Information Technology Journal 7 (4): 607-614, 2008
ISSN 1812-5638
© 2008 Asian Network for Scientific Information

Constraint Rectangle: A Novel Approach to Modeling Continuously Moving Objects

Peiquan Jin, Xu Zhang, Shouhong Wan and Lihua Yue
Department of Computer Science and Technology, University of Sciences and Technology of China, Jinzhai Road 96#, Hefei, 230027, China

Abstract: This research focuses on the issue of modeling continuously moving regions and a new approach called constraint rectangle is presented. According to the new approach, a continuously moving region is represented as a set of constraint rectangles and both x-dimension and y-dimension of each constraint rectangle are functions of time. We develop algorithms to represent a continuously moving region into constraint rectangles and prove the correctness of the algorithms. Furthermore, a prototype is implemented to demonstrate the rightness and effectiveness of the new approach. The experiment results show the constraint rectangle model is not only able to represent continuously moving regions effectively and practically, but also able to support historical and future queries on continuously moving regions.

Key words: Spatiotemporal database, moving objects database, continuously moving region, model

INTRODUCTION

Spatial objects that move or change their locations or shapes over time are often referred to as moving objects, e.g., moving cars, storms, forest fires and oil spills. Generally, moving objects can be cataloged as two types: moving points and moving regions. Moving regions are those moving objects that not only the locations but also the boundaries may change with time, e.g., storms, oil spills, etc. Most of moving regions in real applications change their locations and extents continuously, while others in a discrete way. Though lots of works have been done to model discretely moving regions (Pelekis *et al*., 2004), no satisfied models have been proposed so far for continuously moving regions. On the other hand, many applications show urgent requirements on the management of continuously moving regions. For example, in weather forecast, users need to store and trace the moving phenomenon of storms. Another example is forest fire monitoring, in which the area of forest fire should be represented and traced effectively. So, from a database viewpoint, it is an urgent and critical issue to create effective representation for continuously moving regions in order to efficiently store and query them in database.

A lot of approaches have been proposed to model continuously moving objects. Many of recent researches focus on the continuous trajectories of moving objects (Chen and Meng, 2007; Civilis *et al*., 2005; Meng and Ding, 2003; Trajcevski *et al*., 2004), among which the most famous model is the Moving Objects Spatio-Temporal (MOST) model (Sistla *et al*., 1997; Vazirgiannis and Wolfson, 2001). However, it can only model continuously moving points and fails to deal with the continuous changes of extents. Constraint databases were introduced to represent continuously moving regions recently (Cai *et al*., 2000; Chomicki and Revesz, 1999a, b). The model proposed by Chomicki and Revesz (1999a) was called parametric 2-Spaghetti model, which used triangles to represent the snapshots of moving regions. It represented a moving region as a set of triangles whose vertices were linear functions of time. However, it had been proved that this model was not close on algebraic operations. Chomicki and Revesz (1999b) proposed a framework in which all spatiotemporal objects were described as collections of atomic geometric objects. Each of these objects was given as a spatial object and a function describing its development over time. Inherited from the model of Cai *et al.* (2000) presented a more detailed model called Parametric Rectangle, which represented continuously moving regions into a set of parametric rectangles. This model required that the moving regions should be raster data, thus it was not suitable for most GIS-related applications. Besides, how to map between two observations of a moving region still remained unsolved in this model. The data-type-based spatiotemporal data model presents a new way to model continuously moving regions (Güting *et al*., 2000; Tøssebro and Güting, 2001). Differing from previous models, it uses spatiotemporal data types for continuously moving regions, e.g., mregion: region→time and each vertex of a moving region is described as a linear function of time. Tøssebro and Güting (2001) describes the detailed process named sliced representation to build mapping relationship between the snapshots of continuously moving regions. But in real applications it is

Corresponding Author: Peiquan Jin, Department of Computer Science and Technology, University of Sciences and Technology of China, Jinzhai Road 96#, Hefei, 230027, China

difficult to determine functions of each vertex of a moving region. Also, mapping an octagon into a triangle is unconvincing. The data-type-based approach is also adopted by some other work to model continuously moving objects, such as the models proposed by Praing and Schneider (2007).

In this research, we use a set of so-called constraint rectangles to represent a continuously moving region. Both x-dimension and y-dimension of each constraint rectangle are functions of time. Compared with previous approaches, our model is more effective and practical in modeling moving regions. In addition, our discussion is based on the following preliminary conditions:

- The regions are of vector-based format, not raster data. Nowadays, vector data is the most common format used in GIS applications.
- The paper only discusses the changes of simple polygons. The modeling of complex moving regions which have holes in their extents will be studied in the future.
- The observation of moving regions is discrete. That means users can only observe different snapshots of moving regions in different time instants. This is true for most applications. For example, in forest fire or storm management, the states of forest fire or storm are usually obtained by some satellite images, which are taken in discrete time instants.

