Techniques for Efficient Road-Network-Based Tracking of Moving Objects

Alminas Čivilis, Christian S. Jensen, *Senior Member*, *IEEE*, and Stardas Pakalnis

Abstract—With the continued advances in wireless communications, geo-positioning, and consumer electronics, an infrastructure is emerging that enables location-based services that rely on the tracking of the continuously changing positions of entire populations of service users, termed moving objects. This scenario is characterized by large volumes of updates, for which reason location update technologies become important. A setting is assumed in which a central database stores a representation of each moving object's current position. This position is to be maintained so that it deviates from the user's real position by at most a given threshold. To do so, each moving object stores locally the central representation of its position. Then, an object updates the database whenever the deviation between its actual position (as obtained from a GPS device) and the database position exceeds the threshold. The main issue considered is how to represent the location of a moving object in a database so that tracking can be done with as few updates as possible. The paper proposes to use the road network within which the objects are assumed to move for predicting their future positions. The paper presents algorithms that modify an initial road-network representation, so that it works better as a basis for predicting an object's position; it proposes to use known movement patterns of the object, in the form of routes; and, it proposes to use acceleration profiles together with the routes. Using real GPS-data and a corresponding real road network, the paper offers empirical evaluations and comparisons that include three existing approaches and all the proposed approaches.

Index Terms—Database management, distributed databases, query processing, temporal databases.

1 Introduction

In step with the emergence of an infrastructure for mobile, online location-based services (LBSs) for general consumers, such services are attracting increasing attention in industry and academia.

An LBS is a service that provides location-based information to mobile users. The main idea is to provide the service user with a service that is dependent on positional information associated with the user, most importantly, the user's current location. The service may also be dependent on other factors, such as personal preferences and interests of the user [3].

Examples of LBSs abound. A service might inform its users about traffic jams and weather situations that are expected to be of relevance to each user. A friend monitor may inform each user about the current whereabouts of friends. Other services may track the positions of emergency vehicles, police cars, security personnel, hazardous materials, or public transport. A more advanced location-based "catch the monster" game may allow a group of users to work together to surround and catch a virtual, but geo-positioned, monster.

Services such as these rely to varying degrees on the tracking of the geographical positions of moving objects. For example, traffic jams may be identified by monitoring the movements of service users; and, the users that should receive specific traffic-jam or weather information are identified by tracking the users' positions. Some services require only fairly inaccurate tracking, e.g., the weather service, while other services require much more accurate tracking, e.g., location-based games.

We assume that users have wireless devices (e.g., mobile phones) that are online via some form of wireless communication network. We also assume that the positions of the users are available. Specifically, we rely on the Global Positioning System for positioning. To accomplish tracking with a certain accuracy, each wireless device monitors its real position (its GPS position) and compares this with a local copy of the position that the central database assumes. When needed in order to maintain the required accuracy in the database, the wireless device issues an update to the server. The database may predict the future positions of a device in different ways. In the general case, the database explicitly informs the mobile device about how it predicts the client's position. The challenge is then how to represent, and predict, the future positions of a mobile device in the database so that the number of updates is minimized. Reduction of updates reduces communication and server-side update processing.

A detailed coverage of related work is given in Section 6. In short, to the best of our knowledge the techniques for update reduction proposed in this paper have not been proposed or evaluated in past work. We share the general setting with Wolfson et al. [16], [18], and our proposals take the segment-based technique described by Čivilis et al. [4], which is similar to a technique presented by Wolfson and Yin [18], as the starting point.

Section 2 describes the segment-based approach in some detail. In this approach, the future movement of a mobile device, termed a moving object, is represented by a road segment drawn from the underlying road network and a fixed speed. A road segment is a polyline, i.e., a sequence of

- *A. Čivilis is with the Department of Computer Science II, Vilnius University, 24 Naugarduko Street, Vilnius LT-03225, Lithuania. E-mail: alminas.civilis@maf.vu.it.*
- *C.S. Jensen and S. Pakalnis are with the Department of Computer Science, Aalborg University, Fredrik Bajers Vej 7E, DK-9220 Aalborg Øst, Denmark. E-mail: {csj, stardas}@cs.aau.dk.*

Manuscript received 28 Feb. 2004; revised 5 Aug. 2004; accepted 2 Dec. 2004; published online 17 Mar. 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0060-0204.

connected line segments. So, this representation assumes that a moving object moves on a known road segment with constant speed.

As explained above, a moving object is aware of the server-side representation of its movement. The server uses the presentation for predicting the current position of the moving object. The client-side moving object uses the representation for ensuring that the server's predicted position is within the predefined accuracy.

This paper presents techniques that aim to improve the basic segment-based approach. We are basing our proposals on the segment-based approach because we find this to be the most promising outset for more advanced tracking techniques. As an added benefit, by relating moving objects to the underlying road network, we gain easy access to content that is connected to the road network. Such content may be useful in many LBSs. The paper presents the following techniques that improve the segment-based approach:

- Modification of the road network. The number of updates turns out to be closely related to the segmentation of the road network. We present techniques for modification of a road network with the purpose of finding appropriate segmentations of a road network.
- Use of anticipated routes for the moving objects. Using routes in place of segments allows us to reduce the number of updates caused by changes of segments. Routes are represented as (long) polylines.
- Introduction of acceleration profiles. The basic approach assumes that moving objects are moving at constant speed in-between updates. In order to reduce the number of updates caused by the speed variations of the moving objects, we introduce more accurate speed modeling.

In summary, the paper's main contributions are 1) proposals for three types of techniques that aim to reduce the communication and update costs associated with the tracking of moving objects with accuracy guarantees and 2) empirical evaluations of the best existing tracking techniques and the new techniques based on real data.

The paper proceeds as follows: Section 2 summarizes previously proposed update policies and offers motivation for further investigation of the segment-based approach. Section 3 covers improvements of segment-based approach using road-network modifications. Sections 4 and 5 present the techniques for update reduction using routes and acceleration profiles, respectively. Section 6 offers an overview of related work. The final section summarizes, provides concluding remarks, and offers suggestions for future research.

2 Background

In this section, we first describe the general tracking scenario that we will use. A description of the position data used for tracking follows. Then, we describe the existing tracking approaches, including the segment-based approach. Finally, we compare the approaches and motivate the paper's direction.

2.1 Tracking Scenario

We assume that moving objects are constrained by a road network and that they are capable of obtaining their positions from an associated GPS receiver. Moving objects, also termed clients, send their location information to a central database, also termed the server, via a wireless communication network. We assume that disconnects between client and server are dealt with by other mechanisms in the network than the tracking policies we consider. When a disconnect occurs, these mechanisms notify the server, which may then take appropriate action.

After each update from a moving object, the database informs the moving object of the representation it will use for the object's position. The moving object is then always aware of where the server thinks it is located. The moving object issues an update when the predicted position deviates by some threshold from the real position obtained from the GPS receiver.

Fig. 1 presents a UML activity diagram for the update scenario (activity diagrams model activities that change object states).

The client initially obtains its location information from the GPS receiver. It then establishes a connection with the server and issues an update, sending its GPS information and unique identifier to the server.

Having received this update, the server determines which tracking approach and threshold to use for the client (these are predefined), and it stores the information received from the client in the database. If the tracking approach is the segment-based one, the server also uses map matching to determine on which road segment the client is moving. The server then sends its representation of the client's current and future position to the client.

Having received this information from the server, the client obtains its actual, current location information from the GPS receiver. The client then calculates its predicted position using the representation received from the server, and it compares this to the GPS position. If the difference between these two exceeds the given threshold, the client issues an update to the server. If not, a new comparison is made. This procedure continues until it is terminated by the client. Although the server may also initiate and terminate the tracking, we assume, for simplicity, that the client is in control. This aspect has no impact on the paper's contribution.

2.2 Data Description

As mentioned, GPS is used for positioning of the moving objects. In experiments that will be reported throughout the paper, we use GPS-log data collected during an intelligent speed adaptation project [9]. In this project, GPS receivers and small custom made computers were installed in a number of cars that were driving in the Aalborg area, Denmark. This resulted in the collection of a GPS-log for each car that contains position samples for approximately every second during the periods when the car was being operated during a period of approximately eight weeks.

For our experiments, we also use a digital road network obtained from the same project. The road network is composed of a set of segments, each of which corresponds to some part of the road network that is in-between a pair of

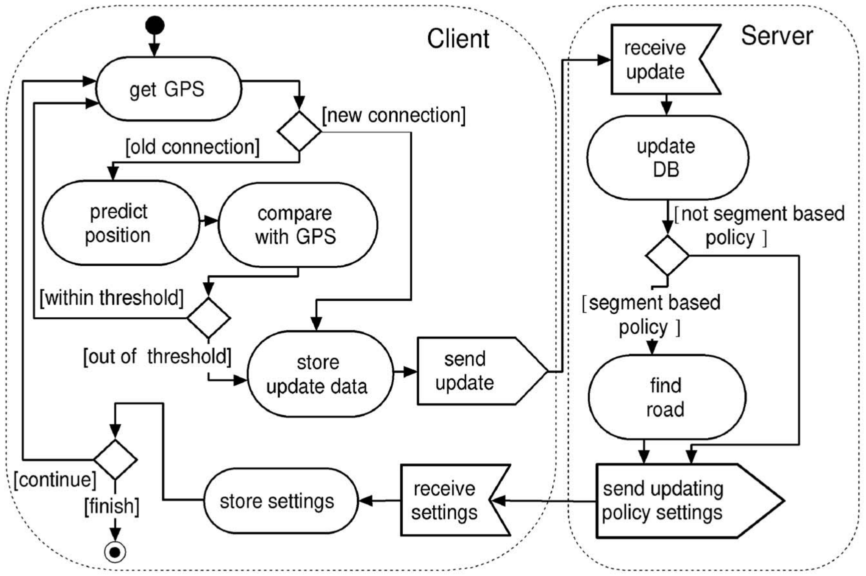

Fig. 1. Tracking scenario diagram.

