Experimentations on Perceptory. In *Conceptual Modeling for Advanced Application Domains*, Vol. 3289, pp. 17–30. Springer, Berlin Heidelberg New York, 2004.
9. J. Bochnak, M. Coste, and M. Roy. *Géométrie Algébrique Réelle*. Springer, Berlin Heidelberg New York, 1987.
10. Boosting location-based services with a moving object database engine. In *Proceedings of the 5th Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'06)*.
11. K. Borges, C. Davis, and A. Laender. Omt-g: An object-oriented data model for geographic applications. *GeoInformatica*, 5:221–260, 2001.
12. J. Brodeur, Y. Bédard, and M.-J. Proulx. Modelling Geospatial Application Database Using Uml-Based Repositories Aligned with International Standards in Geomatics. In ACM, editor, *Proceedings of the 8th International Symposium on Geographic Information Systems (GIS'00)*, pp. 39–46, 2000.
13. E. Camossi, M. Bertolotto, E. Bertino, and G. Guerrini. A Multigranular Spatiotemporal Data Model. In *Proceedings of the 11th International Symposium on Geographic Information Systems (GIS'03)*, pp. 94–101, ACM, 2003.
14. E. Camossi, M. Bertolotto, and E. Bertino. A flexible approach to spatio-temporal multigranularity in an object data model. *International Journal of Geographical Information Science*, 20(5), 2006.

15. L. Chen, T. Özsu, and V. Oria. Robust and Fast Similarity Search for Moving Object Trajectories. In F. Ozcan (ed.), *Proceedings of the International Conference on Management of Data (SIGMOD'05)*, pp. 491–502. ACM, 2005.
16. H. Darwen. Valid Time and Transaction Time Proposals: Language Design Aspects. In *Temporal Databases: Research and Practice, LNCS 1399*, pp. 195–210, 1998.
17. C. Date, H. Darwen, and N. Lorentzos. *Temporal Data and the Relational Model*. Model, Morgan Kaufmann, 2003.
18. S. Dieker and R.H. Güting. Plug and play with query algebras: secondo - a generic dbms development environment. In *Proceedings of the International Symposium on Database Engineering & Applications (IDEAS '00)*, pp. 380–392. IEEE Computer Society, 2000.
19. M.J. Egenhofer. Approximations of geospatial lifelines. 2003.
20. M.J. Egenhofer and R.D. Franzosa. Point set topological relations. *International Journal of Geographical Information Systems*, 5:161–174, 1991.
21. B. El-Geresy and C. Jones. *Five Questions to Answer in Time: A Critical Survey of Approaches to Modelling in Spatio-Temporal GIS*, Chap. 3. GIS and Geocomputation-Innovations in GIS 7. Taylor & Francis, London, 2000
22. 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.
23. M. Erwig and M. Schneider. Developments in Spatio-Temporal Query Languages. In *Proceedings of 10th International Conference and Workshop on Database and Expert Systems Applications (DEXA'99)*, pp. 441–449, 1999.
24. M. Erwig and M. Schneider. Spatio-temporal predicates. *IEEE Transaction Knowledge Data Engeneering*, 14(4):881–901, 2002.
25. L. Forlizzi, R.H. Güting, E. Nardelli, and M. Schneider. A Data Model and Data Structures for Moving Objects Databases. In *Proceedings of the International Conference on Management of Data (SIGMOD'00)*, pp. 319–330, 2000.
26. E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis. Nearest Neighbor Search on Moving Object Trajectories. In *Proceedings of 9th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*, Vol. 3633. *Lecture Notes in Computer Science*, pp. 328–345. Springer, Berlin Heidelberg New York, 2005.
27. F. Geerts. Moving Objects and their Equations of Motion. In *Proceedings of the 1st International Symposium on Applications of Constraint Databases*, volume 3074 of *Lecture Notes in Computer Science*, pp. 41–52. Springer, Berlin Heidelberg New York, 2004.
28. T. Griffiths, A. Fernandes, N. Paton, and R. Barr. The tripod spatio-historical data model. *Data and Knowledge Engineering*, 49:23–65, 2004.
29. S. Grumbach, M. Koubarakis, P. Rigaux, M. Scholl, and S. Skiadopoulos. *Spatio-temporal Models and Languages: An Approach Based on Constraints*, Chap. 5, pp. 177–201, 2003.
30. R.H. Güting, M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and quering moving objects. *ACM Transactions on Database System*, 25(1):1–42, 2000.
31. R. Guting, M. Bohlen, M. Erwig, C. Jensen, M. Schneider, N. Lorentzos, E. Nardelli, M. Schneider, and J. Viqueira. Spatio-Temporal Models and Languages: An Approach Based on Data Types. In *Spatio-Temporal Databases: The Chorochronos Approach, LNCS 2520*, Chap. 4, pp. 117–176, 2003.
