Modeling Moving Objects for Location Based Services

Ouri Wolfson[1], Sam Chamberlain[2], Kostas Kalpakis[3], Yelena Yesha[3]

[1]Department of Electrical Engineering and Computer Science, University of Illinois, Chicago, IL, 60607, USA
wolfson@eecs.uic.edu
[2]Army Research Laboratories, Aberdeen Proving Ground, MD USA
[3]University of Maryland, Baltimore County, 1000 Hilltop Circle Baltimore, MD 21250 USA

Abstract. As prices of basic enabling equipment like smart cell phones, hand holds, wireless modems, and GPS devices continue to drop rapidly, International Data Corp (IDC) predicts that the number of wireless subscribers worldwide will soar to 1.1 billion in 2003. Spurred by the combination of expensive new location-based infrastructure and an enormous market of mobile users, companies will roll out new wireless applications to re-coop their technology investments and increase customer loyalty and switching costs. These applications are collectively called location-based services. In this position paper we outline a novel and comprehensive model for moving objects in location based services. The model is different than other models proposed in the literature in the sense that is captures the spatial, temporal, and uncertain aspects of the location of a moving object. These aspects are captured in the concept of a trajectory. We argue that the existing prevalent model in industry, point location management, has severe drawbacks, and explain why our trajectory model solves these problems. We also outline a set of operators to access the trajectories database.

1. Introduction

In 1996, the Federal Communications Commission (FCC) mandated that all wireless carriers offer a 911 service with the ability to pinpoint the location of callers making emergency requests. This requirement is forcing wireless operators to roll out costly new infrastructure that provides location data about mobile devices. In part to facilitate the rollout of these services, in May 2000, the U.S. government stopped jamming the signals from global positioning system (GPS) satellites for use in civilian applications, dramatically improving the accuracy of GPS-based location data to 50-100 meters.

As prices of basic enabling equipment like smart cell phones, hand holds, wireless modems, and GPS devices and services continue to drop rapidly, International Data Corp (IDC) predicts that the number of wireless subscribers worldwide will soar to 1.1 billion in 2003. Spurred by the combination of expensive new location-based infrastructure and an enormous market of mobile users, companies will roll out new wireless applications to re-coop their technology investments and increase customer

B. König-Ries et al. (Eds.): IMWS 2001, LNCS 2538, pp. 46-58, 2002.

loyalty and switching costs. These applications are collectively called location-based services.

Emerging commercial location-based services fall into one of the following two categories. First, Mobile Resource Management applications that include systems for mobile workforce management, automatic vehicle location, fleet management, logistics, transportation management and support (including air traffic control). These systems use location data combined with route schedules to track and manage service personnel or transportation systems. Call centers and dispatch operators can use these applications to notify customers of accurate arrival times, optimize personnel utilization, handle emergency requests, and adjust for external conditions like weather and traffic. Second, location-aware content delivery services that use location data to tailor the information delivered to the mobile user in order to increase relevancy, for example delivering accurate driving directions, instant coupons to customers nearing a store, or nearest resource information like local restaurants, hospitals, ATM machines, or gas stations. Analyses Ltd. estimates that location based services will generate $18.5B in sales by 2006.

In addition to commercial systems, management of moving objects in location based systems arises in the military (see [C1,C2]), in the context of the digital battlefield. In a military application one would like to ask queries such as "retrieve the helicopters that are scheduled to enter region R within the next 10 minutes".

Due to these developments, location based services have been generating recently a tremendous amount of commercial and scientific interest. In the scientific community the subject is termed Moving Objects Databases and works in this area concentrate on modeling and querying the location of these objects (see for example [SWCD, GBE+]), efficient location updating in the face of uncertainty (e.g. [PJ, WSCY]), and indexing (see [TUW, KGT, PTJ, SJLL]).

In this position paper we outline a novel and comprehensive model for moving objects in location based services. The model is different than other models proposed in the literature in the sense that is captures the spatial, temporal, and uncertain aspects of the location of a moving object. These aspects are captured in the concept of a trajectory. We argue that the existing prevalent model in industry, point location management, has severe drawbacks, and explain why our trajectory model solves these problems. We also outline a set of operators to access the trajectories database.

