DSTTMOD: A Discrete Spatio-Temporal Trajectory Based Moving Object Database System

Xiaofeng Meng[1] Zhiming Ding[2]

[1] Information School
Renmin University of China, Beijing 100872, China
xfmeng@mail.ruc.edu.cn

[2] Praktische Informatik IV
Fernuniversität Hagen, D-58084 Hagen, Germany
zhiming.ding@fernuni-hagen.de

Abstract. In this paper, a new moving objects database model - Discrete Spatio-Temporal Trajectory Based Moving Objects Database (DSTTMOD) model, is put forward. Trajectories are used to represent dynamic attributes of moving objects, including the past, current, and future location information. Moving objects can submit moving plans of different length according to their moving patterns. Moreover, they can divide the whole moving plan into multiple sections, and submit each section only when it is about to be used. Different moving objects can set up different threshold to trigger location updates. When a location update occurs to a moving object, not only its future trajectory is updated, but also the corresponding index records are adjusted. The model can support three kinds of queries (*point queries*, *range queries*, and *K-nearest neighbor* (KNN) *queries*) for location information in not only the near future, but also the far future. In order to evaluate the performance of the DSTTMOD model, a prototype system is developed and a series of experiments are conducted which show promising performance.

1 Introduction

With the development of wireless communications and positioning technologies, the concept of moving objects databases (MOD) has become increasingly important, and has posed a great challenge to the database community. Existing DBMS's are not well equipped to handle continuously changing data, such as the location of moving objects [1]. Therefore, new location modeling methods are needed to solve this problem.

Recently, a lot of research has been focused on MOD technology, and many models and algorithms have been proposed. O. Wolfson *et al.* in [1-4] have proposed a Moving Objects Spatio-Temporal (MOST) model which is capable of tracking not

only the current, but also the near future position of moving objects. L. Forlizzi *et al.* in [5] have proposed a data model to describe the trajectories of moving objects, but their work has not discussed the location update problem, which limits the adaptability of the model. H. D. Chon *et al.* in [6] have proposed a Space-Time Grid Storage model for moving objects, which mainly deals with the storage problem and discusses the Ripple Effect. Besides, C. S. Jensen *et al.* in [7-8] have discussed the indexing problem for moving object trajectories. In their work, however, the trajectories are mainly used to represent the history information.

When querying moving objects, the users may be interested in the past or current position information. However, in many circumstances, location information in the future, especially in the far future, may be more attractive to querying users. For instance, the driver of a broken car would be more interested in service cars that will arrive within the next 20 minutes. In this case, the previously proposed models are not well equipped to provide this kind of information.

In order to solve the above problem, we put forward a new MOD model - Discrete Spatial-Temporal Trajectory Based Moving Object Database (DSTTMOD) model, in this paper. Our aim is to support queries for location information not only in the past and at present, but also in the future.

Compared with other previously proposed MOD models, DSTTMOD has the following features. (1) Trajectories are used to represent dynamic attributes of moving objects, including the past, current, and future location information. (2) Moving objects can submit moving plans of different length according to their moving patterns. Moreover, they can divide the whole moving plan into multiple sections, and submit each section only when it is to be used. (3) Different moving objects can set up different threshold to trigger location updates. When a location update occurs to a moving object, not only its future trajectory is updated, but also the corresponding index records are adjusted. (4) The model can support three kinds of queries (*point queries*, *range queries*, and *K-nearest neighbor* (*KNN*) *queries*) for location information covering a large time range from the past into the future.

2 DSTTMOD: the System

Figure 1 depicts the architecture of the DSTTMOD system we have implemented. The system consists of five components: *Route Map Generator*, *Moving Object Generator and Manager*, *Moving Object Indexer, Moving Object Simulator*, and *Query Processor.*

Route Map Generator is responsible for generating route maps (road networks) on which moving objects move. There are two different ways to generate route maps: automatic and user interactive. The generated route map will be stored in the database.

Moving Object Generator and Manager generates moving objects according to a specified route map. The user can choose to create a moving object either automatically or manually. Information concerning each moving object includes an identifier and a moving plan. The system then generates the corresponding trajectory according to the moving plan, which will be stored in the database.

Moving Object Indexer generates a spatial-temporal index for the moving objects managed by the system. In DSTTMOD, we use a Grid-file based indexing structure which is called GMOI to index moving object trajectories. We have made two major modifications to the original Grid-file based method [9] in order to reduce location update costs (see Section 4).