32. M.A. Hammad, W.G. Aref, and A.K. Elmagarmid. Stream Window Join: Tracking Moving Objects in Sensor-Network Databases. In *Proceedings of 15th International Conference on Scientific and Statistical Database Management (SSDBM'03)*, pp. 75–84, 2003.
33. K. Hornsby and M.J. Egenhofer. Modeling moving objects over multiple granularities. *Annual Mathematics Artificial Intelligence*, 36(1–2):177–194, 2002.
34. B. Huang and C. Claramunt. Stoql: An ODMG-Based Spatio-Temporal Object Model and Query Language. In *Proceedings of the 10th International Symposium on Spatial Data Handling (SDH'02)*, pp. 225–237, 2002.
35. ISO/IEC. *Information Technology – Database languages – SQL – Part 7: Temporal (SQL/Foundation). ISO/IEC 9075-2 Working Draft.* ISO, 2001.

36. ISO/TC211. *Geographic Information and Temporal Schema. ISO 19108:2002.* ISO, 2002.
37. ISO/TC211. *Geographic Information and Spatial Schema. ISO 19107:2003.* ISO, 2003.
38. I. Kakoudakis. *The Tau Temporal Object Model*, M.Sc. Thesis, Umist, 1996.
39. P.C. Kanellakis, G.M. Kuper, and P. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51:26–52, 1995.
40. V. Khatri, S. Ram, and R. Snodgrass. Augmenting a Conceptual Model with Geospatiotemporal Annotations. *IEEE Transactions on Knowledge and Data Engineering*, 16:1324–1338, 2004.
41. B. Kuijpers, J. Paredaens, and D.V. Gucht. Towards a theory of movie database queries. In *Proceedings of the 7th International Workshop on Temporal Representation and Reasoning (TIME'00)*, pp. 95–102. IEEE Computer Society, 2000.
42. G. Kuper, L. Libkin, and J. Paredaens. *Constraint Databases*. Springer, Berlin Heidelberg New York, 2000.
43. S. Larrivée, Y. Bédard, and J. Pouliot. How to Enrich the Semantics of Geospatial Databases by Properly Expressing 3d Objects in a Conceptual Model. In *Proceedings of the Workshops On The Move to Meaningful Internet Systems*, number 3762 in LNCS. Springer, Berlin Heidelberg New York, 2005.
44. J.A.C. Lema, L. Forlizzi, R.H. Güting, E. Nardelli, and M. Schneider. Algorithms for moving objects databases. *The Computer Journal*, 46(6):680–712, 2003.
45. M.F. Mokbel, X. Xiong, W.G. Aref, S.E. Hambrusch, S. Prabhakar, and M.A. Hammad. Place: A Query Processor for Handling Real-Time Spatio-Temporal Data Streams. In *Proceedings of 30th International Conference on Very Large Data Bases (VLDB'04)*, pp. 1377–1380, 2004.
46. M.F. Mokbel, X. Xiong, M.A. Hammad, and W.G. Aref. Continuous query processing of spatio-temporal data streams in place. *GeoInformatica*, 9(4):343–365, 2005.
47. D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query Processing in Spatial Network Databases. In *Proceedings of 29th International Conference on Very Large Data Bases (VLDB'03)*, pp. 802–813, 2003.
48. D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group nearest neighbor queries. In *Proceedings of the 20th International Conference on Data Engineering (ICDE'04)*, pp. 301–312. IEEE Computer Society, 2004.
49. J. Paredaens, G. Kuper, and L. Libkin, editors. *Constraint databases*. Springer, Berlin Heidelberg New York, 2000.
50. C. Parent. A Framework for Characterizing Spatio-Temporal Data Models. In S.S. Y. Masunaga (ed.), *Advances in Multimedia and Databases for the New Century*, pp. 89–97. World Scientific, Singapore, 2000.
51. K. Patroumpas and T.K. Sellis. Managing Trajectories of Moving Objects as Data Streams. In J. Sander and M.A. Nascimento, editors, *Proceedings of 2nd International Workshop on Spatio-Temporal Database Management (STDBM'04)*, pp. 41–48, 2004.
52. N. Pelekis. *STAU: A Spatio-Temporal Extension to ORACLE DBMS*. Ph.D. Thesis, 2002.
53. N. Pelekis, B. Theodoulidis, I. Kopanakis, and Y. Theodoridis. Literature review of spatio-temporal database models. *Knowledge Engeneering Review*, 19(3):235–274, 2004.
54. N. Pelekis, B. Theodoulidis, Y. Theodoridis, and I. Kopanakis. An Oracle data cartridge for moving objects, laboratory of information systems, department of informatics, university of piraeus, unipi-isl-tr-2005-01, 2005. http://isl.cs.unipi.gr/db/ publications.html.
