Probabilistic Measurement of Uncertainty in Moving Objects Databases

Talel Abdessalem, Laurent Decreusefond
Ecole Nationale Supérieure des Télécommunications
Département Informatique et Réseaux, LTCI – UMR CNRS 5141
46, rue Barrault – 75013 Paris – France
{Talel.Abdessalem, Laurent.Decreusefond}@enst.fr

José Moreira
Universidade de Aveiro – Departamento de Electrónica e Telecomunicações, IEETA
Campus Universitário de Santiago – 3810 - 193 Aveiro – Portugal
jmoreira@det.ua.pt

July 11, 2005

Abstract

The representation of moving objects in spatial database systems has become an important research topic in recent years. As it is not realistic to track and store the location of objects at every time instant, one of the issues that has been raised in this domain has to do with handling uncertainty in the location of moving objects. There are several works proposing to enrich spatiotemporal languages with estimates about the validity of the answers to queries about the object's movement. Although, research in developing statistical tools to compute those estimations is nearly non-existent.

In this paper, we propose three statistical tools for computing probabilistic estimates about the location of a moving object at a certain time and show how to use them for evaluating probabilistic range queries. We also compare the three proposals in terms of expressivity and ability for dealing with the different semantics proposed in the spatiotemporal query languages literature. The focus is on applications dealing with the history about the spatiotemporal behavior of non-network constrained moving objects, for monitoring or data-mining purposes, for instance.

1 Introduction

The spatiotemporal databases research community is giving particular attention to moving objects applications. Real-time systems, using recent information about object's movement for anticipation of events in near future, or historical systems, recording information about object's movement during large periods of time for monitoring or data-mining purposes, are noteworthy examples. Both cases require functionality allowing answering questions of the kind where and when. Although, as it is not possible to continuously monitor and record the location of moving objects, the knowledge that may be captured and stored by computer systems is only a partial representation of the actual spatiotemporal behavior of real-world objects.

For that reason, there are numerous proposals for extending spatiotemporal query languages with adequate semantics for dealing with uncertainty in the location of moving objects. The goal is to incorporate functionalities allowing answering questions such as "Which were the moving objects that have been within a certain area during some period of time, with a probability of at least 80%". Answering such kind of queries requires methods for computing probabilistic estimates about the location of moving objects at a given time. Although, research in this domain is almost non-existent.

This paper deals with the computation of probabilistic estimates for the validity of answers to spatiotemporal queries. The focus is on systems representing the history about the movement of non-network constrained moving objects. We propose three statistical tools that allow obtaining realistic estimates for the location of a moving object at a certain time. The methods explore different features and are applicable under different conditions, according to the information that may be disclosed by real applications. We also explain how we can use these methods to implement different semantics proposed in the literature for dealing with uncertainty in the location of moving objects.

The remaining part of this paper is organized as follows. Section 2 presents an overview of current proposals for dealing with uncertainty in the location of non-network constrained moving objects. Section 3 depicts the two main approaches that we have delineated to solve the proposed problem and presents three methods for computing probabilistic estimates about the location of moving objects. Section 4 presents the tools that have been developed to test the proposed methods and puts in evidence the domain of application covered by each one. Finally, section 5 concludes the paper.

2 Uncertainty in objects movement representation

The representation of the objects' movement is inherently imprecise and therefore, answers to user queries based on such information may be inaccurate [8, 13]. Imprecision may be introduced by the measurement process – it may depend on the accuracy of the GPS, for instance –, or by the sampling approach, as depicted below. Notice that the imprecision refers to the spatial dimension only, as it is commonly assumed that measurement instruments are able to determine precisely the time a position sample is taken.

As an example, consider the case of a port authority dealing with a spread of toxic waste in the sea and querying a nautical surveillance system to know which ships have crossed the polluted zone during a specified time interval. Imagine that the ship responsible for the waste has actually followed the trajectory represented in Figure 1. The black dots represent four observations made during the specified time interval, the shaded region represents the polluted area and the hatched line a trajectory that might have been inferred from the observations.

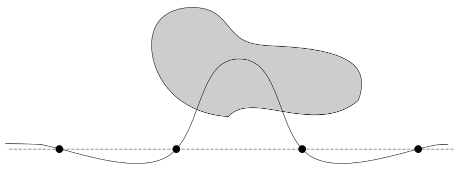


Figure 1: Uncertainty about moving objects trajectories

The hatched line does not cross the shaded region and, thus, an answer to a query based on this estimation of the trajectory would not include the guilty ship. On the contrary an answer may also include false candidates whose inferred trajectory crosses the area even though they have not actually been there.

2.1 Uncertainty of past, present and future positions

The preceding example focuses on the history of the objects' movement. In general, the focus may be put on the *past* movement or on the *future* movement of objects, depending on applications requirements. Two main approaches were raised.

The first approach [8, 6], focusing on past movements, addresses the needs of mining applications of spatiotemporal data: traffic mining, environment monitoring, etc. In this case, uncertainty is bounded using a sequence of positions of objects and some known physical constraints on their movement.

In the second approach [10, 11, 19, 13], the focus is put on the uncertainty about the future movement of objects. This approach addresses the needs of real-time applications and location-based services: real-time traffic control, real-time mobile workforce management, digital battlefields, etc. These systems make use of speed patterns information in the construction of future movements and uncertainty is fixed in advance. The latter allows avoiding frequent updates of the database. In fact, the database is not updated as long as the actual object's movement deviation from its expected location, as inferred from the information stored in the database, is less than the threshold previously fixed.

2.2 Bounding uncertainty

There are physical constraints on the movement of objects allowing limiting uncertainty of their position at a certain time. For instance, the uncertainty zone for a train moving on a railway is a section of the railway and for a ship moving freely in the ocean is an area.

When it comes to future movements, considering a two-dimensional space (Figure 2), the uncertainty area is a circle centered on the expected location of the object. The circle bounds the maximum deviation allowed for an object at a given time instant. Objects are committed to send a location update when the deviation reaches the bound.