DEFINITION OF CONSTRAINT RECTANGLE

The new modeling approach is based on a new concept, which is named constraint rectangle. A constraint rectangle represents moving 2-dimensional regions using constraint expressions in both x and y dimensions.

Definition 1 constraint rectangle: A constraint rectangle is a quintuple:

$$<x^l, x^r, y^b, y^t, p>$$

where, p is a period of time and x^l, x^r, y^b and y^t are all linear functions of time t applicable when t∊p.

A constraint rectangle represents the continuous changes of a rectangle in the period p. Given any t∊p, the state of the constraint rectangle at the instant t is determined by the constraint expression $x^l(t)\le x\le x^r(t)\wedge y^b(t)\le y\le y^t(t)$. Figure 1 shows two states of the constraint rectangle (t, 2t+2, t, t+2, [1, 5]). For simplicity, we use integers to represent time instants.

Having introduced the concept of constraint rectangle, we can further define a continuously moving region.

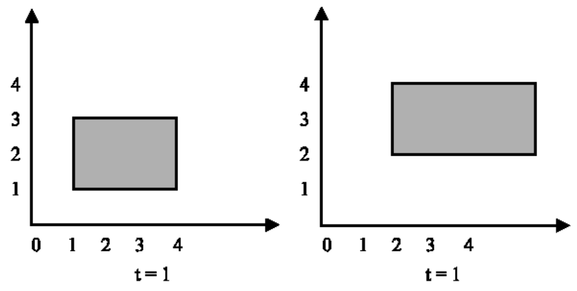


Fig. 1: An example of constraint rectangles

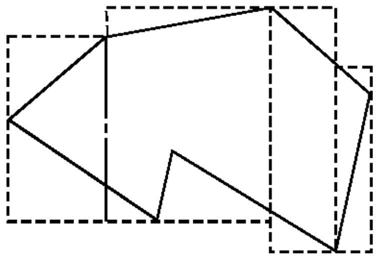


Fig. 2: Representing a continuously moving region using four constraint rectangles

Definition 2 continuously moving region: A continuously moving region is a set of non-overlapping constraint rectangles, each of which represents the continuous changes of the MBR (Minimal Bounding Rectangle) of one part of the region.

Figure 2 shows the representations of continuously moving regions.

REPRESENTING A CONTINUOUSLY MOVING REGION INTO CONSTRAINT RECTANGLES

Given the initial and final snapshots of a continuously moving region, we can construct the constraint-rectangle-based representations to trace the motion of the region. The algorithm is separated into three steps: the first step is to construct bounding rectangles of each snapshot; the second is to find mapping pairs between the rectangles of two snapshots and the final step is to interpolate between the two snapshots and produce the final representation.

Construct bounding rectangles based on grid partition: The first algorithm ConstructBoundingRects produces the set of bounding rectangles for a given region, which refers to a snapshot of a continuously moving region. We use a grid partition method to construct the bounding rectangles, as shown in Fig. 3.

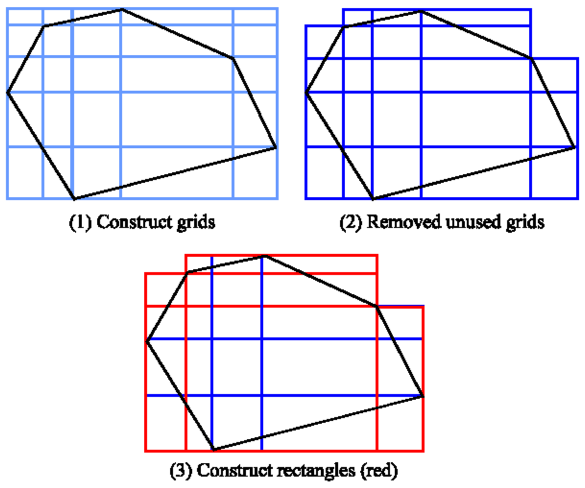


Fig. 3: An example of algorithm ConstructBoundingRects

The algorithm consists of three main steps:

- For each vertex of the region, drawing horizontal and vertical lines which pass through the vertex
- Removing blank rectangles from the grids
- Merging the neighbored grids into rectangles

```
Algorithm 1: ConstructBoundingRects ($R_1$)
Input: Region $R_1$
Output: RR, the set of bounding rectangles of $R_1$
Preconditions: The points in $R_1(p_1,p_2,\ldots,p_n)$ are in a counter-clockwise direction
Begin
1   For each $p \in R_1$ Do // make grids
2       Draw a horizontal line and a vertical line and both pass through p
3   End For
4   For each grid G generated in step 1 to 3 Do
5       If G is not contained in $R_1$ and does not intersect with $R_1$ Then
6           Remove G from the set of grids
7       End If
8   End For
9   Merge the grids into rectangles along the vertical direction
10  Merge the rectangles generated in Step 9 into bigger rectangles along the horizontal direction
11  RR←{rectangles generated in Step 10};
12  Return RR;
End
```