The rest of the paper is organized as follows. In the next section we briefly discuss the vendors of software systems for location based services. In section 3 we explain the point location management and discuss its limitations. In section 4 we introduce the trajectory model, and in section 5 we introduce the spatio-temporal access operators. In section 6 we conclude.

2. Location Based Services Industry

In general, current providers of location based services systems fall into one of four categories. The first is vendors of location-based software to the wireless service providers offering technology aimed at the consumer market. The leader in this arena is Signalsoft (*www.signalsoftcorp.com*) whose software runs on proprietary equipment, mainly telecommunication switches. Second are the software vendors for

mobile workforce management, like eDispatch and American Mobile Satellite Corp. (www.MobilePosition.com) who concentrate on electronic forms and work-orders, and communication between mobile devices and corporate databases. The third group is Geographic Information Systems vendors such as ESRI Corp. *(www.esri.com*) and MapInfo *(www.mapinfo.com)*, typically offering single-user systems that lack the real-time capabilities required to manage moving objects. Finally, there are vendors to the transportation industry such as Kinetic Computer Co. *(www.kin*.com), TMW systems and @Road (www.atroad.com). These systems provide fleet management capabilities, including tracking, routing, scheduling, dispatching, and billing.

3. Point Location Management and Its Drawbacks

A fundamental capability embedded in all the software systems that provide location based services is the management of location information, particularly the location of devices such as cell phones, personal digital assistants, laptops, etc. These devices are carried by people, or mounted on moving objects such as vehicles, aircraft, or vessels. The location information is updated by positioning technologies. Examples of such technologies include 1) GPS (that is transmitted from the device to the location server via a wireless network), 2) network based positioning that computes the location by triangulation of transmission towers, 3) fixed sensors in the environment (e.g. at a toll booth) that identify the moving object, and 4) cell-id that identifies the cell in which the moving object is located (a low resolution method). The management of location information involves tracking the moving object, modeling its location (i.e. representing it in a server), and retrieving it by means of queries and triggers (which are also called alerts).

Existing industrial systems perform location management as follows. For each moving object, a time-location point of the form *(x, y, t)* is generated periodically, e.g. every *m* minutes, indicating that the object is at location with coordinates *(x, y)* at time *t*. The point is stored in a database managed by a Database Management System (DBMS) such as ORACLE or INFORMIX. SQL is used to retrieve the location information. If the exact current location of a moving object is required at a particular time, then the object is contacted, and its location is retrieved.

This method is called *point-location management*, and it has several critical drawbacks. First, the method does not enable interpolation or extrapolation. For example, assume that a user needs to know which police officers were within one mile from the location of an emergency that occurred at 3pm. This information can only be retrieved for the moving objects that happened to generate a location update at 3pm. If an object did not generate an update at 3pm, then its whereabouts at that time are unknown. Now assume that the query is issued at 3pm. Then all the objects are polled for their location (resulting in a bandwidth utilization spike); and furthermore, objects that are disconnected from the network are not identified even if they are at the right location. The problem is even more severe for extrapolation, i.e. if a future location is requested; for example, which field service employees will be closest to a customer location at 5pm? This query cannot be answered by the point-location method.

The second problem of the point-location method is that it leads to a critical precision/resource trade-off. An accurate picture of the precise location of moving objects would require frequent location updates that consume precious resources such as bandwidth and processing power.

Finally, a third problem of this method is that it leads to cumbersome and inefficient software development. Specifically, location based services will require the development of a vast array of new software applications. Doing so on top of existing DBMS technology has several drawbacks. First, existing DBMS's are not well equipped to handle continuously changing data, such as the location of moving objects. The reason for this is that in databases, data is assumed to be constant unless it is explicitly modified. For example, if the salary field is 30K, then this salary is assumed to hold (i.e. 30K is returned in response to queries) until explicitly updated. The second drawback is that location based services applications need to manage space and time information, whereas SQL is not designed and optimized for this type of when/where queries and triggers. For example, the query "retrieve the vehicles that are inside region R *always* between 4pm and 5pm" would be very difficult to express in SQL. Finally, the location of a moving object is inherently imprecise because the database location of the object (i.e. the object-location stored in the database) cannot always be identical to the actual location of the object. This inherent uncertainty has various implications for database modeling, querying, and indexing. For example, there can be two different kinds of answers to queries, i.e. the set of objects that "may" satisfy the query, and the set that "must" satisfy the query. SQL semantics cannot account for this difference. Furthermore, even if existing systems were to use interpolation, they wouldn't be able to calculate or bound the uncertainty; namely, was the vehicle within 100 inches, 100 feet, 100 yards of an interpolated point calculated.
An interesting observation is that the point location management is used for two different cases. One in which the route of the moving object is known a priori (e.g. trucking fleet management, municipal transit), and the other in which such information is not available, as in location-aware content delivery. In other words, the information available a priori is not utilized for tracking, and it is not updated as a result of tracking.