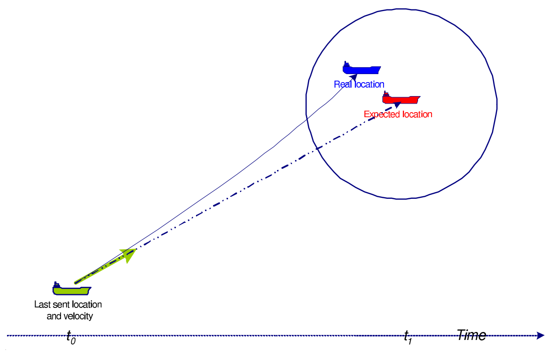


Figure 2: Expected location of a moving object

For past movements, since the positions between two consecutive samples are not measured, the best to do is to limit the possibilities of where the moving object could have been [9, 8, 7]. Let us consider a two-dimensional space and a moving object m, an instant t belonging to a time interval $\langle t_1, t_2 \rangle$ and two consecutive observations (p_1, t_1) and (p_2, t_2), (Figure 3). We denote by p_1 and p_2 the positions of the moving object at observation time instants t_1 and t_2, respectively. d denotes the distance between p_1 and p_2. At time t, the distance d_1 between m and p_1 is inferior to $r_1 = V_{\max} \times \Delta t_1$, where $\Delta t_1 = t - t_1$ and $V_{\max}$ is a user-defined value standing for the maximum velocity of moving object m. The distance d_2 between m and p_2, at instant t, is inferior to $r_2 = V_{\max} \times \Delta t_2$, where $\Delta t_2 = t_2 - t$. So, at time t, the moving object might be at any location within the area defined by the intersection of the two circles of radius r_1 and r_2. This is a so-called *lens* area [8] representing the set of all possible locations for a moving object at a certain time instant.

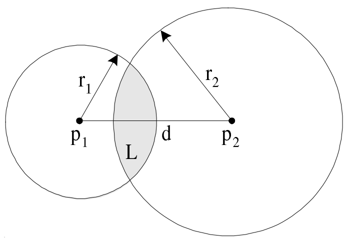


Figure 3: A *lens area* for a time instant

The set of all locations where a moving object might have been between two consecutive observations corresponds to an ellipsis (Figure 4). This means that the ellipsis covers all possible lens areas between the two consecutive observations [7, 2].

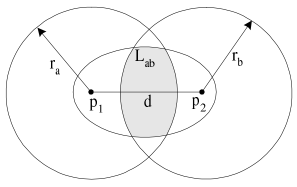


Figure 4: A *lens area* for a time interval between consecutive observations

Figure 4 shows how to compute the lens area for a time interval $\langle t_a, t_b \rangle$, between two consecutive observations. The circle with radius $r_b = V_{\max} \times (t_b - t_1)$ corresponds to the maximum distance from p_1 that could be reached by the moving object at t_b. The same reasoning applies to r_a. In addition, the moving object could not have been outside the ellipsis just defined. So, the lens area is defined by the intersection of the two circles with the ellipsis.

Lens areas for time intervals comprising one or more observations are the union of several lens areas computed using the method described above.

2.3 Using probabilistic methods

Consider now that we have methods that allow estimating the location of a moving object at any time instant. The evaluation of spatiotemporal query expressions could than be augmented with probabilistic estimates of the validity of answers to users queries. Thus, it would be possible to answer queries such as "Which are the planes for which the probability of being inside Area C within 5 minutes is at least 40%?", or "Which were the ships that were in a certain area during a given time interval, with a probably of at least 60%?".

Figure 5 illustrates this kind of query [8]. It considers the case of the anticipation of the location of an object moving on a two-dimensional space. If we assume that the distribution of probability in this lens area is uniform, then, the object is said to be within the given area with a probability of 30%, if at least 30% of its lens area is within that area.

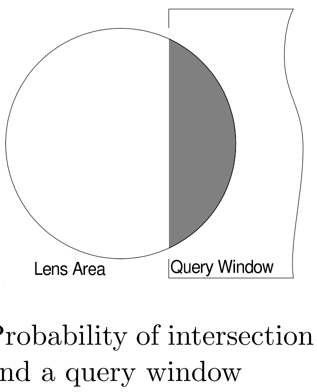


Figure 5: Probability of intersection between a *lens area* and a query window

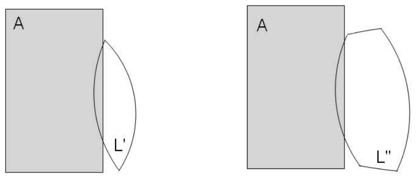
2.4 Related works

In recent years, uncertainty handling emerged as an important issue in moving object database research. Several aspects were investigated and two complementary models were proposed: [8] focusing on past objects' movement and [19, 13] dealing with future objects' movement. Moreover, [18, 17] investigated the communication cost for updating the database in the case of real-time applications. [5] discusses how the uncertainty of network constrained moving objects can be reduced by using reasonable modeling methods and location update policies. Finally, [9] added fuzziness in object location and considered the case of moving objects that may change their geometry in time.

An important issue of the current research activity in this domain is the design of a probabilistic model of uncertainty. The goal is to handle more realistic (non-uniform) distributions of probability on the location of moving objects, and measure the validity of the answers to user queries. Recent results [4, 3, 12] are going toward this goal, even if they just briefly touch upon the possibility of a non-uniform distribution.

Besides, the evaluation of user queries may require handling temporal types such as time instants, time intervals or a combination of them. Evidently, the probability of presence of a moving object within a given area at a certain time, should be less or equal than the probability of presence of the moving object within the same area during any time that includes the previous one. However, the methods presented in section 2.3 are suitable for dealing only with time instants and are not adequate to deal with time intervals.

For instance, consider figure 6 where A is an area of interest, L' is the lens area calculated for the location of an object at a time instant t *(1st case)* and L'' is a lens area for the location of the same object during a time interval $\langle t_a, t_b\rangle \supset t$ *(2nd case)*.



(1st case) Lens area for t *(2nd case)* Lens area for $\langle t_a, t_b\rangle \supset t$

Figure 6: Intersection of a lens area (L) with an area of interest (A)

Considering that the distribution of probability in the lens area is uniform, then the probability of presence of the moving object within A at time t is $P(t) = \frac{area(A \cap L')}{area(L')}$ and the probability of being within A during time $\langle t_a, t_b\rangle$ is $P(\langle t_a, t_b\rangle) = \frac{area(A \cap L'')}{area(L'')}$.

As $A \cap L'$ is equal to $A \cap L''$ and the area of L' is less than the area of L'' then, against evidence, $P(t)$ is greater than $P(\langle t_a, t_b\rangle)$.

3 Probabilistic reasoning

This section presents the statistical tools that we propose for the evaluation of probabilistic estimates about the location of moving objects. We will denote the probability of presence of a moving object within a given region during a certain time, simply as $P(t)$. We only consider the past objects' movement and we assume that it is represented as an ordered sequence of observations, denoted $\{(t, p)\}$, where p is a two-dimensional value denoting the location of the object at time instant t. We also consider that the objects move freely in space with no obstacles or networks constraining their movements.