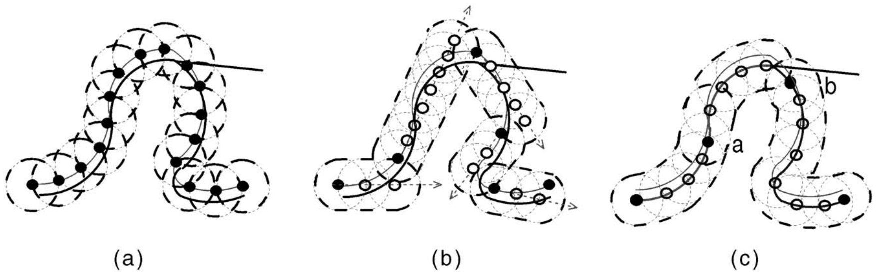

Fig. 2. Tracking policies: (a) Point-based, (b) vector-based, and (c) segment-based.

consecutive intersections or an intersection and a dead end. A segment is defined as a sequence of coordinates, i.e., as a polyline. Further, the road network is partitioned into streets and each segment belongs to precisely one street. Each segment identifies its street by referring to a street code. The top part of Fig. 5 offers a visual image of part of the digital road network.

2.3 Existing Tracking Approaches

We proceed to describe three existing tracking approaches [4]. With minor variations, the first and third of these were previously proposed by Wolfson and Yin [18] (see Section 6 for additional discussion). These techniques follow the scenario described in Section 2.1, but differ in how they predict the future positions of a moving object.

2.3.1 Point-Based Tracking

Using this approach, the server represents a moving object's future positions as the most recently reported position. An update is issued by a moving object when its distance to the previously reported position deviates from its current GPS position by the specified threshold. An example of point tracking is presented in Fig. 2a. Here, the circles indicate the threshold and (solid) points indicate (server-side) predicted positions that result from an update being issued by the object. The two bold lines indicate connected segments of the road network and the thin line represents the actual object movement.

2.3.2 Vector-Based Tracking

In vector tracking, the future positions of a moving object are given by a linear function of time, i.e., by a start position and a velocity vector. Point tracking corresponds to the special case where the velocity vector is the zero-vector.

A GPS receiver computes both speed and heading for the object it is associated with—the velocity vector used in this representation is computed from these two. Using the same notation as the previous figure, Fig. 2b shows also the velocity vectors that are used for prediction. Solid points again indicate predicted positions that result from updates, while the remaining positions are simply predicted.

2.3.3 Segment-Based Tracking

Here, the main idea is to utilize knowledge of the road network in which the clients are moving. A digital representation of the road network is required to be available. The server uses the GPS location information it receives from a client to locate the client within the road network. This is done by means of map matching, which is a technique that positions an object on a road-network segment, at some distance from the start of that segment, based on location information from a GPS device.

In segment-based tracking, the future positions of a client are given by a movement at constant speed along the identified segment, which is represented as a polyline. The speed used is the speed most recently reported by the client. When or if the predicted position reaches the end of its segment, the predicted position remains at the end from then on. In effect, the segment-based tracking switches to point-based tracking.

Segment-based tracking is sensitive to the fidelity of the road network representation used. If for some reason, a matching road segment cannot be found when a moving object issues an update, the segment-based approach switches temporarily to the vector-based approach, which is always applicable. On the next update, the server will again try to find a matching road segment in the database.

Map matching may fail to identify a segment for several reasons. For example, the map available may be inaccurate, or it may not cover the area in which the client is located. The use of vector-based tracking within the segment-based tracking renders segment-based tracking robust.

An example of segment-based tracking is shown in Fig. 2c. Notice that all predicted positions are now located on the road segment, not on the trajectory obtained via the GPS receiver. The example next further explains segment-based tracking.

Example 2.1. Consider a taxi moving in a road network. The taxi starts a trip at 2:00 p.m. It starts at position (x_0, y_0), and it travels at 50 km/h. The threshold is 100 m, i.e., we need to know where the taxi is within 100 m. With point-based tracking, an update is issued when the taxi gets to be more than 100 m away from the previously reported position. This situation is shown in Fig. 2a. Using the vector-based approach, the taxi's movement direction is taken into account. This yields a better approximation of the taxi's movement, thus reducing the number of updates—see Fig. 2b. If we have available a digital representation of the road network in which the taxi is moving, segment-based tracking is possible (see Fig. 2c). Here, the updated point (Fig. 2a) occurs because the taxi slows down so that its predicted position moves ahead of the real position by more than 100 m. The updated point (Fig. 2b) occurs because the taxi reaches the end of a segment so that its predicted position stops, while the taxi keeps moving. The updated point (Fig. 2b) places the taxi on a new road segment.

2.4 Comparison of Update Policies

The tracking approaches described in Section 2.3 were evaluated by Čivilis et al. [4] using the INFATI data described in Section 2.2. Approximately 458,000 GPS positions collected from four cars were used, and thresholds ranging from 40 to 1,000 m were investigated.

Experimental results are presented in Fig. 3. The results were obtained by simulating the scenario described in Section 2.1. Specifically, the movement of each car was simulated using the log of GPS positions for the car. So, a client program and a server program interact, and a simple experiment management system is in charge of the bookkeeping needed in order to obtain the performance results. Instead of obtaining GPS positions from a GPS device in real time, the client program utilizes the GPS log, which, of course, makes the simulation much faster than the reality being simulated. The bookkeeping involves the counting of updates sent from the client program to the server program and keeping track of time.

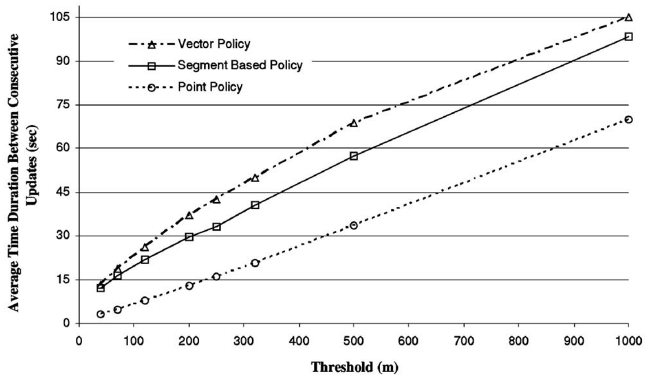

Fig. 3. Comparison of update policies.

All performance studies reported in this paper follow this pattern. The studies differ in the specific GPS data and road networks used, and in the tracking policies used.

In Fig. 3, accuracy threshold values in meters are on the x axis. The client obtains a GPS position from the GPS device every second and performs a comparison between the GPS position and the predicted position. The y axis then gives the average number of seconds in-between consecutive updates sent from the client to the server in order to maintain the required accuracy.

It is seen that the time in-between updates increases as the accuracy threshold increases, i.e., as the required accuracy decreases. The point policy shows the worst performance. Notice that the largest improvement of the segment-based and vector policies over the point policy is for smaller thresholds, while for larger thresholds the improvement is smaller. For thresholds below 200 m, the segment-based and vector policies are more than two times better than the point policy.

We find that segment-based tracking was outperformed because the road segment in the underlying road network were relatively short, having an average length of 174 m. It may be that a relatively straight road is represented by several segments. In this case, vector-based tracking may need less updates. So, although vector-based tracking is simpler and slightly better, we find it likely that it is possible to improve the segment-based tracking to be the best. In addition, segment-based tracking, by relating the location of a moving object to the underlying road network, offers additional advantages:

- Buildings, parking places, traffic jams, points of interest, traffic signs, and other road-related information that is mapped to the road network can be associated with the location of a moving object.
- Road-network-based distances can be used in place of Euclidean distances.
- Acceleration profiles, driver behavior on crossroads, and other road-related data that increase the knowledge about the future positions of moving objects can be exploited.

Consequently, we have chosen to base our proposals for new and more efficient tracking techniques on the segment-based tracking approach.

3 Modification of the Road Network

Recall that with segment-based tracking, the predicted position of an object moves at constant speed along a segment until it reaches the end of the segment, at which time the predicted position remains at the end of the segment. The experimental study reported in the previous section indicates that the numbers of updates in segment-based tracking are closely correlated with the numbers of changes of segments. This motivates modification of the underlying road network representation that may lead to less segment changes.

We proceed to present several road network modifications. The main idea is to connect the road segments in such a way that moving objects would have to change segments as few times as possible as they travel in the road network. We first present a general segment connection algorithm and road network modification approach. Then, three subsequent algorithms are presented that reuse this algorithm. At the end, the effects on tracking of the three algorithms are compared experimentally, and city and suburban driving are compared.

3.1 General Segment Connection Algorithm

The general segment connection algorithm **GSC** captures the overall approach to road network modification.

The idea is to iterate through all segments in the road network to be modified according to some specified ordering. During each iteration, the algorithm thus orders all available segments and then tries to extend the topmost, or current, segment with other segments. To do this, the algorithm identifies all existing segments that start or end at the start or end of the current segment and extends the current segment with the most attractive such segment(s) according to some other specified ordering. A current segment that has been extended is considered in the next iteration, but the segment(s) that were used for the extension are disregarded. A current segment that has not been extended becomes part of the result and is not considered any further.