4. Trajectory Location Management

In this section we outline our proposed model of a trajectory, explain how to construct it, and explain how it solves the problems associated with point location management. In principle these problems are addressed by making use of a priori or inferred information about the destination of an object. For example, the destination can be inferred based on a motion pattern (e.g. the person travels to the office between 8am and 9am), or access to auxiliary information (e.g. a calendar may indicate a meeting at given time and address).

The method proposed is called *trajectory location management.* In this method we first obtain or estimate the source and destination of the moving object. For example, the object starts in New York City at 57th street at 8th Ave. at 7am and heads for Chicago at the intersection of Oak and State streets. Then, by using an electronic map

geocoded with distance and travel-time information for every road section, a trajectory is constructed.

Before defining the trajectory, let us define the format of an electronic map. An *electronic map* is a relation. Each tuple in the relation represents a city block, i.e. the road section in between 2 intersections, with the following attributes:

- *Polyline*: the block polyline given by a sequence of 2D *x,y* coordinates: $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$. Usually the block is a straight line segment, i.e. given by two *(x,y)* coordinates.
- *Fid*: The block id number

The following attributes are used for geocoding, i.e. translating between an *(x,y)* coordinate and an address such as "1030 North State St.":

- *L_f_add*: Left side from street number
- *L_t_add*: Left side to street number
- *R_f_add*: Right side from street number
- *R_t_add*: Right side to street number
- *Name*: street name
- *Type*: ST or AVE
- *Zipl*: Left side Zip code
- *Zipr*: Right side Zip code
- *Speed*: speed limit on this city block
- *One way*: a Boolean One way flag.

The following attributes are used for computing travel-time and travel-distance.

- *Meters*: length of the block in meters
- *Drive Time*: typical drive time from one end of the block to the other, in minutes

Such maps are provided by, among others, Geographic Data Technology Co. (www.geographic.com) An intersection of two streets is the endpoint of the four block-polylines. Thus each map is an undirected graph, with the tuples representing edges of the graph.

The route of a moving object *O* is specified by giving the starting address or (*x,y*) coordinate *(start_point)*, the starting time, and the ending address or (*x,y*) coordinate *(end_point*). An external routine available in most existing Geographic Information Systems, and which we assume is given a priori, computes the shortest cost (distance or travel-time) path in the map graph. This path denoted *P(O)* is given as a sequence of blocks (edges), i.e. tuples of the map. Since *P(O)* is a path in the map graph, the endpoint of one block polyline is the beginning point of the next block polyline. Thus the whole *route* represented by *P(O)* is a polyline denoted *L(O)*. For the purpose of processing spatiotemporal range queries, the only relevant attributes of the tuples in *P(O)* are Polyline and Drive-Time[1].

Given that the trip has a starting time, for each straight line segment on *L(O),* we can compute the time at which the object *O* will arrive to the point at the beginning

[1] Experiments that we conducted using a map of the Chicago metropolitan area indicate that routes of average length 6.5 miles have on average 51 line segments, routes of average length 17.06 miles have on average 122 line segments, routes of average length 22.56 miles have on average 109 line segments, and routes of average length 36.4 miles have on average 147 line segments.

of the segment (using the *Drive-Time* attribute). This is the *certain-trajectory*, or *c-trajectory*. Intuitively, the *c-trajectory* gives the route of a moving object, along with the time at which the object will be at each point on the route. More formally, a *c-trajectory* is a sequence of straight-line segments $(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n)$ in 3-dimensional space. The *c-trajectory* means that when the object starts at a location having coordinates (x_1, y_1) at time t_1, it will move on a straight line at constant speed and will reach location (x_2, y_2) at time t_2, and then it will move on a straight line at constant speed and will reach location (x_3, y_3) at time t_3, etc. The *c-trajectory* is an approximation of the expected motion of the object in space and time. The reason it is only an approximation is that the object does not move in straight lines at constant speed. However, given enough straight lines, the approximation can be accurate up to an arbitrary precision. The number of line segments on the trajectory has an important implication on the performance and precision of queries and triggers. Specifically, the performance increases and the precision decrease as the number of line segments decreases. We adjust and fine-tune the number of line segments on each trajectory by using a method that has been studied in computer graphics, namely *line simplification* [DP, AV].