Find mapping rectangles between snapshots: After constructing bounding rectangles for both snapshots, the second step is to find the mapping pairs of rectangles between the two snapshots. The problem can be defined as follows: For a given rectangle r in the initial snapshot, finding the most appropriate rectangles in the destination that can reflect the change of the r.

In this research, we use a new approach to solve this problem, which is based on space overlap and clustering. The key idea is to first overlap the two snapshots to let their central points in the same position and then to cluster the central points of each rectangles in the initial snapshot with those in the second snapshot, that refers to find nearest neighbors for each rectangle in the initial snapshot. Figure 4 shows the example of such idea.

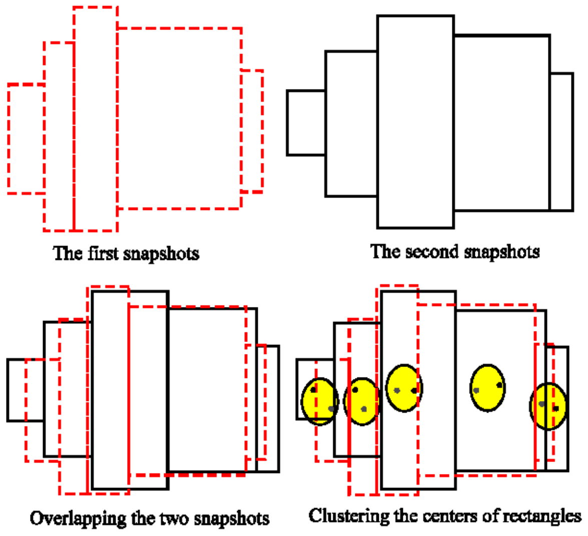

The first snapshots
The second snapshots
Overlapping the two snapshots
Clustering the centers of rectangles

Fig. 4: Finding mapping rectangles between snapshots using space overlap and clustering

The formal algorithm FindMatchingRects is to map the rectangles of the first snapshot with the rectangles in the second snapshot, which is described as follows.

```
Algorithm 2: FindMatchingRects ($R_1$, $R_2$)
Input: Two sets of rectangles: $R_1 = \{A_1, A_2,\ldots,A_m\}$ and $R_2 = \{B_1, B_2,\ldots,B_n\}$
Output: The matching result T. T is a set of point sets, whose elements are point sets indicating the mapping results between the central point of each rectangle with the central points of other mapped rectangles.
Preconditions: Both $R_1$ and $R_2$ are generated by the algorithm ConstructBoundingRects. T is null.
Begin
// to make $R_2$ aligned with $R_1$ in the same center
1   Overlapping($R_1$, $R_2$);
//computing the central positions
//of each rectangles in $R_1$ and $R_2$
2   $CP_1 = \{a_i | a_i = \text{center}(A_i) \wedge i \in [1, m]\}$;
3   $CP_2 = \{b_i | b_i = \text{center}(B_i) \wedge i \in [1, n]\}$;
//for each point in $CP_1$,
//find the nearest point in $CP_2$
4   $NN_1 = \{(a_i, p_i) | a_i \in CP_1 \wedge p_i \in CP_2 \wedge p_i = \text{nearest}(a_i, CP_2) \wedge i \in [1, m]\}$;
// for each point in $CP_2$,
// find the nearest point in $CP_1$
5   $NN_2 = \{(b_i, q_i) | b_i \in CP_2 \wedge q_i \in CP_1 \wedge q_i = \text{nearest}(b_i, CP_2) \wedge i \in [1, n]\}$;
6   T←$NN_1$; //initialize T
7   For each $b_i \in CP^2$ Do
        //processing the isolated points in $CP_2$, i.e., not appear in $NN_1$
8       If $b_i \notin \{p | (a_j,p) \in NN_1 \wedge j \in [1,m]\}$ Then
            //find the $(b_i,a_k)$ in $NN_2$. Hence in $NN_1$ there is corresponding $(a_k,p_k)$
9           $a_k$ = find ($b_i$,$NN_2$);
10          If $p_k \notin NN_1 - (a_k,p_k)$ Then
                //add $b_i$ into the $(a_k,p_k)$ in T
11              remove(T,$(a_k,p_k,\ldots)$);
12              add(T,$(a_k,p_k,b_i,\ldots)$);
```

```
13        Else // $p_k$ has mapping points in $CP_1$
              //first remove $p_k$, then add $b_i$
14          remove(T,($a_k$,$p_k$));
15          add(T,($a_k$,$b_i$));
16        End If
17      End If
18  End For
//merge the pairs in T that have same mapping points
19  If $\exists((a_i,b_k)\in T \wedge (a_j,b_k)\in T)$ Then
20     remove(T,($a_i$,$b_k$));
21     remove(T,($a_j$,$b_k$));
22     add(T,($a_i$,$a_j$,$b_k$));
23  End If
24  Return T;
    End
```