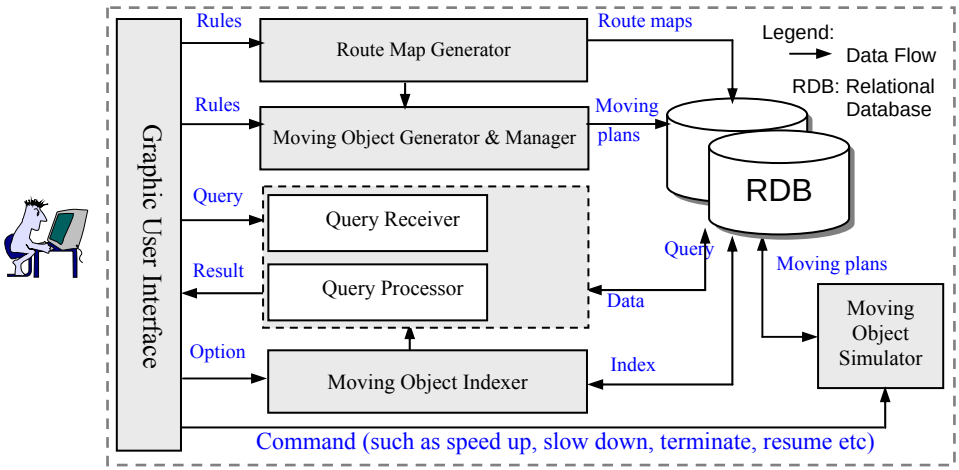


Fig. 1. Architecture of the system

Moving Object Simulator is responsible for simulating the running of moving objects. During its life time, a moving object can change its moving plan proactively, and it may also deviate from the predefined moving plan due to unforeseeable events (say traffic congestions). Both cases will trigger location updates. The system allows users to define parameters which will then affect the frequency of location updates and the uncertainty of the system.

Query Processor can support 3 kinds of queries concerning past, current, or future location information. Results of *Point queries* such as "tell me the current location of MO01" can be directly computed from the trajectories of moving objects. *Range queries* and *KNN queries* can be supported by the index structure of DSTTMOD. The system provides a GUI to interact with querying users.

3 Modeling Moving Objects with Future Trajectories

In the DSTTMOD model, we use a discrete method to describe the location information of moving objects. The whole trajectory of a moving object is represented by a set of line segments in the spatial-temporal space (For simplicity, we suppose that moving objects move on the two-dimensional X×Y plane. Thus, the spatio-temporal trajectory is a curve in three-dimensional X×Y×T space). Within each line segment, the movement of a moving object has the following properties:

a) Spatially, the moving object moves along a straight line;
b) The speed of the moving object keeps constant.

The whole complicated trajectory of a moving object can be represented by a set of such relatively simple line segments, and these line segments are then saved into the indexing structure so that they can be quickly retrieved. Since these line segments form a polyline in the X×Y×T space, we can simply use a set of vertexes to represent the whole trajectory, as defined below.

Definition 1(Trajectory): Suppose M is a moving object whose identifier is *MID*, then, its **Spatio-Temporal Trajectory**, denoted by $\overline{\xi}(MID)$, is defined as a finite sequence of points in the X×Y×T space:

$$\overline{\xi}(MID) = \left((x_i , y_i , t_i)\right)_{i=1}^{n}$$

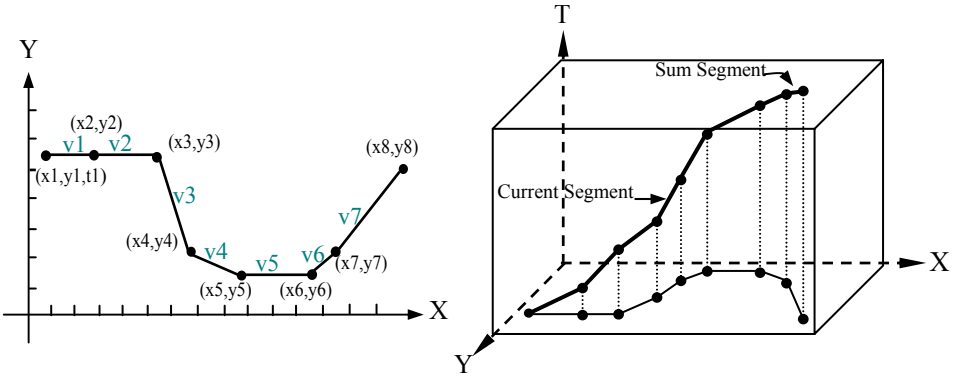
The expression $(x_i , y_i , t_i)(1 \le i \le n)$ is called the *i*th **key point**, denoted by $\overline{\xi}(MID) \bullet \rho(i)$. And the line segment $[(x_i , y_i , t_i), (x_{i+1} , y_{i+1} , t_{i+1})]$ is called the *i*th **spatio-temporal segment**, denoted by $\overline{\xi}(MID) \bullet Seg(i)$.