The *c-trajectory* is stored in the server database and in a computer on board the moving object.. At any point in time t between t_i and t_{i+1} the server can compute the expected location of the moving object at time t. Observe that this technique solves the first problem associated with point location management. Namely, trajectory location management enables both, location interpolation and extrapolation. The server can compute the expected location of the moving object at any point in time between the start and end times of the trip. For example, if it is known that the object is at location (x_5, y_5) at 5pm and at location (x_6, y_6) at 6pm, and it moves in a straight line at constant speed between the two locations, then the location at 5:16pm can be computed at anytime, i.e. before 5:16 (extrapolation) or after (interpolation).

Finally, the trajectory (or the uncertain trajectory) is obtained by associating an uncertainty threshold u_i with the i^{th} line segment on the c-trajectory. The line segment together with the uncertainty threshold constitute an "agreement" between the moving object and the server. The agreement specifies the following. The moving object will update the server if and only if it deviates from its expected location according to the trajectory by u_i or more. How does the moving object compute the deviation at any point in time? Its computer receives a GPS update every 2 seconds, so it knows its actual location at any point in time. It has the trajectory, so by interpolation it can compute its expected location at any point in time. The deviation is simply the distance between the actual and the expected location. More formally, a *trajectory* is polyline $(x_1, y_1, t_1, u_1), (x_2, y_2, t_2, u_2), \ldots, (x_n, y_n, t_n, u_n)$ in 4-dimensional space.

Each uncertainty threshold u_i is obtained by various methods such as the maximum error-tolerance in queries, or by a cost optimization function that we developed; the purpose of the function is to optimize the combined cost of bandwidth and imprecision. The cost function takes into consideration factors such as the behavior of the past deviation, the message cost, the number of expected queries, the probability of disconnection, etc. The moving object may change the uncertainty threshold at each location update. However, business considerations often determine that there is a maximum uncertainty that the uncertainty threshold cannot exceed. For example, it may be required that the location of the repair crew is known with an

uncertainty that does not exceed 3 miles. A lower uncertainty would be desirable if the extra communication cost justifies it; the computation of the optimal uncertainty is done using the cost function that we developed (see [WSCY]).

At the server, the trajectory is maintained by revising it according to location-updates from the moving object, and according to real-time traffic conditions obtained from traffic web sites. We have developed a traffic incident model, and a method of identifying the trajectories affected by a traffic incident. Observe that determining whether or not a trajectory is affected by a traffic incident is not a simple matter, and it requires prediction capabilities. For example, suppose that according to the current information Joe's van is scheduled to pass through highway section x twenty minutes from now, and suppose that a web site currently reports a traffic jam on highway section x. Will Joe's expected arrival time at his destination be affected by this? Clearly it depends on whether or not the jam will clear by the time Joe arrives at highway section x. We use historical information and a novel traffic model to make this prediction.

Observe that the agreement (namely the trajectory plus the uncertainty threshold) between the moving object and the server solves the second problem of point location management. Namely, the tradeoff between resource/bandwidth consumption and precision has been broken. In trajectory location management the location of a moving object can be computed with a high degree of precision, using a small number of location updates, or no updates at all. In particular, if the moving object is "on schedule", i.e., it does not deviate from its prescribed trajectory by more than the uncertainty threshold, then no resources are consumed for updates.