It is also important to notice that the movement of real-world objects is not random. Indeed, it is reasonable to consider that, most often, their movement is smooth, i.e., that the movement between two locations is approximately linear and uniform. This means that the locations in the neighborhood of the position expected for an object[1], denoted $\tilde{p}$, should have a greater weight in the computation of the probabilities, than those locations that are far from $\tilde{p}$.

As we are interested in obtaining realist estimates, it is desirable that the density functions for distribution of the probabilities within lens areas take this feature into account. Moreover, the location of a moving object at the time instants corresponding to the observations is precisely known, and thus, we must have $P(t) = 1$ for those time instants or any time interval containing them. Finally, we also assume that there are systems for which it is not reasonable to estimate in advance the maximum velocity of a moving object with an acceptable accuracy. We propose a specific method for those cases.

We have investigated two main guidelines for the implementation of the proposed statistical tools, which were designated by *point-based* and *trajectory-based* approaches. We have developed methods based on each of these approaches and we will put in evidence the strengths and weaknesses of each one.

3.1 Point-based approach

As referred in section 2.4, there are several authors suggesting using a density function to estimate the probabilities of presence of a moving object at each point inside the lens area. Then, assuming that t denotes a time instant, the values of $P(t)$ may be calculated using the weight of the intersection of the lens area with the region considered, as shown in formula (1).

[1]The position expected for a moving object at a certain time instant is estimated assuming that the movement between consecutive observations is linear and uniform.

$$P(t) = \frac{W_{LensArea(t) \cap Region}}{W_{LensArea(t)}} \tag{1}$$

The main issue is the definition of a density function over a complex form, such as a lens area. Since we can easily define a density function over a circle, we propose to perform an anamorphosis of the lens area into a circle. The method that we propose consists in four steps:

- First, we define a local coordinates system for the lens area, to make the formulation of the lens area equation easier.
- Second, we define the anamorphosis of the lens area into a circle of radius 1, to which the chosen density of probability will be associated.
- Third, we define how to transform the density of probability over the circle of radius 1 into a density of probability over the lens area.
- Finally, we complete the process by the evaluation of $P(t)$.

3.1.1 Lens area equations

The lens area is given by the intersection of two circles. If one circle contains the other, the result of the intersection is the smaller circle. This situation may arise for time instants in the neighborhood of the instants of observations. Otherwise, the result of the intersection is an area similar to the one depicted in figure 7.

The lens area in figure 7 delimits the set of all possible locations for a moving object at time instant t, between two consecutive locations: p_0, observed at instant t_0, and p_1, observed at t_1. As presented in section 2, the left centered circle delimits the lens area of the object during the time interval $[t_0, t]$. The right centered circle delimits the lens area of this object during

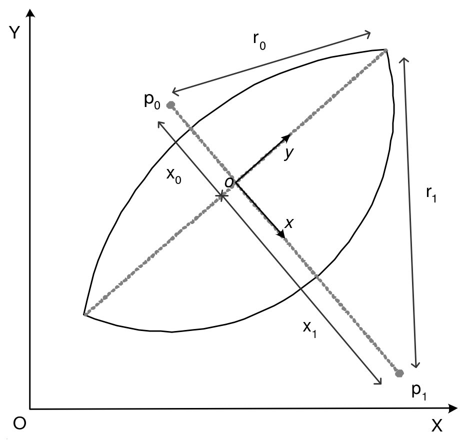


Figure 7: Lens area parameters

the time interval $[t, t_1]$. The radiuses of the circles depend on the maximum velocity (V_{max}) previously estimated:

$$\begin{aligned} r_0 &= V_{max} \times (t - t_0) \\ r_1 &= V_{max} \times (t_1 - t) \end{aligned} \quad (2)$$

To make the formulation of the lens area parameters easier, we use a local coordinates system based on the lens area axes (see figure 7). Let us consider d as the distance between p_0 and p_1. The origin o of the coordinates system corresponds to the projection of the lens area summits over the x-axis. Formally, o is considered as the center of mass, i.e. the barycenter[2], of points p_0 with mass $\frac{x_1}{d=(x_1-x_0)}$, and p_1 with mass $\frac{x_0}{d=(x_1-x_0)}$, and is defined as:

$$o\begin{pmatrix} 0 \\ 0 \end{pmatrix} = bar\left\{ \left(p_0 \begin{pmatrix} x_0 \\ 0 \end{pmatrix}, \frac{x_1}{d}\right), \left(p_1 \begin{pmatrix} x_1 \\ 0 \end{pmatrix}, \frac{x_0}{d}\right) \right\} \quad (3)$$

[2]Consider two points A_1 and A_2 defined by their cartesian coordinates (x_1, y_1) and (x_2, y_2). The mass, also referred to as the weighting coefficients, for each point is m_1 and m_2, respectively. The *barycenter* of $((A_1, m_1), (A_2, m_2))$ is a point d with cartesian coordinates (x_g, y_g) such as: $x_g = \frac{m_1 x_1 + m_2 x_2}{m_1 + m2}$ and $y_g = \frac{m_1 y_1 + m_2 y_2}{m_1 + m2}$.

Applying the Al-Kashi's theorem (law of cosines) [1, 16], we obtain the following abscises for p_0 and p_1 in the local coordinates system:

$$\begin{aligned} x_0 &= -\frac{d^2 + {r_0}^2 - {r_1}^2}{2d} \\ x_1 &= \frac{d^2 - {r_0}^2 + {r_1}^2}{2d} = d + x_0 \end{aligned} \quad (4)$$

In the general case, the lens area is defined by two arcs located in the half-plans $x > 0$ and $x < 0$. Otherwise, it is a circle centered on p_0 or p_1. So, we use the following explicit equations to represent the lens area:

$$f_L(y) = \begin{cases} x_1 - \sqrt{{r_1}^2 - y^2} & \text{, in the general case} \\ x_0 - \sqrt{{r_0}^2 - y^2} & \text{, if } x_1 - \sqrt{{r_1}^2 - y^2} > 0 \end{cases}$$

$$f_R(y) = \begin{cases} x_0 + \sqrt{{r_0}^2 - y^2} & \text{, in the general case} \\ x_1 + \sqrt{{r_1}^2 - y^2} & \text{, if } x_0 + \sqrt{{r_0}^2 - y^2} < 0 \end{cases} \quad (5)$$