Definition 2: Suppose M is a moving object whose identifier is *MID*, $\overline{\xi}(MID)$ and $\overline{\xi}(MID) \bullet \rho(i)$ are its spatio-temporal trajectory and the *i*th key point respectively. We define that:

$\overline{\xi}(MID) \bullet \rho(i) \bullet \varphi$ is the element of $\overline{\xi}(MID) \bullet \rho(i)$ in φ-direction, $\varphi \in \{x , y , t\}$;

$\overline{\xi}(MID) \bullet CurSeg$ is the **serial number** of the spatio-temporal segment on which M is currently moving;

$\overline{\xi}(MID) \bullet SumSeg$ is the **total number** of spatio-temporal segments in $\overline{\xi}(MID)$.



(a) moving plan of a moving object (b) corresponding trajectory in the spatial-temporal space

Fig. 2. Location model of DSTTMOD

The spatio-temporal trajectory can represent not only the historical and current location of moving objects, but also the future locations. In order to get this information, moving objects need to submit their moving plans to the system in advance. During the process of moving, when the deviation of the actual location from the anticipative location exceeds a certain threshold, a location update is triggered. In this case, both the current and the subsequent segments of the trajectory need to be updated and the corresponding indexing structures must also be modified to reflect the up-to-date situation. Figure 2 illustrates the location model of DSTTMOD.

4 Indexing Moving Objects with Low Updating Cost

The index of moving object trajectories is essentially a spatial index in the three-dimensional space X×Y×T, and we can adopt conventional spatial indexing methods to index moving objects. However, this is not the best solution. In moving objects databases, location updates can cause the index structure to be updated frequently. As a result, special considerations should be taken to reduce the updating cost.

There are three main indexing methods for spatial or spatio-temporal data: R-tree and its variation, Quad-tree and its variation, and Grid-file and its variation. R-tree and Quad-tree are both multilevel nested structures in essential. As stated earlier, in MOD, moving objects are subject to frequent location updates, and when location updates occur, the whole indexing structure should be adjusted. Therefore, the index structure must provide optimal updating performance. It is obvious that these two methods are not proper for frequent location updates. In addition, in the process of updating, deleting former trajectory segments and inserting new ones may result in multilevel merges and splits, which can affect the performance enormously. However, it is different for Grid File. The index structure of Grid File is flat and the way of splitting is flexible. So Grid File is more suitable for the case of frequent location updates.

Based on the above analysis, we adopt a Grid-file based indexing method, which is called Grid File based Moving Objects Index (GMOI for short), to organize the trajectories of moving objects. In this section we describe the indexing structure and the corresponding algorithms.

Suppose that the indexed space is X×Y×T*, in which X×Y represents the geographic space of the application and T* represents a section of T axis. Because time extends infinitely, we deal with the period around current time instant when building index and make parallel translation of the indexed space along with time.

Definition 3 (Partition): A **Partition**, denoted by P, in the space of X×Y×T* is expressed as $P = P_x \times P_y \times P_t$. $P_x = (x_0, x_1, ..., x_l)$, $P_y = (y_0, y_1, ..., y_m)$ and $P_t = (t_0, t_1, ..., t_n)$ are sub-partitions in three directions, in which x_i $(0 \le i \le l)$, y_j $(0 \le j \le m)$ and t_k $(0 \le k \le n)$ are successive end-to-end sections in the dimension of X, Y, and T respectively.

In GMOI, we make a partition in the three-dimensional indexed space and derive a set of Grid blocks, as shown in Figure 3. Each of these Grid blocks contains a pointer leading to a certain Grid bucket in the storage, whose size equals to that of the basic I/O units. What stored in the Grid buckets are the indexed records.