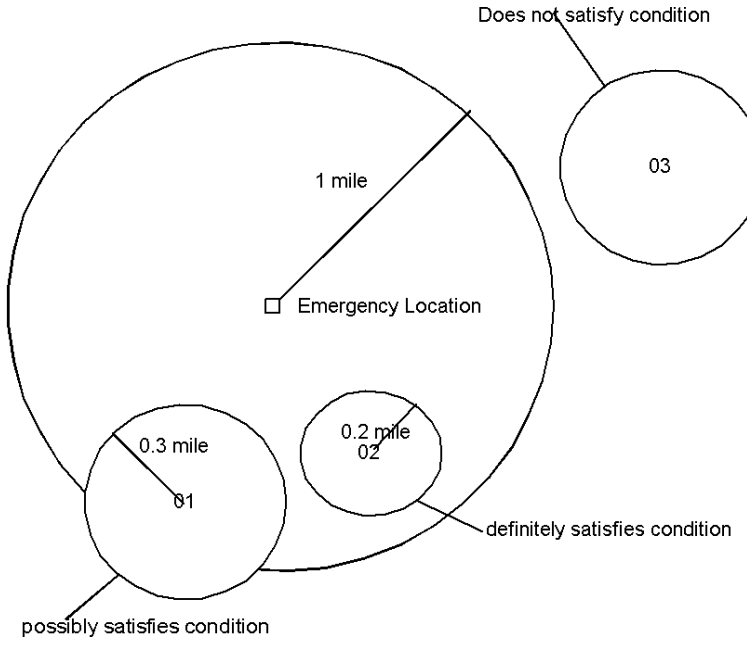

Fig. 1. Illustration of uncertainty quantifiers

Now assume that a user needs to know which police officers are currently within one mile from the location of an emergency. Remember that in the point location management all the objects are polled for their location (resulting in a bandwidth utilization spike); and furthermore, objects that are disconnected from the network are not identified even if they are at the right location. Do objects need to be polled in the trajectory location management method? The answer is that some may need to be, but the number of polled objects is much smaller than in point location management. Since in the trajectory location management method there is an uncertainty associated with the location of each police car, the dispatcher would ask: which objects *may* be within one mile of the location? The retrieved set of objects will be divided into two parts, the objects that are *definitely* within one mile, and the objects that are *possibly* within mile. For example, consider an object whose expected location is within 0.8 miles of the emergency. If its uncertainty threshold is 0.2 miles, it definitely satisfies the condition. If its uncertainty threshold is 0.3 miles, then it possibly, but not necessarily, satisfies the condition (see Fig.1). Only the objects that possibly, but not necessarily, satisfy the condition will have to be polled.

Finally, let us observe that a trajectory can be constructed based on past motion in which an object used the point location management. Namely, the trajectory can be constructed from a set of 3D points $(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n)$ that were transmitted by a moving object using the point location management method. One can simply connect the points along the shortest path on the map, and then associate an uncertainty u_i with line segment i. The uncertainty u_i can be bounded given the maximum speed of the object and the known times of the two GPS points immediately preceding and succeeding the i^{th} line segment (see [PJ]).

5. Data Access Operators

Finally, the third problem associated with point location management is solved by a novel set of operators by which the database is accessed. The operators are used to query the database, and also to set triggers (or alerts) that are fired when interesting conditions are satisfied by the database (e.g. an object is expected to be late by more than one hour). The operators are designed to express when/where questions in an uncertain environment. They can be in cooperated into the traditional SQL query language that has been widely adopted by commercial database systems. This means that one can ask queries and set triggers that combine the traditional database conditions with the new operators. This means, for example, that a dispatcher can ask a query such as: retrieve the service-personnel who have Qualification="dsl", and will be within 1 mile of 851 S. Morgan St. at 5pm. This also means that the operators can be combined using boolean operators such as *and* and *or*. An additional implication is that these operators/queries can be entered by a user on a client computer, and the same set of operators can be invoked from a program. The latter option enables development of complex spatial and temporal applications. Obviously the proposed set of operators is not exhaustive, but each operator in the set can on one hand be implemented efficiently, and on the other hand we believe that it is useful in a large

class of applications. Although the operators are given here in textual form, they can be implemented in point-and-click, drag-and-drop, graphical and visual form.

The new operators are divided into three classes, operators that pertain to a single trajectory, operators that pertain to the relationship of trajectories to fixed-location facilities or regions, and the relationship among multiple trajectories. These loosely correspond to point queries, range queries, and join queries, respectively, in traditional databases (see [GUW]). Each one of the immediately following subsections discusses one of these classes. The last subsection touches on the usage of these operators in triggers and continuous queries.