The algorithm takes four parameters as input. The first is a road network, denoted by rn, which is a set of segments. Each segment is a polyline that represents a small, linear part of the road network. Segments can have connections with other segments only at their start and end points. Further, each segment (initially) belongs to only one street and has one street code assigned to it. Additional detail about the concrete road network used in empirical evaluations in this paper can be found elsewhere [9]. The second parameter of **GSC** is a Boolean valued variable stc that controls the segment connection procedure by allowing or disallowing the connection of polylines with different street codes. The third and fourth parameters, $rnPrioritization$ and $candPrioritization$, are sort order specifications that specify how to sort sets of polylines. By supplying algorithm **GSC** with different parameters, different transformations of a road network result.

Algorithm **GSC** uses function **first**(*set_of_segments, stc, Prioritization*). This function returns the segment in *set_of_segments* that is first according to $Prioritization$, which is a sort order specification. It consists of a list of segment properties, e.g., length, speed limit, number of neighboring segments, along with an indication of whether sorting should be done in ascending or descending order.

A property of a segment can be calculated based on the other segments available in the argument set of segments. An example is the number of segments with which a segment can be extended. When calculating such properties, if stc is set to $true$, segments with street codes that are different from the street code of the current segment are not considered; otherwise, all segments are considered.

A property such as the angle between two spatially connected segments involves two segments. In this case, function **first**(*set_of_segments, stc, Prioritization, pl*) takes an additional parameter: segment *pl*. Then, each segment from *set_of_segments* will have a property "angle" that is equal to the angle between the segment and *pl*.

Algorithm **GSC** is defined next and explained in the following.

```
Algorithm
        GSC($rn, stc, rnPrioritization, candPrioritization$)
1. $cn \leftarrow \emptyset$
2. while $rn \neq \emptyset$ do
3.   $pl \leftarrow \mathbf{first}(rn, stc, rnPrioritization)$
4.   $rn \leftarrow rn \setminus \{pl\}$
5.   $epls \leftarrow \emptyset$
6.   for each $pd \in \{\mathbf{start}(pl), \mathbf{end}(pl)\}$ do
7.       $cand \leftarrow \{plc | plc \in rn \wedge$
                  $(pd = \mathbf{start}(plc) \vee pd = \mathbf{end}(plc)) \wedge$
                  $(plc.streetcode = pl.streetcode \vee \neg stc)\}$
8.      if $cand \neq \emptyset$ then
9.        $pl \leftarrow pl$ extended with
             $\mathbf{first}(cand, stc, candPrioritization, pl)$
10.       $epls \leftarrow epls \cup \{\mathbf{first}(cand, stc, candPrioritization, pl)\}$
11.    if $epls \neq \emptyset$ then $rn \leftarrow (rn \setminus epls) \cup \{pl\}$
12.      else $cn \leftarrow cn \cup \{pl\}$
13. return $cn$
```

A variable cn that will accumulate the result of the algorithm is first initialized. The algorithm then iterates through the polylines of the argument road network in the argument road network in prioritization order. During each iteration, the algorithm will use up to two polylines of the road network for extending polyline pl. Variable $epls$ holds these polylines.

Lines 7 to 10 are iterated through for the two delimiting points of polyline pl. These points are returned by functions $\mathbf{start}(pl)$ and $\mathbf{end}(pl)$. Line 7 computes the set of candidate extension polylines, $cand$, for a delimiting point. If stc is true, extension polylines must have the same street code as the polyline being extended.

If candidate extension polylines exist, the algorithm proceeds with lines 9 and 10; otherwise, it proceeds with the next delimiting point or maintenance of rn and cn. In line 9, the first of the candidate polylines according to the argument candidate prioritization sort order is identified and used for extending polyline pl. The polyline used for extending pl is added to set $elps$ in line 10. Next, if extension was successful, the polylines used for extension are subtracted from rn, and the extended polyline is added to rn. Otherwise, polyline pl is added to the result set.

The algorithm returns the modified road network. It should be noted that the algorithm does not modify the street codes of segments. If stc is $true$, a segment is extended only with segments with identical street code, which implies that the street codes on the resulting road network are correct. If stc is false, the street codes on the resulting road network are not meaningful. Subsequent algorithms do not use street codes when this is the case.

The worst-case running time complexity of the algorithm is $\mathcal{O}(n^3)$ where n is the number of polylines in the argument road network. The analysis is as follows: The main while loop executes at most n times. Within this loop, line 7 may involve iteration over all polylines in the road network. However, the worst-case complexity is caused by the presence of the **first** function calls. In the worst case, it takes n iterations to determine the value for an attribute specified in a sort ordering, and it takes another n iterations to find the first element given the sort ordering attribute values. It should be noted that the road-network modification approaches based on this algorithm are executed only once and in an initial, offline preprocessing step. The running times of the modifications thus do not affect the runtime performance of the tracking.

3.2 Street Code-Based Approach

The general idea is to give priority to connecting polylines with the same street code. This way, longer road segments are constructed that tend to correspond to parts of named streets. In cases where there are several candidates with the same street code, priority is given to the shortest polyline. This strategy reduces the probability that unconnected polylines will be short. The polyline connection algorithm for the street code-based approach is defined as follows:

```
Algorithm NSC($rn$)
1. $rn \leftarrow$ GSC($rn, true, [streetcode_{asc}, length_{asc}],$
                  $[sides_{desc}, length_{desc}])$
2. $rn \leftarrow$ GSC($rn, false, [streetcode_{asc}, length_{asc}],$
                  $[sides_{desc}, length_{desc}])$
3. return $rn$
```

Algorithm **NSC** makes two calls to algorithm **GSC** using differing parameters. The first call uses the argument road network rn and requires that polylines being connected have identical street codes.

The sort order used for specifying the iteration over the road network sorts polylines primarily according to ascending street codes and secondarily according to their ascending lengths. The algorithm thus processes segments in street code order and gives priority to short segments. Notice that a sorting order of $streetcode_{desc}$ would also work instead of $streetcode_{asc}$. Both ensure that the polylines with the same street code are processes together, which is what we want to achieve.

The sort order used when selecting the best candidate polyline for extending a polyline first orders the candidate polylines in descending order according to $sides$, which has values 1 or 2 depending on how many sides to which the polyline can be extended (as it is a candidate, the value is at least 1). The secondary ordering is according to descending length. As a result, candidate segments are preferred that can be extended further, and among such candidates, the longest are preferred.

The second call to **GSC** is applied to the result of the first call. In contrast to the first call, street codes are not taken into account when connecting polylines. The sort orderings used are the same as those used in the first call to **GSC**.

3.3 Tail Disconnection Approach

The street code-based approach does not distinguish between main roads and side streets. The underlying observation that motivates the tail disconnection approach is that moving object can be assumed to be moving on main roads most of the time. In this approach, we thus first connect polylines disregarding side streets, termed tails, and we only subsequently take the tails into account.

Definition 3.1 (Tails). *Let $rn \subset \mathcal{PL}$ be a set of polylines. A polyline $pl \in rn$ is a tail if at least one delimiting point of pl is not connected to any delimiting point of any other polyline on rn. Tails are also termed first level tails. The ith level tails in rn are those polylines that are tails in the set obtained by subtracting all tails at lower levels than i from rn. We define $\mathrm{Tails}(rn)$ of a road network rn as the set of pairs $(pl, level)$ of a polyline pl in rn and a level number $level$ in $\mathbb{N}$ such that pl is a tail at level $level$.*

A few comments are in order. If a road network has a purely hierarchical structure, each polyline may be a tail at some level. Polylines that belong to a circular structure in a road network, i.e., a structure where each constituent polyline is connected at both ends, are not tails. A highest tail level is assigned to all non tail polylines (e.g., $1 + \max(\{level | (pl, level) \in \mathrm{Tails}(rn)\})$).

The polyline connection algorithm for this approach is defined as follows:

```
Algorithm TSC($rn$)
1. $rn \leftarrow$ GSC($rn, true, [streetcode_{asc}, taillevel_{desc}, length_{asc}],$
                  $[taillevel_{desc}, sides_{desc}, length_{desc}])$
2. $rn \leftarrow$ GSC($rn, false, [streetcode_{asc}, taillevel_{desc}, length_{asc}],$
                  $[taillevel_{desc}, sides_{desc}, length_{desc}])$
3. return $rn$
```

Algorithm **TSC** has the same structure as **NSC**. In line 1, the first call to **GSC** requires that polylines being connected have identical street codes.

The sort order used for specifying the iteration over the road network sorts polylines according to ascending street code, then according to descending tail level, and finally according to ascending lengths. The sort order used when selecting a candidate polyline for extending a polyline first orders the candidate polylines in descending order according to tail level, then in descending order according to $sides$, and then according to descending length.

These sort orders ensure that nontail polylines are connected first. Tails will be used only when no nontail polylines are available. Using tails with the highest levels first is also always beneficial, as a polyline with tail level n can be extended with a polyline with tail level $n - 1$.

The second call to **GSC** is applied to the result of the first call. Here, connections between polylines with different street codes are allowed. The sort orderings used are the same as those used in the first call to **GSC**.

It should be noted that because algorithm **GSC** does not update tail levels, a segment being extended retains its tail level. This is exactly as intended.

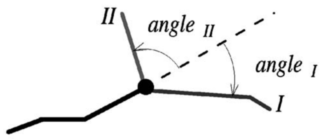

Fig. 4. Angles beween polylines.

3.4 Direction-Based Approach

The last approach takes into account the directions of the candidate polylines at the connection point. The idea is that moving objects are expected to be moving as directly as possible toward their destinations, which means that they will tend to move as straight as possible and by making as few turns as possible.

This approach thus gives preference to polylines that continue in the same direction as much as possible when extending a polyline. Put differently, preference is given to polylines with a direction at the connection point that has a small angle with respect to the direction of the polyline to be extended, again at the connection point.