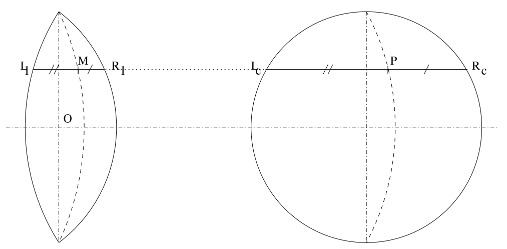
Now, we can easily change from the global coordinates system to the local one, by a translation of vector $\begin{pmatrix} -o.X \\ -o.Y \end{pmatrix}$, that places the origin at o, followed by a rotation of an angle θ, such as:

$$\begin{cases} \cos\theta = \frac{p_1.X - p_0.X}{d} \\ \sin\theta = \frac{p_1.Y - p_0.Y}{d} \end{cases} \quad (6)$$

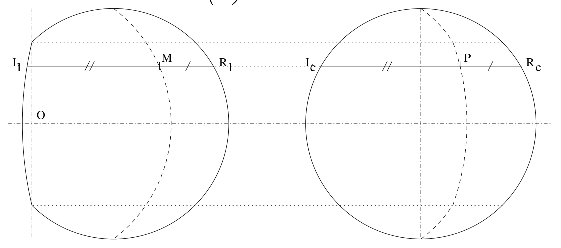
3.1.2 Lens area anamorphosis

As referred above, defining a density function over complex objects such as lens areas would not be a simple task. To cope with this problem, we propose using an anamorphosis, to transform the lens area into a circle of radius 1 (figure 8). The density function will be then defined over the circle.

To achieve such transformation, we define an affine bijection [15, 14] between the lens area and the circle. This one-to-one transformation preserves collinearity (i.e., all points lying on



(a) 1st case



(b) 2nd case

Figure 8: Lens area anamorphosis

a line initially still lie on a line after transformation), as well as the ratios of masses and distances (i.e., the barycenter of a line segment stills the barycenter of the corresponding line segment after transformation). So, determining the point in the circle that corresponds to a point in the lens area, is formally defined as follows:

- Consider a point $M\begin{pmatrix} x \\ y \end{pmatrix}$ that belongs the lens area (figure 8).
- Using the explicit equations (5), we can define the endpoints of the line segment containing M, as $L_l\begin{pmatrix} x_L = f_L(y) \\ y \end{pmatrix}$ and $R_l\begin{pmatrix} x_R = f_R(y) \\ y \end{pmatrix}$.
- Supposing that M is the barycenter of the line segment $\overline{L_l R_l}$, then M must verify equation (7). This means that mass coefficients of L_l and R_l are proportional to their distance from M.

$$M = \ bar\ \left\{\left(L_l, \frac{x_R - x}{x_R - x_L}\right), \left(R_l, \frac{x - x_L}{x_R - x_L}\right)\right\} \quad (7)$$

- Let us denote the maximum height of a lens area by r. Depending on the shape of the lens area, r may be equal to the length of the line segment between the summits of the lens area (figure 8(a)), or it may be equal to the radius of the smaller of the two circles that define the lens area. The latter occurs for time instants near to the instants of observations, when the lens area is a circle or when it looks like the one presented in figure 8(b). Equation 8 shows how to calculate r.

$$r = \begin{cases} \sqrt{{r_0}^2 - {x_0}^2} & \text{, in the general case} \\ r_0 & \text{, if } x_0 > 0 \\ r_1 & \text{, if } x_1 < 0 \end{cases} \quad (8)$$

- The points L_c and R_c in the boundary of the unity disk (figure 8), that correspond to the points L_l and R_l in the boundary of the lens area, are defined as follows:

$$L_c\begin{pmatrix} -\sqrt{1 - \frac{y}{r}^2} \\ \frac{y}{r} \end{pmatrix} \text{ and } R_c\begin{pmatrix} \sqrt{1 - \frac{y}{r}^2} \\ \frac{y}{r} \end{pmatrix} \quad (9)$$

- Point P in the unity disk is the image of M, obtained by an affine one-to-one transformation of line segment $\overline{L_l R_l}$ in the lens area into the line segment $\overline{L_c R_c}$ in the unity disk:

$$P = \ bar\left\{\left(L_c, \frac{x_R - x}{x_R - x_L}\right), \left(R_l, \frac{x - x_L}{x_R - x_L}\right)\right\} \quad (10)$$

- So, P is the image of M obtained by an anamorphosis α such as:

$$M\begin{pmatrix} x \\ y \end{pmatrix} \xrightarrow{\alpha} P\begin{pmatrix} \frac{2x - x_L - x_R}{x_R - x_L}\sqrt{1 - \left(\frac{y}{r}\right)^2} \\ \frac{y}{r} \end{pmatrix} \quad (11)$$

3.1.3 Defining the density function

The density of probability over the lens area is then defined throw the density of probability over the unity disk. However, we cannot simply apply the transformation and keep the probability corresponding to each point, since the differential surface element has been changed by the anamorphosis:

$$\int_L f\left(\alpha\left(x,y\right)\right)\,dx\,dy \neq \int_D f\left(x,y\right)\,dx\,dy = 1 \qquad (12)$$

where, L stands for the lens area and D stands for the unity disk.

Hence, the density of probability must be normalized to guarantee that the whole probability is equal to 1. As in computation it is not possible to deal with infinite sets, we have introduced the notion of granularity g in this model. We define g as the distance between two consecutive points (granules). So, the coordinates of each point become multiples of g. We define the weight of a point as its corresponding density of probability over the unity disk. Considering all the points in a lens area, the sum of their weights gives the total weight of the lens area W_{tot}.

$$W_{tot} = \sum_{\substack{m>0 | m\,g - x_0 < r_0 \\ m<0 | x_1 - m\,g < r_1}} \; \sum_{n | (m\,g-x_0)^2 + (n\,g)^2 < {r_0}^2} f\left(\alpha\left(m\,g, n\,g\right)\right) \qquad (13)$$

The probability associated to each point is then defined as the ratio of its weight over the total weight of the lens area.

$$P\left(M\begin{pmatrix} m_M\,g \\ n_M\,g \end{pmatrix}\right) = \frac{f\left(\alpha\left(m_M\,g, n_M\,g\right)\right)}{W_{tot}} \qquad (14)$$

3.1.4 Probability of presence of an object within a region

Considering a region Z and a lens area L, then $P(t)$ is proportional to the weight of the intersection zone $L \cap Z$. As the space has been decomposed into granules, the value of $P(t)$ may be easily calculated by summing the weight of the points within L that intersect Z:

$$P(t) = \frac{1}{W_{tot}} \sum_{(m\,g, n\,g) \in L \,\cap\, Z} f\left(\alpha\left(m\,g, n\,g\right)\right) \qquad (15)$$