In the process of operation on GMOI, insertion and deletion of records may cause dynamic change of the Grid partition, i.e., splits and merges of Grid blocks. For example, as shown in Figure 3, the broken line divides the section x_2 into two in the x dimension. Dynamic partition of X×Y×T* space and the maintenance of the relation between Grid blocks and buckets are accomplished by managing a Grid directory. In the Grid directory, every item contains the boundary of the block and a pointer leading to the corresponding bucket.

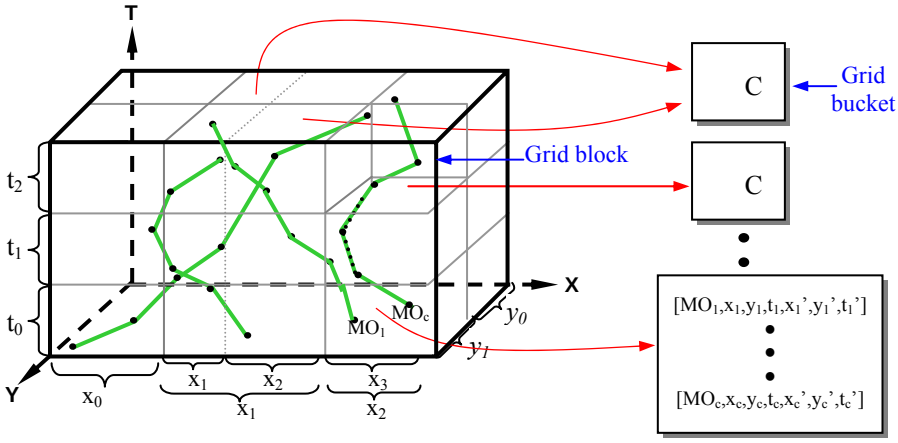


Fig. 3. The Grid partition in three-dimensional spatio-temporal space

In GMOI, the trajectories of moving objects are stored in the forms of trajectory segments. Considering trajectory segment $SEG=[(x_i , y_i , t_i), (x_{i+1} , y_{i+1} , t_{i+1})]$, if it goes through a Grid block, with the pointer in the block, the entire information of *SEG* can be found in the corresponding Grid bucket, including its identifier and the vertex information. Because *SEG* is essentially a line segment in the three-dimensional space, it is quite easy to find out all Grid blocks intersected by *SEG*.

5 Querying Moving Objects in DSTTMOD

In MOD, most queries fall into the following three categories: point query, range query and k-nearest neighbor (KNN) query. In this section, we present the processing approaches of these queries based on GMOI.

5.1 Point Queries

Point queries search for the location of a certain moving object at a given time instant, e.g., "find the location of MO_q at t_q". When processing such a query, the system can get the result directly from the trajectory information without searching the index. Suppose that the starting and ending points of the corresponding trajectory segment is

(x_0 , y_0 , t_0) and (x_1 , y_1 , t_1) ($t_0 \le t_q \le t_1$). Then the location of MO_q at time instant t_q, denoted by (x_q , y_q), can be computed by the following formula:

$$\begin{cases} xq = \varepsilon \times (x1 - x0) + x0 \\ yq = \varepsilon \times (y1 - y0) + y0 \end{cases}, \text{ in which } \varepsilon = \frac{tq - t0}{t1 - t0}$$

5.2 Range Queries

Range queries search moving objects that cross a given geographic region in a given time period, e.g., "find all the objects passing the region *A* in the next 10 minutes". Such a query returns objects whose trajectory goes through a querying box in X×Y×T space, i.e. the cubic range covers a geographic region and a time period. When processing such queries, first the Grid directory is looked up to find all the Grid blocks that has common part with the querying box. Then for each block, find the corresponding Grid bucket and retrieve the desired records.