5.1 Operators That Analyze a Single Trajectory

- *WHERE is object o AT {time t | Currently [Max Uncertainty u]}*. The operator returns the location of the object *o* currently, or at an arbitrary time *t*. If the current location of the object is requested, then the maximum tolerable uncertainty *u* can also be specified. In this case, if the uncertainty associated with the expected location of object *o* is higher than *u*, then the server contacts the object wirelessly to get its exact current location. As a usage example for the WHERE operator, consider a dispatcher that needs to assign a job to a mobile service employee; The job needs to be done at time t, and the dispatcher needs to know where the technician is expected to be at that time according to the current schedule.
- *WHEN object o CLOSEST TO address x*. The operator returns a list of times at which the object passes by or stops at address *x*. Observe that there may be a list of times, since the object may visit or pass by the same location more than once. If the object never passes by or visits *x*, then the operator returns the time when the object passes by the closest location to *x* on its trajectory. This operator is used, for example, when the customer at *x* needs to know when the technician will arrive at her location according to the current schedule.
- *SECTIONS [partition] [within- x] [closest] object o, FACILITIES*. Assume that FACILITIES is a set of n point-coordinates on a map. The construct identifies on the trajectory of *o* a set of *n* segments or points. For ***partition,*** the trajectory is partitioned into *i* segments, such that during segment *i* the object *o* is closest to facility *i* than to any other facility. For ***within-x,*** a set of segments is identified on the trajectory such that during segment i the object is within x meters or drive time of facility *i*. For ***closest,*** the operator identifies (at most) *i* points on the trajectory, such that point *i* is closest to facility *i* than any other point on the trajectory. Usage: 1) Suppose that the trajectory describes the motion of a person at a trade show, and the facilities represent exhibitor booths. This operator (within-x variant) enables one to find out how long the person spent at each booth. 2) If the facilities are the set of motels along a route, then this operator (closest variant) identifies the points when the driver is closest to each motel.
- *VCR object o*. The operator "replays" the trajectory of object *o*. The replay can be done on a certain time-scale (e.g. a minute per second), and it can fast forward/rewind to a certain point in time.

5.2 Operators for Identifying Trajectories That Stand in Certain Relationships to a Region or a Facility

Each one of the following operators is a condition. The condition is satisfied by the objects that stand in a certain relationship (e.g. within distance *x*) to a fixed facility (i.e. a point on a map) or a region *R*, during *T*. Thus the conditions correspond to a spatio-temporal range query. Why then is there more than a single operator? The answer is threefold. First, since the expected location of an object changes continuously, one may ask for the objects that satisfy the condition *always* within *T*, or the objects that satisfy it *sometime* during *T*; similarly one can ask for the objects that satisfy the condition *somewhere* in R or *everywhere* in R. Second, there is an uncertainty associated with the location, and thus one can ask for the objects that *possibly* satisfy the condition, or the ones that *definitely* do so. Third, it turns out that the order in which the temporal quantifier is combined with the certainty quantifier important. The (abbreviated) operators are:

- *Possibly-Within* [distance *d* | travel-time *t*] from *R*, Sometime in the time interval *T*. The condition is satisfied by the objects which are at distance at-most *d* or travel time at-most *t* from R, sometime in the time interval *T*. The time interval *T* may indicate Currently. This operator is used, for example, when a dispatcher needs to assign a job at *R* to a technician. The job needs to be done in the time-interval *T*. The dispatcher needs to know which technicians are expected to be within distance *d* or travel-time *t* from *R* within the time interval *T*. Since there is an uncertainty associated with the location at each object at each point in time, the condition retrieves the objects that are *possibly* Within-Sometime.
- *Definitely-Within* [distance *d* | travel-time *t*] from *R*, Sometime in the time interval *T*. The difference between this operator and the previous one is the certainty quantifier. This operator retrieves the objects that are *definitely-Within Sometime*. The dispatcher may use this operator when many technicians are *possibly-Within Sometime*. To narrow the search s/he is interested in the ones that, according to the current schedule, definitely satisfy the condition.
- *Possibly-Within* [distance d | travel-time t] from R, Always in the time interval *T*. The difference between this operator and operator 1 (*possibly-sometime*) is the temporal quantifier. Here we request the objects that are *Possibly-Within always* during *T*. For example, the police dispatcher may need to know which patrol cars will stay in the region R for a whole duration *T*.
- *Definitely-Within* [distance *d* | travel-time *t*] from *R*, Always in the time interval *T*. Combination of operators 2 and 3. Used when the police dispatcher needs to know which patrol cars will definitely stay in the region *R* for a whole duration *T* (according to the current schedule).
- *Possibly-During T Everywhere in R*. Which police cars will possibly cover the whole region *R* during *T*?
- *Definitely-During T Everywhere in R*. Which police cars will definitely cover the whole region *R* during *T*
- *Possibly-Closest-to (or Farthest from) R Sometime in a time interval T*. Which technician is possibly closest to location *R* sometime during *T*? Note that due to the uncertainty and due to the fact that *T* is a time interval, there may be more