The computation of this probability may be performed simultaneously with the computation of the total weight of the lens area. The weight of each point in the intersection zone is added simultaneously to the weight of the lens area W_{tot} and to the weight of the intersection zone $W_{L \cap Z}$. Then, $P(t)$ is obtained as follows:

$$P(t) = \frac{W_{L \cap Z}}{W_{tot}} \qquad (16)$$

3.2 Trajectory-based approach

The *Trajectory-based* approach consists in the generation of a large number of trajectories between each two consecutive observations recorded for the object. The probability of presence of a moving object within a given region, during a time interval Δt, is then estimated by the number of trajectories intersecting that region, over the total number of trajectories generated (17).

$$P(\Delta t) = \frac{\#_{trajectories(\Delta t)\,\cap\, region}}{\#_{trajectories\ generated}} \qquad (17)$$

Let us now consider a motion section defined by three consecutive observations, as shown in figure 9. Let $\mathcal{T}_{0,1}$ (respectively $\mathcal{T}_{1,2}$) be the set of the $N_{0,1}$ (respectively $N_{1,2}$) trajectories generated for the first (respectively second) step of the motion section. Let $\mathcal{Z}_{0,1}$ (respectively $\mathcal{Z}_{1,2}$) be the subset of the $K_{0,1}$ (respectively

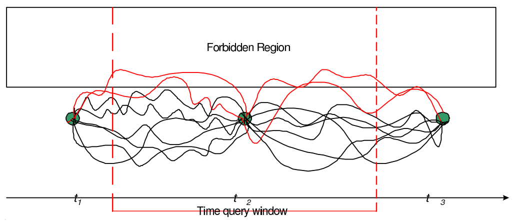


Figure 9: Trajectory-based approach

$K_{1,2}$) trajectories that do not intersect the forbidden region. The obtained set of trajectories over the two steps of the motion section is then $\mathcal{T}_{0,2} = \mathcal{T}_{0,1} \times \mathcal{T}_{1,2}$, and the subset of the trajectories that do not intersect the forbidden region is $\mathcal{Z}_{0,2} = \mathcal{Z}_{0,1} \times \mathcal{Z}_{1,2}$. Thus, the probability of presence of the moving object in the given region during the query window Δt may be calculated as follows:

$$P_{0,2}(\Delta t) = \frac{\#(\mathcal{T}_{0,2} - \mathcal{Z}_{0,2}(\Delta t))}{\#\mathcal{T}_{0,2}} = \frac{\#\mathcal{T}_{0,2} - \#\mathcal{Z}_{0,2}(\Delta t)}{\#\mathcal{T}_{0,2}}$$

$$P_{0,2}(\Delta t) = 1 - \frac{\#(\mathcal{Z}_{0,1}(\Delta t) \times \mathcal{Z}_{1,2}(\Delta t))}{\#(\mathcal{T}_{0,1} \times \mathcal{T}_{1,2})}$$

$$P_{0,2}(\Delta t) = 1 - \frac{\#\mathcal{Z}_{0,1}(\Delta t)\#\mathcal{Z}_{1,2}(\Delta t)}{\#\mathcal{T}_{0,1}\#\mathcal{T}_{1,2}} = 1 - \frac{K_{0,1}}{N_{0,1}}\frac{K_{1,2}}{N_{1,2}}$$

Hence:

$$P_{0,2}(\Delta t) = 1 - \overline{P}_{0,1}(\Delta t)\overline{P}_{1,2}(\Delta t)$$

For the general case, where $0 < i < j$, we obtain:

$$P_{i,j}(\Delta t) = 1 - \prod_{k \in [i,j-1]} \overline{P}_{k,k+1}(\Delta t) \qquad (18)$$

The key issue for this approach is the development of a generator for movement data. The generated movements should be random, but they must comply with some physical constraints on the movement of real world objects.

We have implemented two kinds of generators that we designate by the Brownian motion generator and the vector-oriented motion generator. Only the latter requires knowing the maximum velocity of the object.

3.2.1 Brownian motion generator

The Brownian motion is originally used to describe the movement of particles that receive a random number of impacts of random strength, from random directions, during a certain period of time. The movement of the particles between two impacts is linear and uniform. No other interaction with the particles exists. This theory has been firstly set by Robert Brown in 1827, when observing pollen particles floating on water. It is applied today to many different domains, like financial assets modeling or signal processing. In the later the Brownian motion theory is used to simulate noise associated to processed signals. A brown noise (or "Brownian"), is a noise in which each successive sample is a small random increment or decrement above the previous sample. We follow a similar process to generate our Brownian movements.

Brownian movements

In our case, Brownian movements are generated between consecutive observations. So, the origin and the destination of each generated movement must coincide with the given observations. The Brownian movement generation is then done in a suite of constant steps between the origin and the destination. The number of steps is fixed in advance.

We propose two generators of Brownian movements. The first one generates *one-dimensional Brownian movements.* This kind of movement don't enable backward steps. The moving object executes a suite of constant steps following the same direction: from the origin to the destination. Time is also decomposed on a regular (uniform) basis. Within each step the projection of the movement over the line defined by the origin and destination is uniform. However, the movement over the perpendicular directions is directed by a Brownian law. Figure 10 shows a representation of the so-called one-dimensional Brownian motion.

The second generator we propose combines two one-dimensional Brownian motions

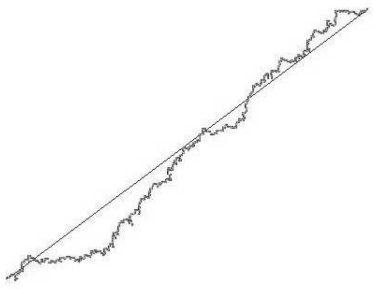


Figure 10: One-dimensional Brownian motion

to obtain a random movement, called *two-dimensional Brownian movement*. There is no longer regularity between the generated movement steps. In particular, we can observe backward movement steps. Thus, the movement is more realistic and the set of trajectories generated covers an area that is closer to the shape of a lens area, than that one covered by the one-dimensional generator. This kind of movement is illustrated in figure 11.

