Constraint Rectangle: A Novel Approach to Modeling Continuously Moving Objects

Peiquan Jin, Xu Zhang, Shouhong Wan and Libua 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 Wollson, 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; Tossebro and Güting, 2001). Differing from previous models, it uses spatiotemporal data types for continuously moving regions, e.g., region-time and each vertex of a moving region is described as a linear function of time. Tossebro 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
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 Fraing 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 called 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: 

\[<x^1, x^2, y^1, y^2, p>\]

where, \(p\) is a period of time and \(x^1, x^2, y^1, y^2\) are all linear functions of time \(t\) applicable when \(t \in p\).

A constraint rectangle represents the continuous changes of a rectangle in the period \(p\). Given any \(t \in p\), the state of the constraint rectangle at the instant \(t\) is determined by the constraint expression \(x^1(t) \leq x \leq x^2(t)\) and \(y^1(t) \leq y \leq y^2(t)\). Figure 1 shows two states of the constraint rectangle \((t, t+2, t, t+2, [1, 5])\). 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

![Constraint Rectangles Example](image)

**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.
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.

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 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: \( R_1 = (A_1, A_2, ..., A_n) \) and \( R_2 = (B_1, B_2, ..., B_m) \)

**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**

1. Overlapping(\( R_1, R_2 \));
2. //computing the central positions
   3. \( CP_1 = \{ (a_i, \text{center}(A_i)) | i \in [1, m] \} \);
   4. \( CP_2 = \{ (b_i, \text{center}(B_i)) | i \in [1, n] \} \);
   5. //for each point in \( CP_1 \)
      6. //find the nearest point in \( CP_2 \)
   7. \( NN_1 = \{ (a_i, p_i, CP_1 \cap CP_2 | p_i = \text{nearest}(a_i, CP_2) \} | i \in [1, m] \} \);
   8. //for each point in \( CP_2 \)
      9. //find the nearest point in \( CP_1 \)
   10. \( NN_2 = \{ (b_i, q_i, CP_2 \cap CP_1 | q_i = \text{nearest}(b_i, CP_1) \} | i \in [1, n] \} \);
   11. \( T = NN_1 ; //initialize T \)
   12. For each \( b_i \in CP_2 \) Do
      13. //processing the isolated points in \( CP_2 \), i.e., not appear in \( NN_1 \)
      14. If \( b_i \in T \) Then
         15. //find the \( (a_i, p_i) \) in \( NN_1 \), Hence in \( NN_1 \), there is corresponding \( (a_i, p_i) \)
         16. \( a_i = \text{find}(b_i, NN_1) \);
         17. If \( a_i = (a_i, p_i) \) Then
            18. add \( b_i \) into the \( (a_i, p_i) \) in \( T \)
      19. remove(T, (a_i, p_i));
      20. add(T, (a_i, p_i, 0, ..., 0));

**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 $CP_{R_1}$, the set of central points of $R_1$, has unique correspondence with $R_2$, so we only need to prove that each point in $CP_{R_1}$ will finally appear in the matching result $T$.
- Since each point in $CP_{R_1}$ has corresponding pairs in $NN_{R_1}$ and at the beginning $T = NN_{R_1}$, so initially each point in $CP_{R_1}$ appears in $T$. In addition, in the next steps, the only update to $T$ is to change $(a_b,p_b,...)$ into $(a_b,p_b)$ or to merge $(a_b,p_b)$ and $(a_t,p_t)$ into $(a_b,p_t)$ and both do not remove any $a_b$ from $T$, so each point in $CP_{R_1}$ 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 $CP_{R_2}$ appears in $T$. For each point in $CP_{R_2}$, there are two cases:
  - It has been matched in $NN_{R_1}$. Since initially $T = NN_{R_1}$ and the update on $T$ does not change the set of $b$, it will finally appear in $T$.
  - It is not contained in $NN_{R_1}$. 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 $CP_{R_1}$ appears in $T$, that means each rectangle in $R_2$ has corresponding mapping rectangles in $R_1$.

From (1-3), we can draw the conclusion that each rectangle in $R_1$ has corresponding mapping rectangles in $R_3$ 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 $T = NN_{R_1}$ and in $NN_{R_1}$ each point of $CP_{R_1}$ has only one pair like $(a_b, p_b)$. Besides, in the next steps, the only update to $T$ is to change $(a_b,p_b,...)$ into $(a_b,p_b)$ or to merge $(a_b,p_b)$ and $(a_t,p_t)$ into $(a_t,p_t)$ and both do not place an $a_b$ into more than two pairs. So finally each $a_b$ 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 $CP_{R_2}$ to appear in more than two pairs in $T$. Suppose it is true, that means there exists $(a_b,b_a)$ and $(a_t,b_a)$ in $T$. The only update to $T$ is (i) to add $b_t$ into $(a_b,p_b,...)$, or (ii) to merge $(a_b,p_b)$ and $(a_t,p_t)$ into $(a_t,p_t)$.

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

So, each point in $CP_{R_2}$ 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 ($R_1,R_2,t_1$)**

**Input:** Two snapshots of region: The initial snapshot $R_1$ at $t_1$, the final snapshot $R_2$ at $t_2$

**Output:** $RR$, the set of constraint rectangles in the period $[t_1, t_2]$

**Preconditions:** $t_1 > t_2$; the vertices of $R_1(p_1, p_2, p_3, p_4)$ are in a counterclockwise direction.

**Begin**

1. $RR = \text{ConstructBoundingRects}(R_1)$
2. $RR = \text{ConstructBoundingRects}(R_2)$
3. $T = \text{FindMatchingRects}(RR_1, RR_2)$
4. For each $t$ in $T$ Do
5.   If $(a_b,b_a)$ Then
6.     $RR_1 \leftarrow \text{dividing} RR_1[i]$ into rectangles;
7.   $RR_1 \leftarrow (RR_1, RR_1[i]) \cup R'$

610
Elseif $t = (a_{ab}, a_{bh}, b_{ab}, b_{bh})$ Then
9 $R' + \text{dividing } RR[i] \text{ into } k \text{ rectangles}$
10 $RR[i] \cup (RR[i], RR[j]) \cup R'$
11 End If
12 End For
13 For $i = 1 \text{ to } RR[i], \text{count Do}$
14 //min. and max. of $x$ and $y$ dimension
15 $x_i = RR[i].[i] \text{ min. } x_i; y_i = RR[i].[i] \text{ max. } y_i$
16 $x_i = RR[i].[i] \text{ min. } x_i; y_i = RR[i].[i] \text{ max. } y_i$
17 $x_i = RR[i].[i] \text{ min. } x_i; y_i = RR[i].[i] \text{ max. } y_i$
18 $x_i = x_i \times x_i + y_i \times y_i - t_x \times x_i - t_y \times y_i + t_x \times t_y$
19 $x_i = x_i \times x_i + y_i \times y_i - t_x \times x_i - t_y \times y_i + t_x \times t_y$
20 $x_i = x_i \times x_i + y_i \times y_i - t_x \times x_i - t_y \times y_i + t_x \times t_y$
21 $x_i = x_i \times x_i + y_i \times y_i - t_x \times x_i - t_y \times y_i + t_x \times t_y$
22 $C[i] \equiv (x_i, y_i, [i], [i], [i], [i])$
23 End For
24 Return CR
End

**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 input 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, we first 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. 5: Mapping the rectangles between the two snapshots](image-url)
Fig. 6: The interface of the prototype

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


613