The polyline connection algorithm for the direction-based approach is defined as follows:

```
Algorithm DSC($rn$)
1. $rn \leftarrow$ GSC($rn, true,$
       $[streetcode_{asc}, taillevel_{desc}, angleAvg_{asc}, length_{asc}],$
       $[taillevel_{desc}, sides_{desc}, angle_{asc}, length_{desc}])$
2. $rn \leftarrow$ GSC($rn, false,$
       $[streetcode_{asc}, taillevel_{desc}, angleAvg_{asc}, length_{asc}],$
       $[taillevel_{desc}, sides_{desc}, angle_{asc}, length_{desc}])$
3. return $rn$
```

The algorithm extends the **TSC** algorithm by introducing the new properties *angle* and *angleAvg*.

Property *angle* denotes the angle between a polyline being extended and a candidate for use in the extension. Specifically, the line segment at the connection point of the polyline to be extended is itself extended toward the candidate polyline. This extension corresponds to a straight extension of the polyline's line segment at the connection point. Property *angle* is then the angle between the extended line segment and the line segment of the candidate polyline at the connection point. See Fig. 4. A small angle is thus preferable.

Next, property *angleAvg* of a polyline being extended denotes the average of the smallest *angle* values possible for both ends of the polyline. If the polyline cannot be extended to one side, an angle of 180 degrees is used for that side. Thus, for a polyline that can be extended with three polylines to one side with angles of 34, 22, and 90 degrees, respectively, and that has no extensions on the other side, $angleAvg = (22 + 180)/2$. The other parameters are the same as in the TSC algorithm.

3.5 Comparison of Approaches

The goal of all the road modification approaches is to connect the polylines of road segments into longer polylines, so that moving objects travel on fewer polylines. In doing this, we assume that objects in a road network move mostly along the main roads. We proceed to evaluate the results of the road network modifications in terms of how well the constructed polylines correspond to the main roads.

All policies succeeded in connecting short polylines into longer ones. Before modification, the road network has 14,708 segments in total, and the average length of a segment is 174 m. Application of each of the three

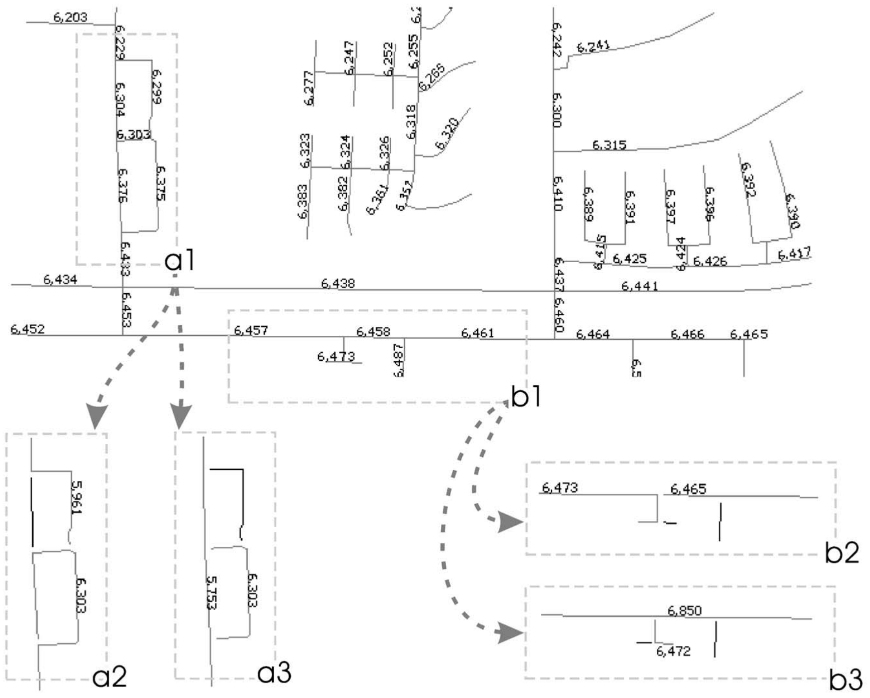

Fig. 5. Road network modifications.

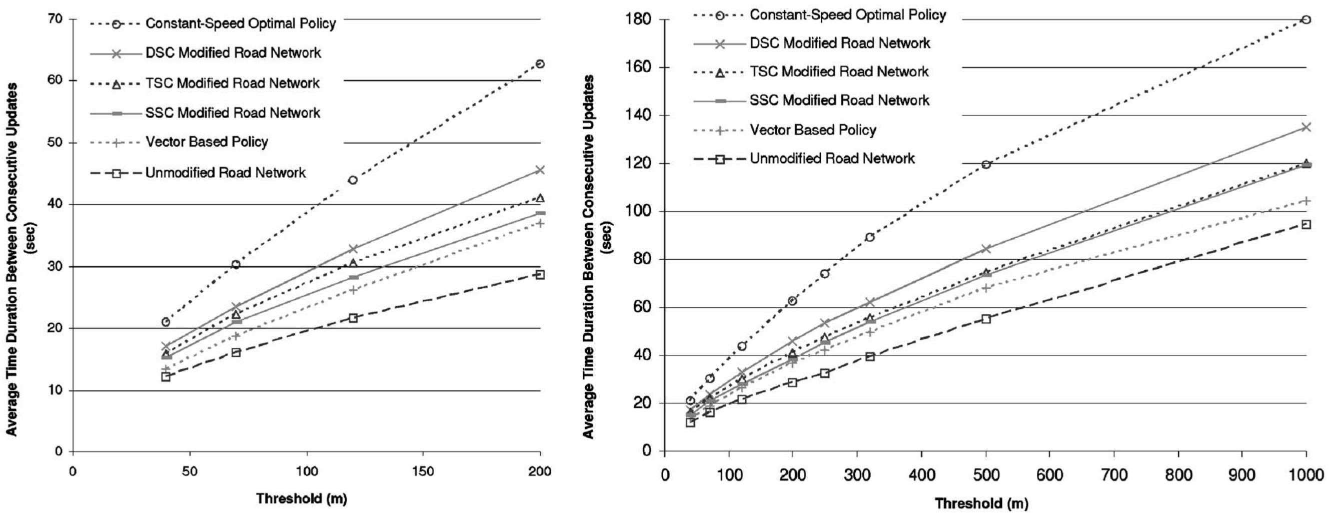

Fig. 6. Comparison of road network modifications.

modification approaches resulted in close to 5,800 polylines and an average length of about 450 m.

Before presenting the results of general experiments with the different modified networks, we use the examples in Fig. 5 to offer the reader a feel for the approaches and their differences. The figure displays part of the unmodified road network at the top, with two smaller parts, labeled a1 and b1, being identified for further consideration.

The polylines created by the street-code-based approach were the worst in connecting the main roads. (In residential and other areas, it is common for a main road and its side streets to have the same street code.) Parts a2 and b2 exemplify how the street-code-based approach fails to capture the main road as a single polyline, or segment, for these two parts of the network. In Part a2, the main road is vertical and straight; in Part b2, it is also fairly straight, but horizontal. The number next to a short polyline is the identifier ("id" for short) of the "long" polyline to which it belongs. A bold line without an id represents an unconnected polyline.

In Part a2, segment 6,303 makes a loop, and segment 5,961 starts on the side road, while the main road has a single segment that was not extended by the modification procedure. In Part b2, segment 6,473 "ends" because of a turn to a side road.

The tail disconnection approach fixes some of the problems. In Part b3 of the figure, a single segment now represents the main road. This is in contrast to the situation in Part b2, where two segments share the main road. Specifically, we see that segment 6,850 represents the entire main road on the map and that side roads are "eliminated." However, the tail disconnection approach does not improve the situation in Part a2, where there are no tail polylines.

The direction-based approach is the best at assigning main roads to few polylines. This approach solves problems like those shown in Parts a2 and b2. Since priority is given to straight extensions of polylines, a single polyline, with id 5,753, represents the straight part of the road—see Part a3. With the direction-based approach, and unlike the two other approaches, the main road is represented by one polyline, and the side roads are represented by two polylines.

In Fig. 6, we present a comparison of the update performances for the segment-based policy using the unmodified road network and the road networks resulting from application of the street-code-based approach (algorithm SSC), the tail disconnection approach (algorithm TSC), and the direction-based approach (algorithm DSC). The vector-based policy is also included.

In the comparison, 568,307 GPS records were used. The curves to the right show experimental results using thresholds ranging from 40 to 1,000 m, and the curves to the left provide a better view of the results for thresholds in the range of 40 to 200 m.

All three road network modifications increase the performance of the segment-based policy and outperform the vector-based policy. The segment-based policy has the best performance when using the road network resulting from the direction-based modification. The performance of a theoretical, constant-speed optimal policy, to be explained and discussed in the next section, is also included in Fig. 6. This policy in effect assumes that a moving object always stays on the same segment and moves at constant speed, meaning that updates thus only occur due to speed variations. Based on these experiments, we select the direction-based approach as the best of the three road network modification approaches.