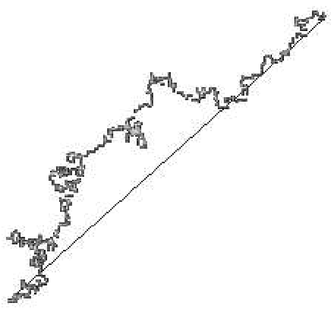


Figure 11: Two-dimensional Brownian motion

Implementation

To implement the generators described above, we define the Brownian as follows:

$$\begin{cases} B_0 = 0 \\ B_k = B_{k-1} + \sqrt{p}\,.\,N(0,1) \qquad \forall k \in [1, n-1] \end{cases} \tag{19}$$

where n is the number of points of the Brownian (which corresponds to the number of intermediate points between the origin and the destination), p is the fixed step length and $N(0,1)$ is a generator of random values accordingly to a Gaussian law, with an average null and a variance equal to 1.

To constraint the movement origin and destination, the Brownian generator must fulfill the following criteria: $\tilde{B_0} = \tilde{B}_{N-1} = 0$. This is obtained in the following equation:

$$\tilde{B_k} = B_k - k\frac{B_{n-1}}{n-1} \qquad \forall k \in [0, n-1] \tag{20}$$

The advantage of the one-dimensional Brownian motion is that it is simple to implement. This movement is generated easily using a Brownian value to which we apply a rotation and a translation. The obtained vector is then added to an initial movement vector obtained on the basis of uniform movement following the axis origin-destination.

Let τ be the path generated between two points $A(x_a, y_a)$ and $B(x_b, y_b)$ and τ_d the path from A to B following a straight line:

$$\begin{cases} \tau_d(0) = A \\ \tau_d(n-1) = B \\ \tau_d(k) = A + k\frac{\vec{AB}}{n-1} \qquad \forall k \in [0, n-1] \end{cases} \tag{21}$$

The one-dimensional movement is then calculated for each step using the following formula, where σ is a user defined parameter that enables amplifying the Brownian values:

$$\tau(k) = \tau_d(k) + trans(rot(\sigma\,.\,\tilde{B_k})),\ \forall k \in [0, n-1] \tag{22}$$

The generation of two-dimensional Brownian movement requires using four parameters $\sigma_{[2,2]}$ instead of a single σ, and two independent Brownians.

$$\tau(k) = \tau_d(k) + \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix} . \begin{pmatrix} \tilde{B}_k^1 \\ \tilde{B}_k^2 \end{pmatrix} \tag{23}$$
$$\forall k \in [0, n-1]$$

The shape of the movements generated is influenced by the four parameters $\sigma_{[2,2]}$. The parameters in the first line have an influence on the shifting of positions in the direction of the line connecting the origin and destination, and the other parameters have an influence on the shifting of positions in the perpendicular direction.

3.2.2 Vector-oriented motion generator

This movement generator may be suitable for applications for which the Brownian movements doesn't represent a realistic solution. Since there is no restrictions for the changing of cape between two successive steps of a Brownian movement, instantly U-turns are made possible. This is not adequate for moving objects like fishing ships, for instance.

Vector-oriented movements

A vector-oriented movement is a movement composed of a suite of steps, each one is calculated using a random speed and orientation values. The main issue for this approach is the convergence of the generated movements towards the chosen destination, without interfering with the desirable random features of the generated values. To deal with this problem, we have defined a circumference with a center at the origin and radius equal to the distance between the origin and the destination. Then the movement generated must converge towards the circumference. For each step, the cape angle is generated accordingly to a distribution centered towards the closest point of the circumference. Once a point of the circumference is attained, it is enough to perform a rotation to make the last point generated coincide with the destination intended (figure 12).

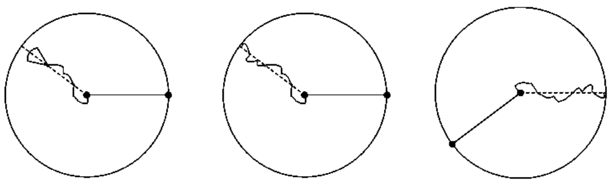


Figure 12: Generation of vector-oriented movements

Implementation

The time interval between two observations is decomposed into steps representing the smallest unit for the simulation. All steps have the same duration. New orientation and speed values are generated at each step accordingly to the following criteria:

- for the cape angle, the distribution is centered towards the closest point of the circumference. Considering that $\alpha_{avg}(n)$ is the mean value of the angles generated for steps 1 through $(n-1)$, the new orientation $\alpha(n)$ generated at step n is bounded by Δ_{max}, as follows:

$$\alpha(n) \in [\alpha_{avg}(n) - \Delta_{max}, \alpha_{avg}(n) + \Delta_{max}] \quad (24)$$

- for the speed value at step n, the distribution is centered on the average speed $v_{avg}(n)$, which should be equal to the average speed required to reach the closest point of the circumference.

Our experiments revealed that the trajectories generated converge quickly towards the neighborhood of the circumference, but after that, they turned around during a certain time before reaching it.

To cope with this situation, we tried to decrease the variance of the random variables over time accordingly to a chosen law (linear, quadratic or exponential). In this way, convergence is imposed artificially but the results obtained were satisfactory for our model.

Finally, it may arise that, during the generation of a movement, we obtain a average speed value greater than the maximum speed allowed. In this case, that generated part of the movement is simply discarded and a new one is generated.

4 Tools and applications

This section compares the proposed methods and introduces some issues related with their application for the implementation of spatiotemporal operations. As an example, we show how to use them for evaluating probabilistic range queries.

4.1 Comparison of the methods

The point-based and the vector-oriented methods are applicable to systems where it is possible to estimate the maximum velocity of the objects in advance. Otherwise, the Brownian motion method should be used.

The *point-based* method allows estimating values for $P(t)$, only when t is a time instant. Extending this method to cope with time intervals is not trivial. We could conceive a method based on the notion of a temporal granularity, as we did for space in section 3.1.3. The method would consist in the specification of a sequence of lens areas, one for each temporal granule within the desired time interval, and building a density function for each lens area, using the point-based method proposed in this paper. Then, the density function for a time interval would result from a combination of density functions defined for each lens area. The problem is that the probability for the presence of a moving object at a certain location depends on its location at previous instant. As the location of the moving object at the previous instant may be one within a possibly large set of points, we were not able to find a solution for such a complex problem, allowing maintaining the desirable properties from the statistical point of view.

The *vector-oriented motion* method allows estimating values for $P(t)$, where t is a time instant or a time interval. The method is also simpler than the previous one, but a definite choice should only be made after benchmarking for performance and reliability evaluation purposes. The former, benchmarking for performance evaluation, would indicate which method is the most efficient from response times point of view. The latter concerns the quality of the estimations. This is an interesting issue, also raised in [3], as we are interested in comparing the reliability of the results of probabilistic queries, but in fact we do not know which are the true results.