For Grid blocks which are totally contained in the querying box, the records in the corresponding bucket are sure to be desired and can be output directly. For other cases, further calculation is performed to determine whether the trajectory segment stored in the bucket is really in the querying box. For example, suppose *G* is a Grid block intersecting with the querying box *Q*. *Q* is determined by three sections: [qx_1 , qx_2] in x-direction, [qy_1 , qy_2] in y-direction, and [qt_1 , qt_2] in t-direction. Assume [*MO*, x_0 , y_0 , t_0 , x_1 , y_1 , t_1] is one of the records in the corresponding bucket. It represents a trajectory segment [(x_0 , y_0 , t_0), (x_1 , y_1 , t_1)] of *MO*. Through some simple mathematical computation, the two intersecting points of *G* with [(x_0 , y_0 , t_0), (x_1 , y_1 , t_1)] can be figured out, denoted by (x_m , y_m , t_m) and (x_n , y_n , t_n). If [qx_1 , qx_2] intersects with [x_m , x_n], [q_{y1} , q_{y2}] intersects with [y_m , y_n], and [qt_1 , qt_2] intersects with [t_m , t_n] simultaneously, *MO* is output. Otherwise, *MO* is ignored.

5.3 K-Nearest Neighbor Queries

An example k-nearest neighbor query is "find *k* moving objects which are closest to (x_q , y_q) around time t_q". Different from range queries, KNN queries are evaluated by incremental searching. Originally the searching box is a small circular range surrounding the given point, and then gradually expanded by extending the radius until *k* nearest objects are found. That is, at first, find the moving objects in the searching box with (x_q , y_q , t_q) as the center. If the number of the found objects $m>k$, select k nearest ones from them as the result. If *m* is equal to *k*, just output these objects. Otherwise, i.e., $m<k$, extend the searching radius to find the remaining *k-m* objects. The increase of searching radius follows a kind of strategy, such as linear increase. Because some Grid blocks have been checked in last round, the remaining *k-m* objects are searched only in the Grid blocks which are involved because of the expansion of the searching radius.

6 Performance Evaluation

In order to evaluate the performance of the proposed data structures and algorithms, we have developed a prototype system. Based on the system, we have carried out a series of performance experiments to compare the Grid-File based Moving Objects Indexing (GMOI) method with other indexing methods. Since the GMOI method is a HDD-based solution, we choose R-tree based Moving Objects Indexing (RMOI) method [6], which has the same feature, as the control of the experiments. Table 1 summarizes the parameters used in the experiments.

Table 1. Main parameters in the experiment

Parameter	***Value***	***Meaning***
A	*1000*1000*1000*	size of indexed space (X×Y×T*)
M	*50-300*	number of moving objects
N	*32*	number of moving objects generated/expired per second
C	(*50, 25*), (*25, 12*)	maximal/minimal number of records in one data block of R-tree
U	*500*	frequency of location updates for moving objects
R	*100*100*10*	size of querying box in range queries
T	*2000*	number of range query issued per second
S	*10*	average number of the segments in each Spatio-Temporal trajectory
W	*500*	average size of active window with moving plans

6.1 Index-Create Performance

As shown in Figure 4(a), when the number of moving objects is small, it will take more time for Grid-File to create the index structure than for the R-Tree method. This is because the inserting of the Grid records will lead to the Grid block's splitting and merging based on the actual state. However, with the number of the moving objects increasing, the R-tree's increment of the creating time will be larger than that of the Grid-File, and will finally exceed Grid-File.

From Figure 4(b), we can see that the storage consumption for Grid-File is greater than for R-Tree. In order to implement the flat structure of the Grid bucket and speed up querying and updating, a number of navigation pointers are used in the nodes of the Grid-File. On the contrary, there is no redundancy in R-Tree. The nodes of R-Tree only store location information.

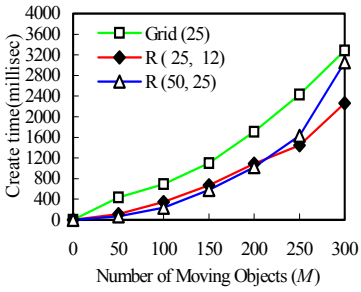
6.2 Update Performance

In this experiment, we compare GMOI and RMOI in terms of the time of processing *U* (*U=500*) update operations for the objects.

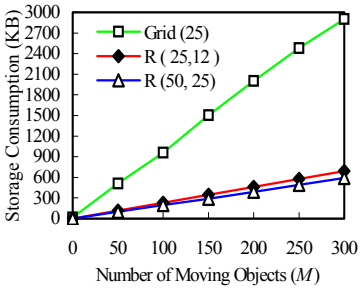
As shown in Figure 4(c), there is an obvious difference between the Grid-File and R-Tree in location updating performance. The reason is that the R-Tree has a nested structure, when updating, it will lead to nested merges and splits of the tree node and nested adjustment of MBR's because of the delete and insert operations. On the contrary, the Grid-File method has a flat structure, which will not lead to such actions. At the same time, the choice of splitting point is flexible in Grid-File method. So the latter will cost much less in terms of location updating.