than a single technician that may be closest to *R* during. To find out if there is a technician that will definitely be closest, one would use the next operator.

- *Definitely-Closest-to (or farthest from) R Sometime in a time interval T.* The operator identifies objects *o1, o2*, .., such that *o1* is definitely closest between *t1* and *t2, o2* is definitely closest between t_3 and t_4, etc. Observe that due to the location uncertainty there may be time intervals within T for which no single moving object will definitely be closest.
- *Possibly-Closest-to (or Farthest from) R Always in a time interval T.* Which technician is possibly closest to location *R* always during *T*?
- *Definitely-Closest-to (or Farthest from) R Always in a time interval T.* Same as the previous operator, except that we are looking for the moving objects that definitely satisfy the condition.

Some other operators are *Always Possibly-Within, Sometime Definitely-Within*, *During Possibly-Everywhere, During Somewhere-Definitely, During Definitely-Somewhere*.

Each one of the operators in this section can be used in one of two variations: *Along Existing Route* (AER), or *Along Shortest Route* (ASR). Consider for example the first condition, Possibly-Within-Sometime. The condition is satisfied with the AER variant for object o, if R is on *o*'s route, and *o* is within distance *d* from R while traveling along its predefined route. However, a police dispatcher may be interested in the patrol cars that can reach an emergency destination while traveling along the shortest path from their current location to the destination. In other words, in the ASR variant the patrol cars are allowed to abandon their predefined route in order to reach the destination.

Each one of the operators in this subsection can be combined with operators in the previous one. For example, to find the current location of the object that will be closest to R at 5pm one would write:
WHERE is object o Currently
For o Definitely closest-to R at 5pm

5.3 Operators for Identifying Relationships between Trajectories

Each one of the relationship-to-facilities operators can be applied as a relationship-between-trajectories operator. These are called *join* operators. For example:

- *Possibly-Within [distance d | travel-time t], Sometime in the time interval T.* The condition is satisfied by the pairs of trajectories which are within distance *d* or travel time *t* from each other, sometime in the time interval *T*. This operator is used, for example, in an air-traffic-control system that stores the trajectories of planes. We assume that, in contrast to the existing system in which planes fly on "highways in the sky", the new free-flight system has been implemented (see www.faa.gov/freeflight) . The air traffic controller needs to know which planes are expected to be within distance d from each other, thus representing a safety hazard.

The opposite operator also applies. Specifically:

- *Possibly-Fartherthan [distance d | travel-time t], Sometime in the time interval T.* The condition is satisfied by the pairs of trajectories which are

farther than distance *d* or travel time *t* from each other, sometime in the time interval *T*. This operator is used, for example, in a military situation in which a database represents vehicles/aircraft moving as a unit. This operator enables the general to know if a vehicle moves too far away from the rest of the company.

Some single-trajectory-analysis operators can also be applied to pairs of trajectories. For example:
WHEN trajectory *i* and trajectory *j* are CLOSEST. The operator returns a list of times at which the two trajectories come closest to each other.

5.4 Triggers and Continuous Queries

Each one of the above operators can be applied as a trigger. In this capacity an alert message is sent when the condition of the operator is satisfied. For example, consider the SECTIONS operator, and suppose that the facilities are motels. When used as a trigger, this operator will alert the driver every time he is closest to, or within x minutes drive-time from a motel. Or, consider the air traffic control application mentioned in the previous subsection. The Possibly-Within-Distance operator, when used as a trigger, will send an alert message when the trajectories of two airplanes are too close to each other.