3.6 Comparison of Suburban versus City Driving

Another round of experiments were conducted to see the effects of city versus suburban driving. For these experiments, we used GPS logs from 10 cars that total more than one million GPS points. We divided these points into two parts, .56 million points that are located within a rectangular region enclosing the center of Aalborg were designated as city points, and .45 million points outside this region were designated as suburban points. Fig. 7 shows how different techniques perform for the city and suburban data. Specifically, we consider the segment-based policy with an unmodified road network and the network resulting from application of the direction-based approach

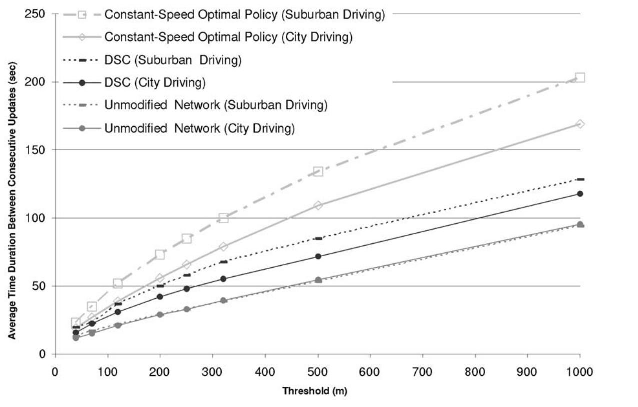

Fig. 7. Segment-based and "optimal" tracking for city and suburban driving.

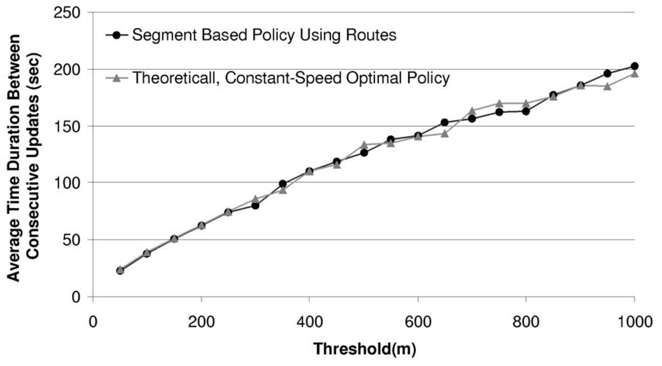

Fig. 8. Use of routes versus the theoretical approach.

transformation (algorithm DSC), and we consider the "optimal" policy. Thresholds range from 40 to 1000 m.

We expect smoother speed variations for suburban driving than for city driving. The better performance of the "optimal" policy (which is sensitive only to speed variations) for suburban driving than for city driving confirms this. It can also be observed that the experiments with the unmodified road network differ little for city and suburban driving. This suggests that using the unmodified road network, the majority of updates happens due to segment changes, not due to speed variation. It should also be noticed that use of the transformed road network yields better performance for city as well as suburban driving, in comparison to use of the unmodified network. This indicates that many updates caused by segment changes were avoided. Finally, it is observed that, with the modified network, the segment-based policy performs better for suburban driving than for city driving. This may be due to both the smoother suburban speed variations and longer suburban road segments.

4 Update Reduction Using Routes

The focus of this section is the use of the routes of moving objects for update reduction. At first, we introduce a theoretical, constant-speed optimal policy. Then, we consider the use of a user's routes, which are "long" segments, in the segment-based policy instead of the use of road-network segments.

4.1 Theoretical, Constant-Speed Optimal Policy

One may distinguish between the updates sent from client to server based on the outcomes of the associated map matching. Recall that in segment-based tracking, when the server receives an update with a position p_i, it attempts to map match the position onto the road network, rn, to find the most probable polyline mpl and point mp on it.

With $\mathbf{MM}$ being the map matching function then $(mpl, mp) = \mathbf{MM}(p_i, rn)$. If $\mathbf{MM}(p_i, rn) = (\texttt{null}, \texttt{null})$, the map matching is unsuccessful and tracking is done in vector mode. Assuming that the map matching is successful and expression $\big(\mathbf{MM}(p_i, rn)\big).mpl$ returns polyline mpl to which a given point p_i is map matched, then if $\big(\mathbf{MM}(p_{i-1}, rn)\big).mpl = \big(\mathbf{MM}(p_i, rn)\big).mpl$, we say that the update is caused by *speed*, while if $\big(\mathbf{MM}(p_{i-1}, rn)\big).mpl \neq \big(\mathbf{MM}(p_i, rn)\big).mpl$, we say that the update is caused by a *segment change* (position p_{i-1} is that of the previous update).

The theoretical, constant-speed optimal policy introduced here indicates how few updates it is possible to achieve with the segment-based policy in the best case that occurs when a moving object travels on only one segment and no updates occur due to segment change. The policy is optimal under the assumption that the speed of a moving object is modeled as being constant in-between updates.

This policy is included here because it gives a measure of optimality under the assumption of constant-speed prediction. The policy is used for comparison purposes only and is not a practical policy. The policy is impractical because it assumes that the entire polyline along which a vehicle will ever move is known in advance. We are able to use this policy here because we have the entire GPS logs for each vehicle. Using these, we simply construct (very long) polylines that precisely track each vehicle "ahead of time." In practice, we receive GPS positions in real time.

In Fig. 6, the curve for the constant-speed optimal policy gives the lower bound for the number of updates needed by the segment-based policy. The deviation of the segment-based policy using the unmodified road network from the optimal case is substantial. Using the modified road networks, the performance is significantly closer to the optimal case. For example, for a threshold of 200 m, the use of the road network modified using the direction-based approach increases the average time duration in-between consecutive updates from 28 to 46 s in comparison to the use of the unmodified road network.

4.2 Use of Routes

It seems reasonable to assume that individuals who travel are traveling in order to reach a destination. Folklore also has it that travelers frequently use the same routes to their destinations. For example, a person going from home to work may be expected to frequently use the same route. This general type of behavior is confirmed by the GPS logs we have available [9].

Taking advantage of knowledge of the routes used by a moving object can reduce the number of updates caused by segment changes. Since a route is a sequence of partial road segments, a route is represented simply as a polyline. Therefore, the segment-based policy, which is applicable to

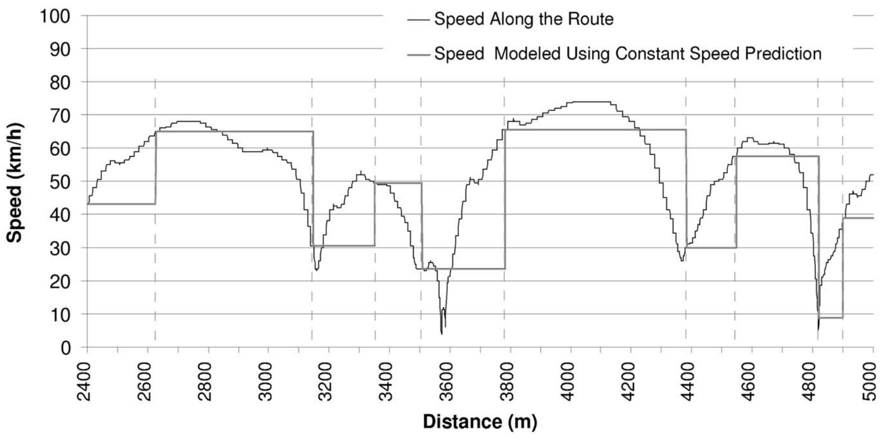

Fig. 9. Speed modeling using constant speed prediction.

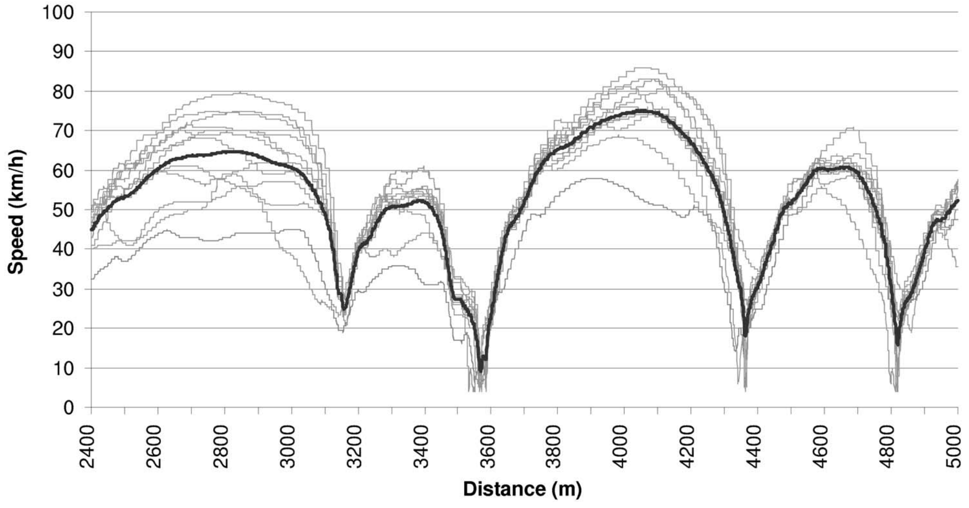

Fig. 10. Speed pattern for 12 traversals of a partial route.

any polylines, is also *directly* applicable to routes. All that is needed is to collect the routes of each user [2].

When using the segment-based policy with routes, we effectively assume that we know the future positions of an object. This is like in the theoretical, constant-speed optimal policy. The differences from that policy are that the polylines that represent routes are created from the road network, not from GPS logs, and that deviations from the assumed route are handled. Specifically, if an object deviates from its route, this is treated simply as a segment change. This will most likely trigger an update, but it will not lead to failure.

For experiments, we have extracted log data that represents routes from home to work of drivers represented in our GPS data. The data set contains 56,000 log entries. Using this data, Fig. 8 reports the performance of the segment-based policy when routes are used, as well as of the theoretical approach based on the same data.

The policies have practically the same performance. The small deviations between the two are only visible for higher threshold values, which is due to the small numbers of updates for these. For example, at a 50 m threshold, each policy has more than 1,850 updates, which renders the difference of 96 updates invisible. At a 950 m threshold, each policy has just above 228 updates, rendering the difference of 14 updates visible. The slight deviations between the policies may be explained by the differences between the routes used by the two policies. In particular, the routes constructed from GPS points and used by the theoretical policy are slightly more detailed and, thus, longer than the ones used by the segment-based policy.