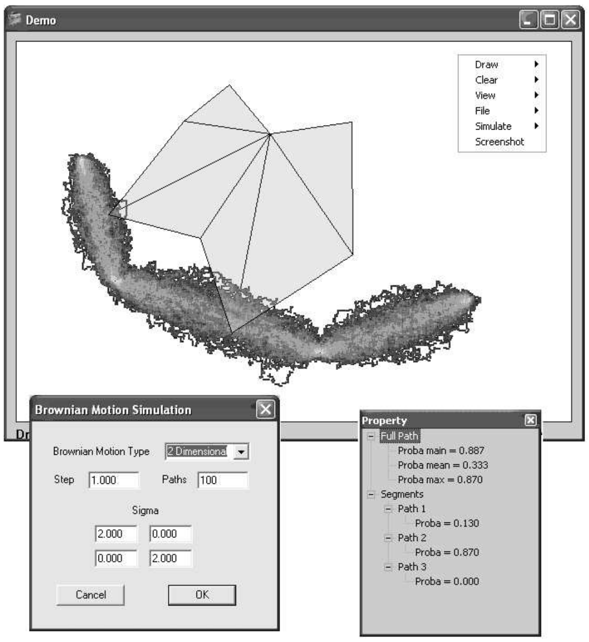
The *Brownian motion generator* should be used for systems where the maximum velocity of the moving objects cannot be estimated accurately in advance. Under these circumstances, it is not possible to define lens areas. Consequently, it is not possible to guaranty that when $P(t) = 0$, the moving object has not been within the region of interest during the specified time instant or time interval t. As we will see below this feature restricts the applications of this method.

4.2 Tools and functionality

We have developed two tools for the evaluation of the proposed methods: one implementing the Brownian motion generator and the other implementing the methods based on the knowledge of the maximum velocity.

The Brownian motion generator tool has been implemented in C language and runs on Windows systems (figure 13).

The tool allows drawing polygons and paths. Polygons are shown in its triangulated form. The paths are line segments defined by a sequence of points. The parameters for each sim-



```
Brownian Motion Type   2 Dimensional
Step   1.000        Paths   100

              Sigma
        2.000     0.000
        0.000     2.000
```

```
Full Path
    Proba main = 0.887
    Proba mean = 0.333
    Proba max = 0.870
Segments
    Path 1
        Proba = 0.130
    Path 2
        Proba = 0.870
    Path 3
        Proba = 0.000
```

Figure 13: Brownian motion tool

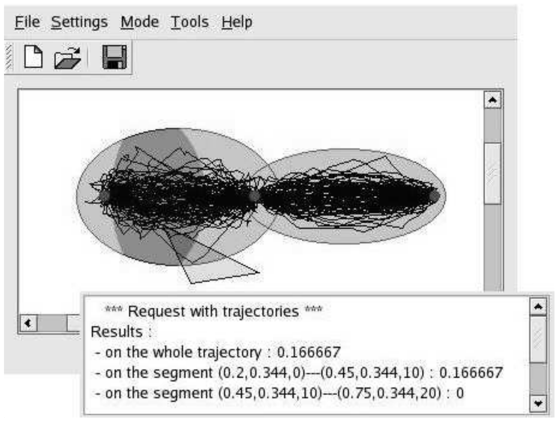
ulation are entered in the simulation window, where *Brownian motion type* allows choosing between one or two-dimensional Brownian motions, *Paths* stands for the number of trajectories that should be generated, *Step* defines the number of motion sections between each pair of consecutive points, and *Sigma* are the coefficients that direct the spatial distribution of the generated values.

The color of the pixels in the neighborhood of the paths that could have been followed by a moving object are darker for smaller probabilities and lighter for higher probabilities.

The results are displayed in the property window, where *Proba main* denotes the value estimated for the probability of the presence of the object within the region defined by the polygon during the whole simulation, and *proba* denotes the values estimated for each individual segment.

Notice that the *Sigma* parameters may be adjusted to mimic the features of real-world applications. We have tried to establish a policy for controlling these parameters in order to apply this solution for cases where the maximum speed is known as well. Although, we were not succeeded in finding a solution that would allow preserving the desirable statistical properties of the generated values and giving realistic answers to user queries.

The second tool implements the point-based and the vector-oriented methods. It has been developed in C language and runs on Linux systems (figure 14).



```
 *** Request with trajectories ***
Results :
 - on the whole trajectory : 0.166667
 - on the segment (0.2,0.344,0)---(0.45,0.344,10) : 0.166667
 - on the segment (0.45,0.344,10)---(0.75,0.344,20) : 0
```

Figure 14: Vector-oriented motion tool

The functionalities offered here are approximately the same as those offered by the previous tool. In addition, as this tool implements the methods based on the maximum velocity, it also allows drawing lens areas, for time intervals defined between two consecutive observations (the ellipsis) or for arbitrary time intervals (the area defined by the intersection of two circles). Another difference is the displaying method. This tool does not use a colored scale for representing the probabilities. Hence, the zones of higher or lower probabilities should be inferred visually, considering the density of the trajectories displayed.

Previous figure shows that there are locations within a lens area, which are not crossed by any trajectory. As the probabilities inferred for those locations would be zero, this is not adequate to implement some operations proposed in the spatiotemporal query languages literature. To overcome this problem it is required to consider that every location within a lens area has an initial probability equal to a very small value.

4.3 Application to spatial-temporal query languages

The application of the proposed methods to answer probabilistic range queries, such as "Which were the ships that were in a certain region during a time interval, with a probability of at least 30%" [8, 4] could be expressed as follows in a SQL-like language:

select objId
from movingObjects
where probWithin(objMotion, region, time) > 0.3;

In this example, *movingObjects(objId, objMotion)* is a table holding the information about the spatiotemporal behavior of the moving objects[3], and *probWithin* is a function implemented using one of the methods proposed in this paper.

Notice that, queries defining that a probability is equal to some precise value are not meaningful, as these values are merely estimations for what could have happen in a real situation. Although, the methods have been developed and tested for the validity of the bounding values, i.e., when $P(t) = 0$ and $P(t) = 1$. For instance, when one of the observations falls into a particular region during the specified time interval, all methods assure that $P(t) = 1$. For methods grounding on a maximum velocity value, it is also possible to guaranty that when $P(t) = 0$, the moving object has not been within the specified zone during that time.

[3]For simplicity, we assume in this example that the movement of an object, denoted *objMotion*, is defined as an abstract data type, but other kinds of representation might have been used.

Consequently, these methods may also be used to implement semantics adequate to answer questions such as "Which were the moving objects that surely have been within a specified region during a certain time" or "The list of all moving objects that could have been within a specified region", as proposed in [6].