The correctness of the algorithm FindMatchingRects is supported by the following theorems.

Theorem 1 (Completeness): Given two sets of bounding rectangles R_1 and R_2, which are generated by algorithm ConstructBoundingRects, each rectangle in R_1 will find corresponding mapping rectangles in R_2 through the algorithm FindmatchingRects and vice versa.

Proof:

- Since CP_1, the set of central points of R_1, has unique correspondence with R_1, so we only need to prove that each point in CP_1 will finally appear in the matching result T.
- Since each point in CP_1 has corresponding pairs in NN_1 and at the beginning $T = NN_1$, so initially each point in CP_1 appears in T. In addition, in the next steps, the only update to T is to change $(a_k,p_k,\ldots)$ into $(a_k,p_k,b_i,\ldots)$ or to merge (a_i,p_k) and (a_j,p_k) into (a_i,a_j,p_k) and both do not remove any a_k from T, so each point in CP_1 appears in T, which means each rectangle in R_1 has matching rectangles in R_2.
- In order to prove that each rectangle in R_2 has corresponding match in R_1, we only need to prove the each point in CP_2 appears in T. For each point in CP_2, there are two cases:
  - It has been matched in NN_1. Since initially $T = NN_1$ and the update on T does not change the set of b_i appears in T, it will finally appear in T.
  - It is not contained in NN_1. Then it will be added into T from Step 7 to Step 17. So it will also appear in T finally.

Hence, each point in CP_2 appears in T, that means each rectangle in R_2 has corresponding matching rectangles in R_1.

From (1-3), we can draw the conclusion that each rectangle in R_1 has corresponding mapping rectangles in R_2 through the algorithm FindmatchingRects and vice versa.

Theorem 2 (Mutual exclusion): Given two sets of bounding rectangles R_1 and R_2, which are generated by algorithm ConstructBoundingRects, each rectangle in R_1 will only appear in one mapping pair in the final matching result T, which is produced by the algorithm FindmatchingRects and vice versa.

Proof:

- Since initially $T = NN_1$ and in NN_1 each point of CP_1 has only one pair like (a_k, p_k). Besides, in the next steps, the only update to T is to change $(a_k,p_k,\ldots)$ into $(a_k,p_k,b_i,\ldots)$ or to merge (a_i,p_k) and (a_j,p_k) into (a_i,a_j,p_k) and both do not place an a_k into more than two pairs. So finally each a_k appears in only one pair in T, which means each rectangle in R_1 only has one matching pair in T.
- We can also prove that it is impossible for one point in CP_2 to appear in more than two pairs in T. Suppose it is true, that means there exists (a_i,b_k) and (a_j,b_k) in T. The only update to T is (i) to add b_i into $(a_k,p_k,\ldots)$, or (ii) to merge (a_i,p_k) and (a_j,p_k) into (a_i,a_j,p_k).

If (i) occurs, then in Step 10 to Step 12 the duplicate appearance of b_k is eliminated. If (a) does not occur, then (ii) must occur and the duplicate occurrence of b_k is also eliminated in Step 13 to 15.

So, each point in CP_2 only appears in one pair in T, which means each rectangle in R_2 has only one matching way with rectangles in R_1.

From (1) to (2), we can draw the conclusion that each rectangle in R_1 will only appear in one mapping pair in the final matching result and vice versa.

Create representation for continuously moving regions: The algorithm MakeConstraintRects generates the representation of a continuously moving region from two snapshots.