The conclusion is that knowing the route of an object in advance can eliminate virtually all updates caused by segment changes and thus significantly improves the performance of the segment-based policy.

5 Update Reduction Using Acceleration Profile

Even if the future trajectory of an object is known precisely and updates caused by segment changes thus are eliminated, updates still occur due to variations in speed. The reason is that the segment-based policy assumes that objects move at constant speed—it takes an update to change the speed.

In this scenario, the modeled speed of an object moving along a road is a stair function. Fig. 9 presents the variation of a car's speed along a part of its route from home to work. The stepwise constant speed is the one used by the segment-based policy with a 70 m threshold. Each new step in the stair function is marked with a dashed line and

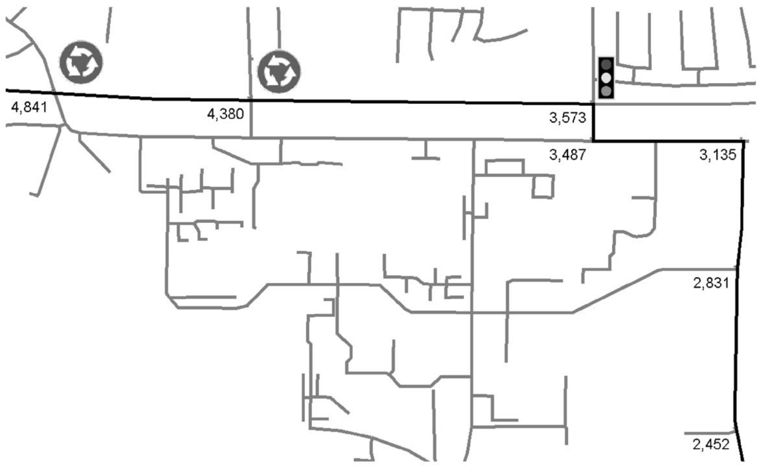

Fig. 11. Geometry of partial route with indications of traffic lights and rotaries.

represents an update. The density of the steps depends on the threshold—smaller thresholds yield more updates.

It is reasonable to expect that more accurate modeling of the speed variation of an object along its route, e.g., using averages of the speeds during past traversals of the route, can help better predict the future position of the object as it moves along the route. Fig. 10 illustrates the speed variation of one car along part of its route from home to work. Here, the thin lines represent the speeds for 12 different traversals of the route, and the solid line represents the average speed along the route.

The figure reveals a clear pattern in how fast the car drives along the route during different traversals. The geometry of the route, the driver's habits, and the traffic situation are probably the primary causes for this correlation. Fig. 11 displays the geometry of our partial route. The figure contains distance measures that allow the reader to correlate the geometry with the patterns displayed in Fig. 10. The first deceleration of the car happens in preparation for negotiating a sharp 90 degree curve. Then, the car accelerates, decelerates, makes a right turn, and decelerates further as it reaches a traffic light. On green, the car accelerates along a main road where it subsequently passes through two large rotaries. It can be seen that the car reaches its highest speed on the long, straight stretch of main road and that the speed as it enters a rotary is on average higher than the speed at the traffic light. It can also be seen that the car decelerates more quickly than it accelerates. We expect this type of behavior to be typical.

The clear pattern in Fig. 10 indicates that tracking with better performance can be achieved by more accurate modeling of the predicted, future speed of a moving object.

We consequently create an acceleration profile for capturing the average speed variation of the movement of an object along a route. It should be noted that a profile is created for each combination of a route and object using the route. Assigning profiles to the road network that are to apply to all moving objects and for all uses of the segments of the road network is expected to be less useful. We assume the presence of a separate software component that generates frequently used routes for the moving objects being tracked [2]. Having this be a separate component is reasonable, as routes are useful for other tasks than tracking.

An acceleration profile consists of acceleration values together with the distance intervals during which these values apply. A profile is created by first dividing the average speed variation along the route into intervals where the acceleration changes sign (i.e., from positive to negative or vice versa). Then, the average acceleration is calculated for each interval. We define an acceleration profile apf as a sequence of $n+1$ measures m_i and n accelerations a_i, $(m_0, a_0, \ldots, m_{n-1}, a_{n-1}, m_n)$. Acceleration a_i is valid in interval $[m_i, m_{i+1})$.

To see how an acceleration profile is used, assume an object moves with speed v_0 and that its current location ("measure") along the route is m_0 distance units after the start of the route, where m_0 belongs to the interval $[m_{begin}, m_{end})$ in which the acceleration profile has acceleration value a. Then, the predicted position m_{pred} and speed v_{pred} of the object within interval $[m_{begin}, m_{end})$ at time t is given by: $m_{pred} = m_0 + v_0 t + (a/2)t^2 \;\; (v_{pred} = v_0 + at)$.

Fig. 12 exemplifies speed modeling when using an acceleration profile. The figure concerns the movement of one moving object along a route. We assume that the segment-based policy with a 70 m threshold is used. In this figure, the light vertical dotted lines mark updates. To provide better insight into the behavior of the policy used, we include the deviation between the real position of the moving object and its position as predicted by the policy.

The algorithm "Predict Positions with Segment policy and Acceleration profile," **PPSA**, extends a previously proposed algorithm [4] with the ability to modify the speed of an object according to an acceleration profile.

The algorithm takes two parameters as input, $mopa$ and t, where the first parameter is a structure with five elements:

1. a polyline, $mopa.pl$, that specifies the geometrical representation of the moving object's route,
2. an acceleration profile, $mopa.apf$, for speed prediction along the route,
3. the location of the client, $mopa.m$, given as a measure value on the route,
4. the speed, $mopa.plspd$, of the object, and
5. the time, $mopa.t$, when the location and speed were acquired. Parameter $t > mopa.t$ is the time point for which the location of the object should be calculated.

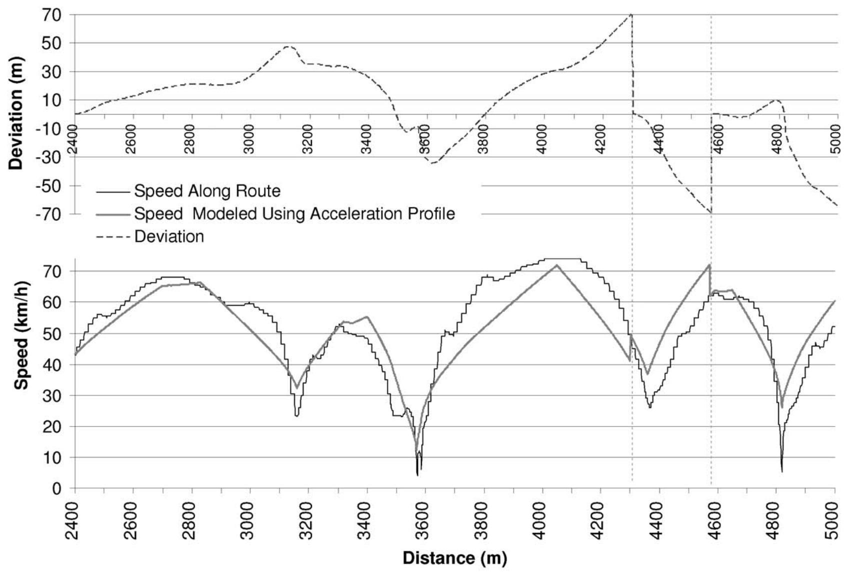

Fig. 12. Speed modeling using acceleration profile.

The result is the coordinates of predicted location of the object at time t.

```
Algorithm PPSA$(mopa, t)$
1. $m_{pred} \leftarrow mopa.m$
2. $v_{pred} \leftarrow mopa.plspd$
3. $t_{pred} \leftarrow t - mopa.t$
4. while $t_{pred} > 0$ do
5.     $accel \leftarrow$ getAcceleration$(m_{pred}, mopa.apf)$
6.     $S \leftarrow accel.end - m_{pred}$
7.     $dt \leftarrow 0$
8.     if $v_{pred}^2 + 2 \cdot accel.a \cdot S \geq 0 \wedge accel.a \neq 0$ then
9.         $dt_1 \leftarrow \big(-v_{pred} + \sqrt{v_{pred}^2 + 2 \cdot accel.a \cdot S}\big)/accel.a$
10.        $dt_2 \leftarrow \big(-v_{pred} - \sqrt{v_{pred}^2 + 2 \cdot accel.a \cdot S}\big)/accel.a$
11.        $dt \leftarrow \max\big(\{0, \min(\{dt | dt \in \{dt_1, dt_2\} \wedge dt > 0\})\}\big)$
12.    if $dt = 0$ then $dt \leftarrow S/v_{pred}$
13.        $accel.a \leftarrow 0$
14.    if $t_{pred} < dt$ then $dt \leftarrow t_{pred}$
15.    $m_{pred} \leftarrow m_{pred} + v_{pred} \cdot dt + accel.a \cdot dt^2/2$
16.    $v_{pred} \leftarrow v_{pred} + accel.a \cdot dt$
17.    $t_{pred} \leftarrow t_{pred} - dt$
18. if $m_{pred} \geq \mathcal{M}\ (mopa.pl, mopa.pl.p_{end})$ then
       return $mopa.pl.p_{end}$
19. return $\mathcal{M}^{-1}(mopa.pl, m_{pred})$
```

The algorithm first initializes temporary variables: Variables m_{pred} and v_{pred} are set to contain starting location and speed of the moving object and variable t_{pred} initially holds the time elapsed since the time when the moving object's location was acquired. The object's movement should be predicted for this duration of time. In general, several acceleration intervals are passed through during this duration of time, meaning that different acceleration values should be applied during the prediction. The algorithm iteratively calculates the time required to pass through each acceleration interval and reduces the prediction time t_{pred} with this time. When the prediction time is exhausted (line 4), the loop stops, and the algorithm calculates and returns the coordinates of the predicted location.