6.3 Query Performance

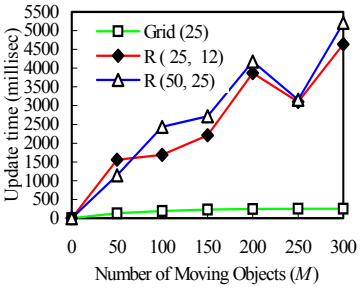
Figure 4(d) shows the time elapsed from the issuing of a query to the returning of the result. We can see that Grid-File has a little bit better query performance than R-Tree. This is because the scope overlap of the R-Tree's nodes will be more serious with the number of the moving object increasing, and in most cases it will be necessary to look through the whole tree structure to find the objects that match the query. All the above will not occur while using Grid-File indexing method.



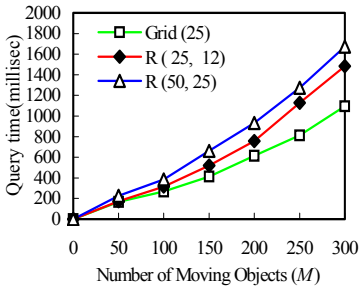
(a) Creating Time Comparison



(b) Storage Consumption Comparison



(c) Updating time Comparison



(d) Querying time Comparison

Fig. 4. Performance comparisons between GMOI and RMOI

7 Conclusion

The key research issue with moving objects databases (MOD) is the modeling of location information. In this paper, we put forward a new MOD model - Discrete Spatio-Temporal Trajectory Based Moving Objects Database (DSTTMOD) model. Our aim is to support queries for location information not only in the past and at present, but also in the future (including the near and far future). To obtain this goal, we use a trajectory based method to represent moving objects and a Grid-file based indexing method to organize these trajectories. We have also developed a prototype system and conducted a series of experiments which show promising performance results.

Acknowledgements

This research was partially supported by the grants from 863 High Technology Foundation of China (2002AA116030), the Natural Science Foundation of China (60073014, 60273018), the Key Project of Chinese Ministry of Education (03044) and the Excellent Young Teachers Program of MOE, P.R.C (EYTP). We would also like to thank Yun Bai, Chen Wu and Xiaoqing Wu of Renmin University of China for their valuable help in the detailed implementation of the prototype system.

References

[1] Wolfson O., Xu B., Chamberlain S., Jiang L., Moving Object Databases: Issues and Solutions. *Proceedings of the 10th International Conference on Science and Statistical Database Management*, Capri, Italy, July 1998:111-122.

[2] Wolfson O., Chamberlain S., Dao S., Jiang L., Location Management in Moving Objects Databases. In *Proceedings of WOSBIS'97*, Budapest, Hungary, Oct. 1997.

[3] Sistla A.P., Wolfson O., Chamberlian S., Dao S.. Modeling and querying Moving Objects. In: *Proceedings of ICDE* 1997.

[4] Wolfson O., Chamberlain S., Dao S., Jiang L., Mendez G. Cost and Imprecision in Modeling the Position of Moving Objects. In: *Proceedings of ICDE* 1998.

[5] Forlizzi L., Guting R. H., Nardelli E., Schneider M. A Data Model and Data Structures for Moving Objects Databases. *In Proc. of ACM SIGMOD 2000,* Texas, USA, 2000.

[6] Chon H. D., Agrawal D., Abbadi A. E. Query Processing for Moving Objects with Space-Time Grid Storage Model. *In Proceedings of the 3rd International Conference on Mobile Data Management (MDM2002).* Hong Kong. Jan. 2001.

[7] Pfoser D., Jensen C. S., Theodoridis Y. Novel Approach to the Indexing of Moving Object Trajectories. *Proceedings of the 26th VLDB*, Cairo, Egypt, 2000.

[8] Saltenis S., Jensen C. S., Leutenegger S. T., Lopez M. A. Indexing the Position of Continuously Moving Objects. *In Proc. of ACM SIGMOD 2000,* Dallas, TX, USA, 2000.

[9] J. Nievergelt, H. Hinterberger, and K. C. Sevcik, The grid file: An adaptable, symmetric multikey file structure, *ACM Trans. on Database Sys.* 9(1), 1984:38-71.