Algorithm 3: MakeConstraintRects (R_1,R_2,t_1,t_2)
Input: Two snapshots of region: The initial snapshot R_1 at t_1, the final snapshot R_2 at t_2
Output: CR, the set of constraint rectangles in the period $[t_1, t_2]$
Preconditions: $t_2 > t_1$; the vertices of $R_1(p_1,p_2,p_3,p_4)$ are in a counter-clockwise direction.

```
Begin
1  $RR_1$=ConstructBoundingRects($R_1$);
2  $RR_2$=ConstructBoundingRects($R_2$);
3  T = FindMatchingRects($RR_1$, $RR_2$);
4  For each t in T Do
5     If $t = (a_i,b_j,b_{j+1},b_{j+k})$ Then
6        R' ←dividing $RR_1$[i] into k rectangles;
7        $RR_1$←($RR_1$-$RR_1$[i]) ∪ R';
```

```
8        **ElseIf** $t = (a_i, a_{i+1}, a_{i+k}, b_j)$ **Then**
9          R' ← dividing $RR_2[j]$ into k rectangles;
10         $RR_2 \leftarrow (RR_2 - RR_1[j]) \cup R'$;
11     **End If**
12     **End For**
13     **For** i = 1 to $RR_1$.count **Do**
         //min. and max. of x and y dimension
14     $x_1 = RR_1[i]$.min_x; $x_1' = RR_1[i]$.max_x;
15     $y_1 = RR_1[i]$.min_y; $y_1' = RR_1[i]$.max_y;
16     $x_2 = RR_2[i]$.min_x; $x_2' = RR_2[i]$.max_x;
17     $y_2 = RR_2[i]$.min_y; $y_2' = RR_2[i]$.max_y;
18     $x_l \leftarrow \frac{x_2 - x_1}{t_2 - t_1} \cdot t + \frac{t_2 x_1 - t_1 x_2}{t_2 - t_1}$
19     $x_r \leftarrow \frac{x_2' - x_1'}{t_2 - t_1} \cdot t + \frac{t_2 x_1' - t_1 x_2'}{t_2 - t_1}$
20     $y_b \leftarrow \frac{y_2 - y_1}{t_2 - t_1} \cdot t + \frac{t_2 y_1 - t_1 y_2}{t_2 - t_1}$;
21     $y_t \leftarrow \frac{y_2' - y_1'}{t_2 - t_1} \cdot t + \frac{t_2 y_1' - t_1 y_2'}{t_2 - t_1}$;
22     $CR[i] \leftarrow (x_l, x_r, y_b, y_t, [t_1, t_2])$ ;
23     **End For**
24     **Return** CR;
**End**
```

The algorithm Make Constraint Rects first constructs bounding rectangles for each snapshot and then maps between bounding rectangles. The mapping strategy is described in algorithm FindMatchingRects. The simplest situation is to map one rectangle in the first snapshot into one rectangle in the second snapshot. However, since the number of rectangles in the initial snapshot may be different from that in the second snapshot, the algorithm FindMatchingRects will map one rectangle into several rectangles or map several rectangles into one rectangle. Thus we need divide the rectangle into several rectangles in order to generate constraint rectangles, as shown in Step 6 and Step 9 of the algorithm MakeConstraintRects. In order to map one rectangle in the first snapshot into k rectangles in the second snapshot, we refer the lengths of x dimensions of the k rectangles in the second snapshot. For example, in Fig. 5, as the rectangle noted as b is mapped into rectangle d and e, then because the lengths of x dimensions of the rectangle d and e are 4 and 1, respectively, so we divide the rectangle b into two rectangles according to the proportion 4:1, which is denoted as a vertical dotted line in the figure.

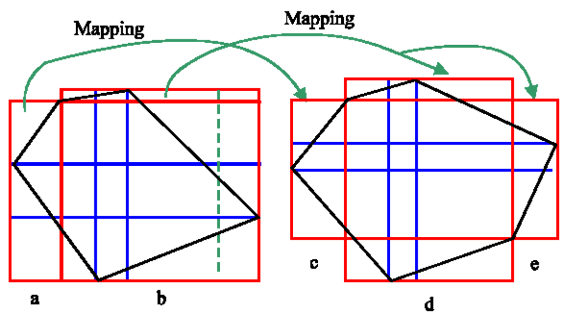


Fig. 5: Mapping the rectangles between the two snapshots

EXPERIMENT RESULTS

We build a prototype to test our approach, as shown in Fig. 6.

The prototype is implemented with C# language under Windows XP Professional. The inputted dataset is two data files, each of which contains a list of points and represents a snapshot of a continuously moving region. The timestamp unit of the snapshots is set to day.

After reading all points, first we draw the two snapshots on the screen. And then, for all vertices of the snapshot, a horizontal straight line and a vertical straight line are drawn according to the algorithm. The area of each snapshot is turned into a set of grids (as shown in Fig. 7a). Then the partitioned grids are merged into bounding rectangles. The result is shown in Fig. 7b.

In most situations, the numbers of generated bounding rectangles of each snapshot are different. As described in the algorithm FindMatchingRects, we use space overlapping and clustering to create the mapping relationships between the initial bounding rectangles with the second ones. Figure 7c shows the result of the algorithm FindMatchingRects, in which the two rectangle sets contain the same number of rectangles.