In line 5, acceleration value a for the predicted location of the object m_{pred} and boundary point end of the acceleration interval where acceleration value a applies are retrieved and stored in $accel$; these are returned by function **getAcceleration**. In the case where m_{pred} is equal to boundary point m_i, the boundary point m_{i+1} of the next acceleration interval is returned. If there are no more acceleration intervals, an acceleration value of 0 is returned and the boundary point is set to ∞. Notice that m_{pred} is initially equal to the location of the object at the time of the update (line 1).

In line 6, the distance S to the end of the acceleration interval with acceleration $accel.a$ is calculated.

The time dt required for the object to reach the end of the acceleration interval (moving with acceleration $accel.a$) is calculated in lines 9-11. This time is calculated using the quadratic equation $accel.a \cdot dt^2/2 + v_{pred} \cdot dt - S = 0$. It has solutions only if $v_{pred}^2 + 2 \cdot accel.a \cdot S \geq 0$ (line 8), and only positive solutions are valid, as the meaning of the solution is time. If there are two positive solutions, the solution with the smaller value is the valid one (line 11). If the equation has no valid solution, the result dt is equal to 0. In this case, prediction using constant speed is performed (lines 12 and 13).

After the time required to reach the end of the acceleration interval is calculated, this time is compared to remaining prediction time t_{pred}. If the time left for which prediction should be done, t_{pred}, is less than time required to go distance S, then the algorithm does prediction only for time t_{pred} (line 14). Lines 15 and 16 then calculate the predicted location m_{pred} and speed v_{pred}. The prediction time is reduced in line 17 and the loop is repeated if $t_{pred} > 0$.

Finally, the coordinates corresponding to location m_{pred} are calculated and returned. This is done in lines 18 and 19. If the predicted location m_{pred} is beyond the end of the route as described by polyline $mopa.pl$ (line 18), the end point of

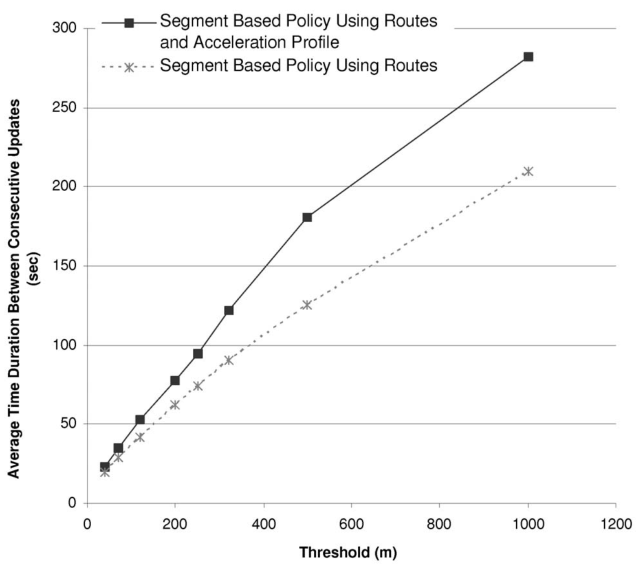

Fig. 13. Results using acceleration profile.

the polyline is returned. This is done by comparing the predicted measure on the polyline with the measure of the end point $pl.p_{end}$ of the polyline. Function $\mathcal{M}$ calculates the measure value on a given polyline of a given coordinate point. Otherwise, the coordinate point of m_{pred} is calculated with the inverse function $\mathcal{M}^{-1}$, which calculates the coordinate point of a given measure value on a given polyline.

Experimental results for the segment-based policy using routes and acceleration profiles are presented in Fig. 13.

These experiments are based on data from the movement of five cars along different routes. The GPS data set used here consists of a total of 57,202 records. The experiments shows that the use of acceleration profiles is able to improve performance. This confirms that, when knowing the past acceleration pattern of an object's movement along a route, it is possible to more accurately predict the future positions of the object along the route. For example, using a threshold of 250 m, the average time in-between updates is increased from 72 to 98 s. We note that with acceleration profiles, we outperform the previously introduced theoretical policy that was optimal only under the assumption of constant-speed prediction.

In closing, it is also worth considering a few speed modeling alternatives and some implication of our choice. In reality, the travel speed associated with a road segment varies during the day, and different drivers may well negotiate the same segment with different speeds. By associating acceleration profiles with routes that are specific to individual drivers, we capture the variation among drivers. And, because the same route (e.g., from home to work or from work to home) is typically used during the same time of the day, the variation of speeds across during the day is also taken into account fairly well. Next, if significant variations exist within the observations based on which the acceleration profile of a route is constructed, it is possible to create several speed profiles, e.g., so that rush hour and nonrush hour profiles are available. We did not find a need for several acceleration profiles in the GPS data we have used. Finally, we feel that the alternative of associating acceleration profiles with the road network itself leads to solutions that either will be more complex or will be less accurate.

6 Related Work

When predicting the future position of an object, the notion of *trajectory* is typically used [10], [14], [15], [19], where a trajectory is defined in 3-dimensional [14] or 4-dimensional [15] space. The dimensions are a two-dimensional "geographical" space, a time dimension, and an uncertainty thresholds dimension. A point in this space indicates when an object is in a given location and what the uncertainty of the representation of the location is. Such points may be computed using speed limits and average speeds on specific road segments belonging to a trajectory. Xu and Wolfson [19] use average real-time speeds reported every 5 minutes by in-road sensors. In our techniques, the prediction of an object's movement is done using the speed received from the object. For more accurate prediction, we introduce acceleration profiles that allow for quite accurate modeling of the speed variation along a route. An acceleration profile is a property of the combination of a physical road network and the habits of a concrete driver.

Wolfson et al. [16] propose two location update policies, termed immediate linear and delayed linear. These do not provide accuracy guarantees, as an object does not update its location when the deviation reaches some threshold. The occurrence of an update depends on the overall behavior of the deviation, estimated using a linear function, since the last update. Experiments on simulated data show that these policies are inferior to more recent policies, also by Wolfson et al. [17]. Like ours, these offer accuracy guarantees. Unlike ours, they assume that objects move on predefined routes already known to the objects and route selection is done on the client side. If an object changes its route, it sends a position update with information about the new route to the server. In contrast, we accommodate objects with memory restrictions, and we consider the case where routes are not known and where map matching may not even succeed.

Lam et al. [11] present an adaptive monitoring method (AMM) that takes into consideration not only update, deviation, and uncertainty costs, but also the cost of providing incorrect results to queries, during the process of determining when to issue updates. In AMM, the moving objects that fall into a query region need close monitoring, and a small update threshold is used for them. Objects not inside a query region may have big thresholds. Our algorithm allows different objects to have different thresholds and allows threshold to change dynamically.

Karimi and Liu [10] describe a technique for trajectory prediction. This technique assigns probabilities to the roads emanating from an intersection according to how likely it is that an object entering the intersection will proceed on them. The subroad network within a circular area around an object is extracted and the most probable route within this network is used for prediction. When the object leaves the current subnetwork, a new subnetwork is extracted and the procedure is repeated. The probabilities are not individual to each object, but are used for all objects and they do not take into account past choices during the trip of an object. In contrast, we use complete routes. To calculate routes, not only the trajectory of a moving object, but the time of the trip and start and destination points are taken into account. Moreover, we use speed profiles.

Wolfson et al. [19] have recently investigated how to incorporate travel-speed prediction in a database. They assume that sensors that can send up-to-date speed information are installed in the roads. In contrast, we use so-called GPS-based floating-car data and we predict positions based on historical records and for each moving object in isolation. This avoids the need for in-road sensors and for gathering information from such sensors.

Next, Wolfson and Yin [18] consider tracking with accuracy guarantees. Based on experiments with artificial data generated to resemble real movement data, they conclude that a version of the point-based tracking as discussed in Section 2.3 is outperformed by a tracking technique that resembles the segment-based tracking also discussed in Section 2.3. For a small threshold of 80 m, the latter is a bit more than twice as good as the former; for larger thresholds, the difference decreases. Their metric is numbers of updates per distance unit. They consider neither road-network modification, the use of routes, nor acceleration profiles. Their versions of point and segment-based tracking assume that map-matching always works and fail if this is not the case. This was possible because the data used in experiments was generated to be perfectly map matched. We believe that the techniques presented in Section 2 are good representatives of the techniques presented by Wolfson and Yin. It should also be noted that Ding and Güting [5] have recently discussed the use of what is essentially segment-based tracking within an envisioned system based on their own proposal for a data model for the management of road-network constrained moving objects.

Gowrisankar and Nittel [8] introduce a dead-reckoning policy that uses angular and road deviations, so that an update is issued whenever one of these deviations exceeds a defined threshold.

When only low accuracy of predicted positions are needed, cellular techniques [1], [12], [13] may be used. With such techniques, the mobile network tracks the cells of the mobile objects in real time in order to be able to deliver messages or calls to the objects. In this approach, update is handled in the mobile network. In contrast to these techniques, we consider scenarios where higher accuracy, well beyond those given by the cells associated with the base stations in a cellular network, are needed and where positioning with respect to a road network is attractive.

Assuming a network of geo-stationary "presence" sensors, Goel and Imielinski [7] propose to use an MPEG-based prediction model in order to determine the current location of an object while using as little sensor battery power as possible.

Next, Fox et al. [6] explore the use of statistical methods, e.g., multiple hypothesis tracking, in a more abstract location estimation context than the one we consider. Integration of such methods into our setting may enable more detailed analysis of the proposed tracking techniques.