Triggers are implemented using a mechanism called continuous queries, which have a merit in their own right. Continuous queries are queries that conceptually execute "continuously" (although obviously this would be prohibitive from a performance point of view, and we have developed mechanisms to avoid this). Each one of the above operators can be applied as a continuous query. In this capacity the query executes continuously, and the resulting set of retrieved objects changes as the database changes. Consider for example a visualization application, in which the user "flies over" a terrain (e.g. a military battlefield), and the database displays objects that are visible from every location traversed. In such an application the system issues a continuous query in which the objects within a certain range (depending on the altitude of the flight) from the current location are being retrieved from the database and displayed continuously.

6. Conclusion

Miniaturization of computing devices and wireless networks are propagating computing from the stationary desktop to the mobile outdoors. An important attraction of this revolutionary development has recently become known as location based services. These services promise to deliver the right information at the right time and at the right location, and also to enable accurate management of mobile resources. In this paper we focused on modeling issues for location based services. We proposed the trajectory as a four dimensional object that captures the essential aspects of the moving object location. These aspects are two-dimensional space, time, and uncertainty (the concepts carry over in a straight forward way to 3D space). This model addresses the deficiencies of the point location management model that is

currently prevalent in industry because it enables interpolation, extrapolation, breaking the accuracy-resource tradeoff, and bounding uncertainty. We also proposed a set of operators to access a database of trajectories. We classified these operators into three categories. First, operators that analyze a single trajectory (analogous to point queries), second operators that retrieve trajectories in a spatio-temporal range, and third join operators that retrieve pairs of trajectories. Finally we explained that, in addition to instantaneous queries, these operators can be used in triggers and in continuous queries. In future work we intend to focus on efficient processing of the operators, and then consider an environment in which the trajectories database is distributed or mobile. We will also consider methods of mining of the trajectories database, privacy and security issues, and traffic modeling and prediction. We believe that the pervasive, context aware computing era has just begun, and location based services are only one of the first manifestations of this revolution.

References

[WSCY] O. Wolfson, A. P. Sistla, S. Chamberlain, Yelena Yesha. Updating and Querying Databases that Track Mobile Units. *Distributed and Parallel Databases*, 7, 257-287, 1999

[PJ] D. Pfoser, C.S. Jensen: "Capturing the uncertainty of moving objects representations". *Proc. of the 12 Intl. Conf. on Scientific and Statistical Database Management*, 2000. IEEE Computer Society.

[PTJ] D. Pfoser, Y. Theodoridis, and C. Jensen. Indexing trajectories of moving point objects. Technical Report 99/07/03, Dept. of Computer Science, University of Aalborg, 1999

[SJLL] S. Saltenis, C.S. Jensen, S.T. Leutenegger, M.A. Lopez: "Indexing the Positions of Continuously Moving Objects. *ACM SIGMOD Conference 2000*.

[JUW] J. Jayeb, O. Ulusoy, and O Wolfson. A Quadtree - based dynamic attribute indexing method. *The computer Journal*, (41(3), 1998

[DP] D.H. Douglas and T. K Peucker Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canad, Cartog*. 10(2):112-122, Dec. 1973

[C1] S. Chamberlain. Automated information distribution in bandwidth-constrained environments. *MILCON-94 conference*, 1994

[C2] S. Chamberlain. Model-based battle command: A paradigm whose time has come. *1995 Symposium on C2 Research & Technology, NDU*, June 1995

[SWCD] A. P. Sistla, O. Wolfson, S Chamberlain, and S. Dao: Modeling and Querying Moving Objects, In *Proc. of the International Conference on Data Engineering* (1997) pp. 422-432.

[GUW] H. Garcia-Molina, J. D. Ullman, J. Widom. *Database System Implementation*, Prentice Hall, Upper Saddle River, NJ.

[GBE+] Ralf Hartmut Guting, Michael H. Bohlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos. Markus Schneider, and Michalis Vazirgiannis.: A Foundation for Representing and Querying Moving Objects, in *ACM-Transactions on Database Systems Journal* (2000), 25(1), 1-42

[KGT] G. Kollios, D. Gunopulos, V.J. Tsotras: "On indexing moving objects". *ACM PODS 1999 conference*. ACM Press.

[AV] P.K. Agarwal and K.R. Varadarajan. Efficient Algorithms for Approximating Polygonal Chains. *Discrete Comput. Geom.*, 23:273-291(2000)