After creating constraint-rectangle-based representation for a continuously moving region, we can query the historical states or predict the future states of the continuously moving region. For example, assume the date of the first snapshot is Oct.2, 2007 and that of the second snapshot is Oct.5, 2007, the historical state of the moving region at Oct.3, 2007 can be generated easily, which is also a set of bounding rectangles (as shown in the right-bottom of Fig. 8).

Previous modeling techniques for continuously moving regions all rely on the observations on continuously moving regions. Since any discrete observations can not express the precise moving properties and most of applications related to continuously moving regions mainly need the prediction of moving states of regions, so it is reasonable to introduce some approximation into the representation of continuously moving regions to make it more suitable for applications. To test the applicability of the constraint-rectangle-based approach, we construct the constraint-rectangle-based representations based on two given snapshots of a continuously moving region and compute the boundary vector of the moving region at a specific instant. As shown in Fig. 8, the shape of the generated

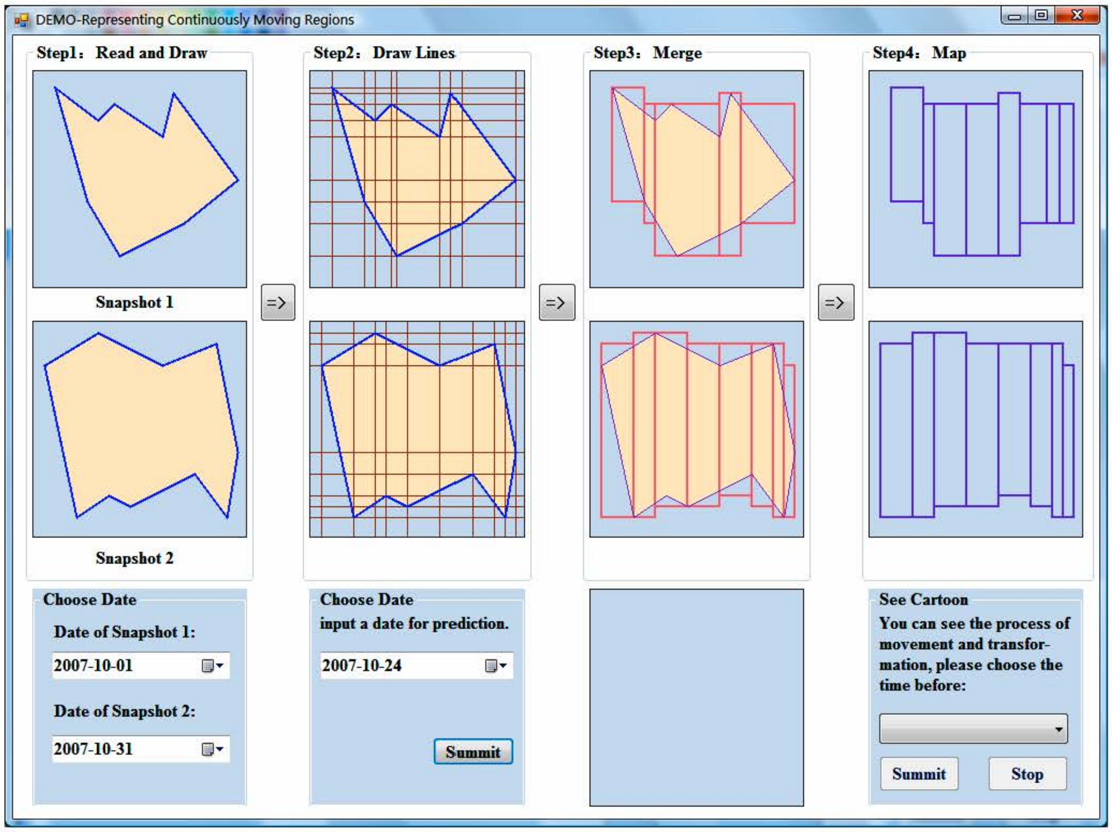


Fig. 6: The interface of the prototype

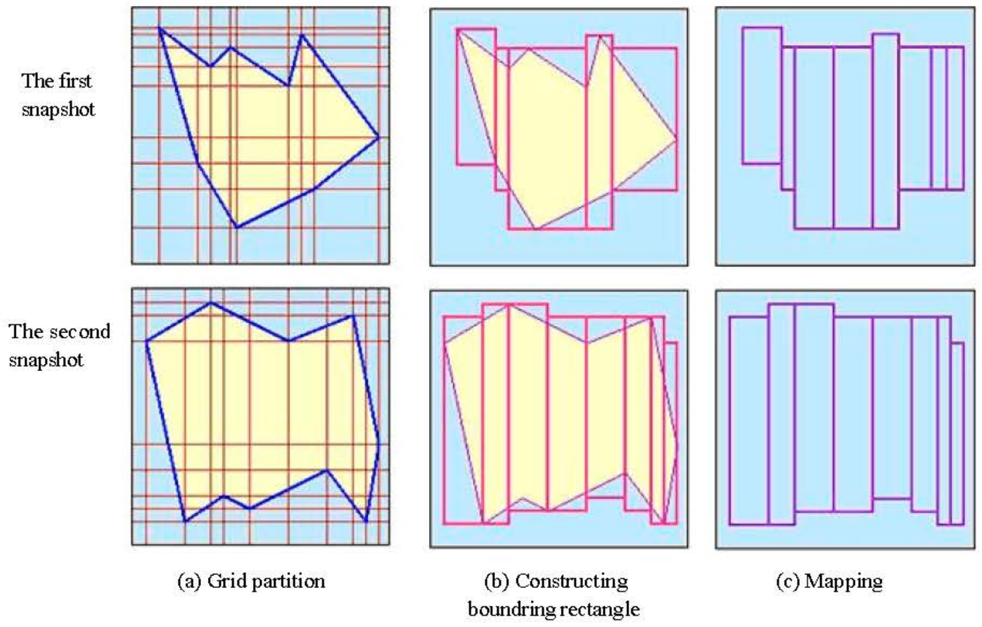
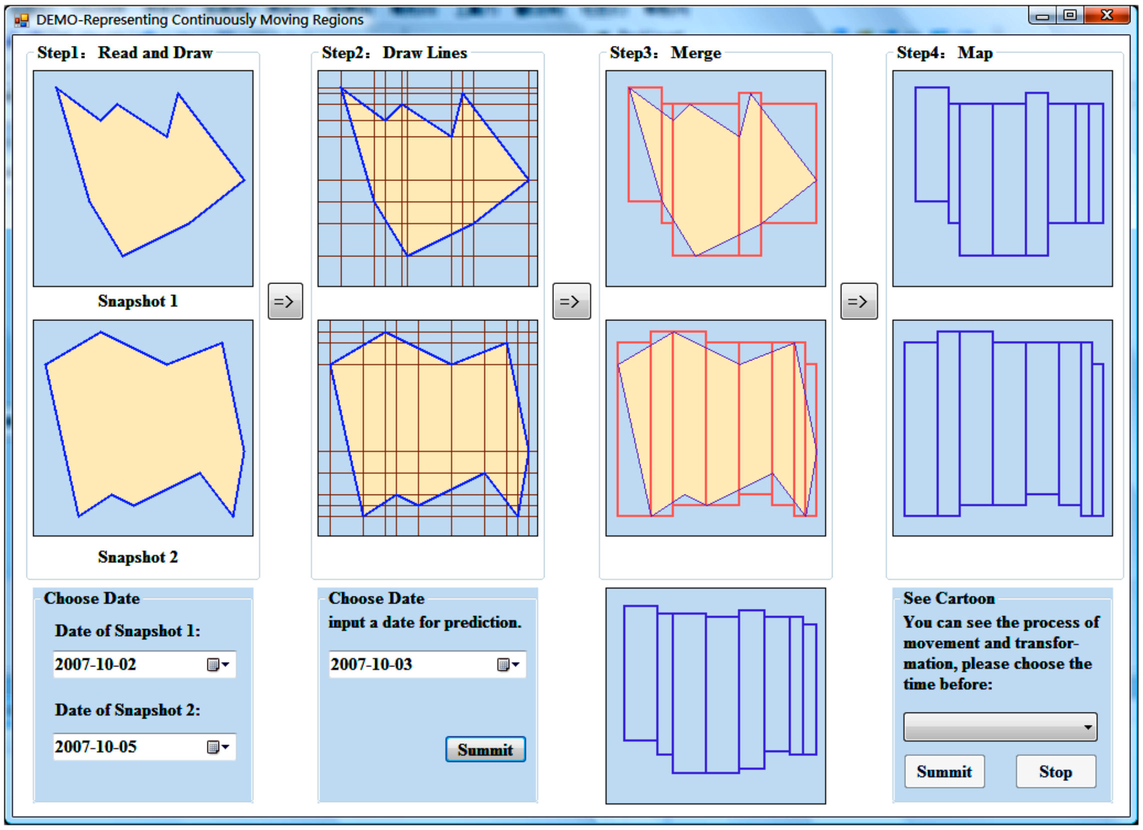


Fig. 7: Grid partition, constructing bounding rectangles and mapping



Date of Snapshot 1: 2007-10-02

Date of Snapshot 2: 2007-10-05

input a date for prediction. 2007-10-03

You can see the process of movement and transformation, please choose the time before:

Fig. 8: Querying the historical states of a continuously moving region

region is regular, that is, each border is either horizontal or vertical. However, in practical applications, the modeling of continuously changes has to rely on predictions, so the predicted shape does not make sense, the only thing we should know is the location and extent of the moving region. From the generated regular region at instant *t*, it is very convenient to compute the location and extent of the moving region at the instant.