55. N. Pelekis, Y. Theodoridis, S. Vosinakis, and T. Panayiotopoulos. Hermes – A Framework for Location-Based Data Management. In *Proceedings of 10th International Conference on Extending Database Technology (EDBT'06)*, pp. 1130–1134, 2006.
56. D.J. Peuquet. Making space for time: Issues in space-time data representation. *Geoinformatica*, 5(1):11–32, 2001.
57. D. Pfoser. Indexing the trajectories of moving objects. *IEEE Data Engeneering Bullettin*, 25(2):3–9, 2002.
58. D. Pfoser and C.S. Jensen. Capturing the uncertainty of moving-object representations. In R.H. Güting, D. Papadias, and F.H. Lochovsky, (eds.), *Proceedings of the 6th International Symposium on Advances in Spatial Databases (SSD'99)*, Vol. 1651. *Lecture Notes in Computer Science*, pp. 111–132. Springer, Berlin Heidelberg New York, 1999.

59. D. Pfoser, C.S. Jensen, and Y. Theodoridis. Novel Approaches in Query Processing for Moving Object Trajectories. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB'00)*, pp. 395–406, 2000.
60. R. Price, N. Tryfona, and C. Jensen. Extended spatiotemporal uml: Motivations, requirements, and constructs. In *Journal of Database Management*, 11:14–27, 2000.
61. R. Price, N. Tryfona, and C. Jensen. *Extending UML for Space- and Time-Dependent Applications*. Idea Group Publishing, 2002.
62. S. Ram, R. Snodgrass, V. Khatri, and Y. Hwang. *DISTIL: A Design Support Environment for Conceptual Modeling of Spatio-temporal Requirements*, pp. 70–83. 2001.
63. P. Rigaux, M. Scholla, L. Segoufin, and S. Grumbach. Building a constraintbased spatial database system: Model, languages, and implementation. *Information Systems*, 28:563–595, 2003.
64. A.P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and Querying Moving Objects. In *Proceedings of the 13th International Conference on Data Engineering (ICDE'97)*, pp. 422–432. IEEE Computer Society, 1997.
65. R. Snodgrass, M. Böhlen, C. Jensen, and N. Kline. *Adding valid time to SQL/Temporal. ANSI X3H2-96-501r2, ISO/IEC JTC1/SC21/WG3 DBL MAD-146r2*, 1996.
66. R. Snodgrass, M. Böhlen, C. Jensen, and A. Steiner. *Adding transaction time to SQL/ Temporal: Temporal change proposal. ANSI X3H2-96-152r, ISO-ANSI SQL/ISO/IECJTC1/ SC21/WG3 DBL MCI-143.* ISO, 1996.
67. R. Snodgrass, M. Böhlen, C. Jensen, and A. Steiner. Transitioning Temporal Support in tsql2 to sql3. In *Temporal Databases: Research and Practice, LNCS 1399*, pp. 150–194, 1998.
68. *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, Vol. 2520 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 2003.
69. J. Su, H. Xu, and O.H. Ibarra. Moving Objects: Logical Relationships and Queries. In C.S. Jensen, M. Schneider, B. Seeger, and V.J. Tsotras, editors, *Proceedings of 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*, volume 2121 of *Lecture Notes in Computer Science*, pp. 3–19. Springer, Berlin Heidelberg New York, 2001.
70. Y. Theodoridis. Ten benchmark database queries for location-based services. *The Computer Journal*, 46(6):713–725, 2003.
71. G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM Transactions Database System*, 29(3):463–507, 2004.
72. N. Tryfona and C. Jensen. Conceptual data modeling for spatiotemporal applications. *GeoInformatica*, 3:245–268, 1999.
73. N. Tryfona, R. Price, and C. Price. *Spatiotemporal Conceptual Modeling.*, chapter 3, pp. 79–116, Berlin, 2003.
74. M. Vlachos, D. Gunopulos, and G. Kollios. Discovering Similar Multidimensional Trajectories. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, pp. 673–684. IEEE Computer Society, 2002.
75. N.V. de Weghe, F. Witlox, A.G. Cohn, T. Neutens, and P.D. Maeyer. Efficient storage of interactions between multiple moving point objects. In *OTM Workshops (2)*, pp. 1636–1647, 2006.
76. O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving Objects Databases: Issues and Solutions. In *Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM'98)*, pp. 111–122, IEEE Computer Society, 1998.
77. O. Wolfson, A.P. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 7(3):257–387, 1999.
78. J. Zhang and M. Goodchild. *Uncertainty in Geographical Information*. Taylor & Francis, New York, 2002.
79. E. Zimanyi, C. Parent, and S. Spaccapietra. *Conceptual Modeling for Traditional and Spatio-Temporal Applications – The MADS Approach*. Springer, Berlin Heidelberg New York, 2006.