In contrast to all related work, this paper uses a substantial data set of real GPS logs for guiding the process of designing practical techniques for the tracking of moving objects.

7 Summary, Conclusions, and Future Work

The paper presents and empirically evaluates several techniques for the segment-based tracking of moving objects. These extend the basic segment-based tracking previously proposed [4]. The proposed techniques are robust generally applicable: They function even if no underlying road network is available or if map matching is not unsuccessful, and then apply to mobile objects with even stringent memory restrictions.

The performance of basic segment-based tracking is sensitive to the segmentation of the road-network representation used and to the speed variations of the moving objects. Based on these observations, the paper presents several techniques that aim to reduce the number of updates needed for segment-based tracking with accuracy guarantees:

- *Road network modification*. The segment-based representation of the underlying road network used in segment-based tracking is modified with the goal of arriving at a segmentation that enables objects to use as few segments as possible as they move in the road network. This then reduces the number of updates caused by segment changes.
- *Use of routes*. A route is a polyline, constructed from (partial) road-network segments, that captures an object's entire movement from a source to a destination. As segments are themselves polylines, segment-based tracking readily accommodates the use of routes. Routes are specific to individual moving objects and the use of routes is expected to reduce the number of updates caused segment changes.
- *Use of acceleration profiles*. An acceleration profile divides a route into intervals with constant acceleration and thus enables quite accurate modeling of the speed of an object as it travels along a route. The idea underlying the use of acceleration profiles is to reduce the number of updates incurred by speed variations.

Experimental performance studies using real GPS logs and a corresponding real road network representation leads to the following main conclusions:

- It is possible to improve the performance of segment-based tracking by automatic resegmentation of the underlying road-network representation. Experiments with three resegmentation algorithms demonstrate this as well as offer insight into which types of modification are most effective in reducing the number of updates. Experiments with city and suburban driving indicate that segment-based tracking is more efficient for the latter.
- It is indeed very attractive to use precomputed routes for the moving objects in segment-based tracking, instead of using segments from the road-network representation. The GPS logs used confirm conventional wisdom, that mobile users are creatures of habit (or efficiency) that frequently use the same routes through the road network to reach their destinations.
- The GPS data used also reveal distinctive speed patterns for the mobile users. The experimental results show that the use of acceleration profiles increases the performance of segment-based tracking. With acceleration profiles, tracking with a 200 m accuracy guarantee can be done with an average of one update each 77 s. This is in contrast to one update every 30 s for basic segment-based tracking.

Several promising directions for future work exist. First, it would be of interest to evaluate the costs of data transmission, in terms of actual phone-bill cost for a mobile user. Such modeling should take into account the pricing policies of mobile network operators. Second, it would be interesting to study the creation and incremental maintenance of acceleration profiles further. Self-learning techniques may be applicable. Third, a road network can be modified according to the GPS data collected from all users. This way, the connection of the road segments can be based on the use of the road network by the users. This may lead to longer segments for the majority of users, thus improving the performance of the segment-based tracking.

ACKNOWLEDGMENTS

The authors wish to thank the handling editor for careful and timely handling of the manuscript. This work was supported in part by grants from ETRI, South Korea, from the Danish National Center for IT Research, and from the Nordic Academy for Advanced Study. In addition to his primary affiliation with Aalborg University, C.S. Jensen is an adjunct professor in the Department of Technology, Agder University College, Norway.

REFERENCES

[1] I.F. Akyildiz and J.S. M. Ho, "A Mobile User Location Update and Paging Mechanism under Delay Constraints," *ACM-Baltzer J. Wireless Networks*, vol. 1, pp. 244-255, 1995.

[2] A. Brilingaitė, C.S. Jensen, and N. Zokaitė, "Enabling Routes as Context in Mobile Services," *Proc. ACM Int'l Workshop Geographic Information Systems*, pp. 127-136, 2004.

[3] J.D. Chung, O.H. Paek, J.W. Lee, and K.H. Ryu, "Temporal Pattern Mining of Moving Objects for Location-Based Services," *Proc. Int'l Conf. Database and Expert Systems Applications*, pp. 331-340, 2002.

[4] A. Čivilis, C.S. Jensen, J. Nenortaite, and S. Pakalnis, "Efficient Tracking of Moving Objects with Precision Guarantees," *Proc. Int'l Conf. Mobile and Ubiquitous Systems: Networking and Services*, pp. 164-173, 2004, extended version available as DB-TR-5, Dept. of Computer Science, Aalborg Univ., Denmark, http://www.cs.aau.dk/DBTR/DBPublications/DBTR-5.pdf.

[5] Z. Ding and R.H. Güting, "Managing Moving Objects on Dynamic Transportation Networks," *Proc. Int'l Conf. Scientific and Statistical Database Management*, pp. 287-296, 2004.

[6] D. Fox, J. Hightower, L. Liao, D. Schultz, and G. Borriello, "Bayesian Filters for Location Estimation," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 24-33, 2003.

[7] S. Goel and T. Imielinski, "Prediction-Based Monitoring in Sensor Networks: Taking Lessons from MPEG," *ACM Computer Comm. Rev.*, vol. 31, no. 5, 2001.

[8] H. Gowrisankar and S. Nittel, "Reducing Uncertainty in Location Prediction of Moving Objects in Road Networks," *Proc. Conf. Geographic Information Science*, 2002, http://www.spatial.maine.edu/~nittel/publications/giscience02_hari.pdf.

[9] C.S. Jensen, H. Lahrmann, S. Pakalnis, and J. Runge, "The INFATI Data," Aalborg Univ., TimeCenter TR-79, 2004, http://www.cs.aau.dk/TimeCenter.

[10] H.A. Karimi and X. Liu, "A Predictive Location Model for Location-Based Services," *Proc. ACM Int'l Symp. Advances in Geographic Information Systems*, pp. 126-133, 2003.

[11] K.Y. Lam, O. Ulusoy, T.S.H. Lee, E. Chan, and G. Li, "An Efficient Method for Generating Location Updates for Processing of Location-Dependent Continuous Queries," *Database Systems for Advanced Applications*, pp. 218-225, 2001.

[12] G. Li, K. Lam, and T. Kuo, "Location Update Generation in Cellular Mobile Computing Systems," *Proc. Workshop Parallel & Distributed Real-Time Systems*, p. 96, 2001.

[13] Z. Naor and H. Levy, "Minimizing the Wireless Cost of Tracking Mobile Users: An Adaptive Threshold Scheme," *Proc. IEEE INFOCOM*, pp. 720-727, 1998.

[14] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain, "The Geometry of Uncertainty in Moving Objects Databases," *Proc. Int'l Conf. Extending Database Technology*, pp. 233-250, 2002.

[15] O. Wolfson, "The Opportunities and Challenges of Location Information Management," *Proc. Intersections of Geospatial Information and Information Technology Workshop*, 2001.

[16] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez, "Cost and Imprecision in Modeling the Position of Moving Objects," *Proc. Int'l Conf. Data Eng.*, pp. 588-596, 1998.

[17] O. Wolfson, A.P. Sistla, S. Camberlain, and Y. Yesha, "Updating and Querying Databases that Track Mobile Units," *Distributed and Parallel Databases*, vol. 7, no. 3, pp. 257-287, 1999.

[18] O. Wolfson and H. Yin, "Accuracy and Resource Concumption in Tracking and Location Prediction," *Proc. Symp. Spatial and Temporal Databases*, pp. 325-343, 2003.

[19] B. Xu and O. Wolfson, "Time-Series Prediction with Applications to Traffic and Moving Objects Databases," *Proc. ACM Int'l Workshop Data Eng. for Wireless and Mobile Access*, pp. 56-60, 2003.

Alminas Čivilis received the MS degree in computer science from Aalborg University, Denmark, in 2003. He is currently a PhD candidate in the Department of Computer Science II, Vilnius University, Lithuania. His research interests include the management of moving objects in location-based services, spatial data mining, and geographic information systems.

Christian S. Jensen received the PhD and DrTechn degrees from Aalborg University, Denmark, in 1991 and 2000, respectively. He is a professor of computer science at Aalborg University, an honorary professor at Cardiff University, United Kingdom, and an adjunct professor at Agder University College, Norway. His research concerns data management technology and spans issues of semantics, modeling, and performance. With his colleagues, he receives substantial national and international funding for his research and he has authored or coauthored more than 150 scientific papers. He is a member of the Danish Academy of Technical Sciences. He received the Ib Henriksens Research Award 2001 for his research in mainly temporal data management and Telenor's Nordic Research Award 2002 for his research in mobile services. He is on the editorial board of *ACM Transactions on Database Systems*, and he has served on the editorial boards of *IEEE Transactions on Knowledge and Data Engineering* and the *IEEE Data Engineering Bulletin*. He was the general chair of the 1995 International Workshop on Temporal Databases and a vice program committee chair for the 1998 IEEE ICDE Conference. He was coprogram committee chair for the Workshop on Spatio-Temporal Database Management, held with VLDB '99, and for the Eighth International Symposium on Spatial and Temporal Databases, and he was the program committee chair for the 2002 EDBT Conference. In 2005, he will be technical program chair of the VLDB conference. He serves on the boards of directors and advisors for a small number of companies. He serves regularly as a consultant and delivers lectures to industrial audiences. He is a senior member of the IEEE and a member of the IEEE Computer Society.

Stardas Pakalnis received the M.S. degree in computer science from Aalborg University, Denmark, in 2003. He is currently a PhD candidate in the Department of Computer Science and Engineering, Aalborg University. His research interests include mobile services and spatial and temporal databases.