The same does not hold for Brownian motion, as the parameters used for carrying out this method do not allow establishing an accurate bounding of lens areas.

It is also possible to implement many other semantics. For instance, one may be interested in defining a probable operator and consider that probable means a probability of at least 80%, unlikely means at most 20%, or that a query yields true if the probability is at least 60%, among many others reasoning possibilities.

5 Conclusion and future work

This paper deals with the computation of estimates about spatiotemporal events for moving objects systems. The focus is on the history of object's movement and it is assumed that the objects move freely in space.

We have followed two main approaches, a point-based and a trajectory-based approach, and proposed three methods, with different features, that may be applied to a wide range of moving objects systems. We succeeded in obtaining non-uniform statistical distributions for estimating the location of moving objects, congruent with the features of real-world systems. These methods can be used to put into practice the operations proposed in different query languages for dealing with uncertainty in the location of moving objects.

We also expect that the issues presented in this paper could give useful insights for developing new solutions in the future.

At a first glance, it seamed that the point-based approach was the most obvious solution for estimating the probability of presence of a moving object within a certain region. This also was the solution that was envisaged in the literature, but, as previous section shows, the method based on this approach is considerably more difficult to implement than the others. Besides, this method can only be applied to answer probabilistic queries about the location of a moving object at a certain time instant, i.e., time intervals are not allowed.

On the other hand, trajectory-based methods are simpler to implement and they are able to cope with a greater variety of applications: it is possible to answer probabilistic queries about the location of a moving object during any temporal domain specified by users, and it is also possible to develop methods that do not require knowing the maximum velocity in advance.

The purpose of this work was to look for statistical tools suitable for the implementation of probabilistic spatiotemporal queries. The emphasis was on feasibility – identification of approaches and development of methods to put them into practice – and expressivity – which are the domains of application and the limitations, when applying these methods to the implementation of the different semantics proposed in the literature for dealing with uncertainty –.

Performance and reliability issues were left for future work. This task involves establishing policies for fine-tuning the controllable parameters defined for each method, in order to obtain the best response times without compromising the desired reliability of the results. To fulfil this task and to put in evidence the strengths and the weaknesses of each method, a benchmarking methodology, defining appropriate metrics for performance and reliability, is required. Other related tasks include looking for adequate filter steps for eliminating a large number of non-qualifying candidates, using methods computationally non-exigent. This issue is also related with the development of data structures and efficient access methods for movement data.

Acknowledgments

We would like to thank Cyril Branciard, Aurélien Coquard, Yvain Thonnart, Nghoc-Minh Vo, Cédric Lacage, Haykel Tagourti and Thomas Sirvent for their precious collaboration. They have worked on the development and implementation of the statistical tools presented in this paper, and their contribution was fundamental for the achievement of this work.

References

[1] Theoreme d'Al-Kashi. *Wikipedia, l'encyclopedie libre, http://fr.wikipedia.org.*

[2] T. Abdessalem, C. du Mouza, J. Moreira, and P. Rigaux. *Management of Large Moving Objects Databases: Indexing, Benchmarking and Uncertainty in Movement Representation*, chapter 10, pages 225–249. In Y. Manolopoulos, A. Papadopoulos and M. Vassilakopoulos eds, Spatial Databases: Technologies, Techniques and Trends. IDEA Group Publishing, 2005.

[3] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD'03, Proc. of ACM SIGMOD International Conference on Management of Data*, San Diego, USA, June 9–12 2003.

[4] R. Cheng, S. Prabhakar, and D. Kalashnikov. Querying Imprecise Data in Moving Object Environments. In *ICDE 2003, Proc. of the 19th IEEE International Conference on Data Engineering*, Bangalore, India, March 5–8 2003.

[5] Z. Ding and R. H. Güting. Uncertainty management for network constrained moving objects. In *DEXA 2004, Proc. of 15th International Conference on Database and Expert Systems Applications*, pages 411–421, Zaragoza, Spain, 2004.

[6] J. Moreira, C. Ribeiro, and T. Abdessalem. Query Operations for Moving Objects Database Systems. In *Proc. of the 8th International Symposium on Advances in Geographic Information Systems (ACMGIS-00)*, pages 108–114. ACM Press, 2000.

[7] J. Moreira, J.-M. Saglio, and C. Ribeiro. Representation and manipulation of moving points: An extended data model for location estimation. *Journal of Cartography and Geographic Information Systems (CaGIS), ACSM*, 26(2):109–123, 1999.

[8] D. Pfoser and C. Jensen. Capturing the uncertainty of moving-object representations. *Lecture Notes in Computer Science*, 1651:111–132, 1999.

[9] D. Pfoser and N. Tryfona. Capturing fuzziness and uncertainty of spatiotemporal objects. *Lecture Notes in Computer Science*, 2151:112, 2001.

[10] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Querying the Uncertain Position of Moving Objects. In *Temporal Databases: Research and Practice*, volume 1399, pages 310–337. Springer-Verlag, 1998.

[11] P. Sistla, Wolfson, Chamberlain, and S. Dao. Querying the uncertain position of moving objects. *Lecture Notes in Computer Science*, 1399:310, 1998.

[12] G. Trajcevski. Probabilistic range queries in moving objects databases with uncertainty. In *Proc. of the 3rd ACM international workshop on Data engineering for wireless and mobile access*, pages 39–45. ACM Press, 2003.

[13] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain. The geometry of uncertainty in moving objects databases. In *EDBT 2002, Proc. of 8th International Conference on Extending Database Technology*, volume 2287 of *Lecture Notes in Computer Science*, pages 233–250. Springer, 2002.

[14] E. W. Weisstein. Affine transformation. *MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com.*

[15] E. W. Weisstein. Bijection. *MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com.*

[16] E. W. Weisstein. Law of cosines. *MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com.*

[17] O. Wolfson, L. Jiang, A. P. Sistla, S. Chamberlain, N. Rishe, and M. Deng. Databases for tracking mobile units in real time. In C. Beeri and P. Buneman, editors, *ICDT*, volume 1540 of *Lecture Notes in Computer Science*, pages 169–186. Springer, 1999.

[18] 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.

[19] O. Wolfson, A. P. Sistla, B. Xu, J. Zhou, S. Chamberlain, Y. Yesha, and N. Rishe. Tracking moving objects using database technology in DOMINO. In R. Y. Pinter and S. Tsur, editors, *NGITS*, volume 1649 of *Lecture Notes in Computer Science*, pages 112–119. Springer, 1999.