CONCLUSIONS

The representation and modeling of continuous moving objects is one of the critical issues in moving object databases. We have presented in this paper a novel approach for the representations of continuously moving regions, which is called constraint rectangle. The new approach constructs a set of constraint rectangles from two snapshots of a moving region and can produce future or historical states of moving regions. The algorithms to construct bounding rectangles and further to construct constraint-rectangle-based representation are described, as well as the prototype implementation. Compared with previous solutions, our approach is more convenient and practicable for such applications as storm forecast and forest fire monitoring that mainly require prediction on continuously moving objects or phenomenon.

However, this research does not deal with the uncertainty issue of moving regions, as well as the modeling of complex moving regions. Our future work will focus on the uncertainty modeling of moving regions and the modeling of complex moving regions.

ACKNOWLEDGMENT

This research is supported by the National Science Foundation of China (No. 60776801).

REFERENCES

Cai, M., D. Keshwani and P. Revesz, 2000. Parametric Rectangles: A Model for Querying and Animation of Spatiotemporal Databases. In: Proceedings of 7th International Conference on Extending Database Technology (EDBT). Zaniolo, C., P.C. Lockemann and M.H. Scholl *et al.* (Eds.). LNCS 1777. Konstanz, Germany.

Chen, J. and X. Meng, 2007. Indexing future trajectories of moving objects in a constrained network. J. Comput. Sci. Technol., 22 (2): 245-251.

Chomicki, J. and P. Revesz, 1999a. Constraint-based interoperability of spatiotemporal databases. GeoInformatica, 3 (3): 211-243.

Chomicki, J. and P. Revesz, 1999b. A Geometric Framework for Specifying Spatiotemporal Objects. In: Proceedings of 6th International Workshop on Temporal Representation and Reasoning (TIME). IEEE CS Press, 1999, Orlando, Florida, USA.

Civilis, A., C.S. Jensen and S. Pakalnis, 2005. Techniques for efficient road-network-based tracking of moving objects. IEEE T Knowl. Data En., 17 (5): 698-712.

Güting, R., M. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider and M. Vazirgiannis, 2000. A foundation for representing and querying moving objects. ACM T. Database Syst., 25 (1): 1-42.

Meng, X. and Z. Ding, 2003. DSTTMOD: A Future Trajectory Based Moving Objects Database System. In: Proceedings of 14th International Conference on Database and Expert Systems Applications (DEXA), LNCS 2736, September 2003, Marík, V., W. Retschitzegger and O. Stepánková (Eds.). Prague, Czech Republic.

Pelekis, N., B. Theodoulidis, I. Kopanakis and Y. Theodoridis, 2004. Literature review of spatio-temporal database models. Knowledge Eng. Rev., 19 (3): 235-274.

Praing, R. and M. Schneider, 2007. Modeling Historical and Future Movements of Spatio-Temporal Objects in Moving Objects Databases. In: Proceedings of ACM 16th Conference on Information and Knowledge Management (CIKM). Silva, M.J. and A.H. Laender *et al.* (Eds.). ACM Press, November, Lisbon, Portugal.

Sistla, A., O. Wolfson, S. Chamberlain and S. Dao, 1997. Modeling and Querying Moving Objects. In: Proceedings of the 13th International Conference on Data Engineering (ICDE). Gray, W.A. and P. Larson, (Eds.). IEEE CS Press, April, Birmingham UK.

Tøssebro, E. and R. Güting, 2001. Creating Representations for Continuously Moving Regions from Observations. In: Proceedings of 7th International Symposium on Spatial and Temporal Databases (SSTD), LNCS 2121, Jensen, C.S., M. Schneider, B. Seeger and V.J. Tsotras (Eds.). Redondo Beach, CA, USA.

Trajcevski, G., O. Wolfson, K. Hinrichs and S. Chamberlain, 2004. Managing Uncertainty in Moving Objects Databases. ACM T. Database Syst., 29 (3): 463-507.

Vazirgiannis, M. and O. Wolfson, 2001. A Spatiotemporal Model and Language for Moving Objects on Road Networks. In: Proceedings of 7th International Symposium on Spatial and Temporal Databases (SSTD), LNCS 2121, Jensen, C.S., M. Schneider, B. Seeger and V.J. Tsotras (Eds.). Redondo Beach, CA, USA.