Available online at www.sciencedirect.com

SciVerse ScienceDirect

AASRI Procedia 1 (2012) 448 – 453

AASRI Procedia

www.elsevier.com/locate/procedia

2012 AASRI Conference on Computational Intelligence and Bioinformatics

Research on Object-orientation-based Open Shape Model in Three-dimensional Space

Wang Yunliang, Gu Weijie *

Changzhou Institute of Mechatronic Technology, Changzhou 213164, China

Abstract

Based on the description model of object-orientation-based direction relation in two-dimensional space, the description mode of object-orientation-based direction relation in three-dimensional space is proposed. The basic idea is that the actual direction region is modeled as an open shape. The computation related to the world boundary of spatial direction region is eliminated, and the processing of the direction predicates is converted into the processing of topological operations between open shapes and closed geometry objects. The algorithms of topological operations between open shapes and closed geometry objects are presented and the theoretical proof for the correctness and completeness of the algorithms is performed.

Keywords: Open shape; Object-oriented-based; Direction relation

1. Introduction

Direction relation, which is frequently used as selection conditions in spatial queries, is an important concept in spatial database [1]. At present, the research on the description model for direction relation is mainly in two-dimensional space, and there is relatively little research on that in three-dimensional space, which is only mentioned that the description model for the object-orientation-based direction relation in two-dimensional space can be extended and discussed in three-dimensional space, but there is no further study in

* Corresponding author. Tel.: 13775205406.
E-mail address: gwjysu@163.com.

doi:10.1016/j.aasri.2012.06.070

the reference [2]. It is very important for a spatial database management system to provide a method for modelling and processing direction queries in three-dimensional space [3]. Therefore, the research on object-orientation-based open shape model in three-dimensional space is significant.

Based on the description model of object-orientation-based direction relation in two-dimensional space, the corresponding description model of in three-dimensional space is proposed. The computation related to the world boundary of spatial direction region is eliminated, and the processing of the direction predicates is converted into the processing of topological operations between open shapes and closed geometry objects. The algorithms of topological operations between open shapes and closed geometry objects are presented and the theoretical proof for the correctness and completeness of the algorithms is performed.

2. Description of object-orientation-based direction relation in three-dimensional space

Any spatial entity in three-dimensional space is a directional n-dimensional false manifold [4]. It can be divided into a number of k-simplex ($k \leq n$) which has lower dimension than the entity or equivalent dimension to it and the connectivity but not overlapping with each other. Spatial entity in three-dimensional space is composed of five basic elements, including 0-simplex, 1-simplex, 2-simplex, and 3-simplex [5]. The corresponding geometries are point, line segment, triangle and tetrahedron respectively, which are called simplex data structures. Simplex data structure is the simplest geometry structure in three-dimensional, which can make the spatial operations easier.

Front, *right* and *above* of the reference object is represented by three orthogonal vectors of $\vec{front}$, $\vec{right}$ and $\vec{above}$. For simplicity, $\vec{r}$, $\vec{f}$ and $\vec{a}$ is used to represent the three vectors of $\vec{front}$, $\vec{right}$ and $\vec{above}$.

Therefore, the space, in which the reference object A is, is divided into $3 \times 9 = 27$ parts by planes $\pi_{r_{\max}}$、$\pi_{r_{\min}}$、$\pi_{f_{\max}}$、$\pi_{f_{\min}}$、$\pi_{a_{\max}}$ and $\pi_{a_{\min}}$ as shown in Fig.1. That is $O_{i,j}$, $i \in$ {EF,RF,ER,RB, EB,LB, EL, LF,SP}, $j \in$ {up,between,down}.

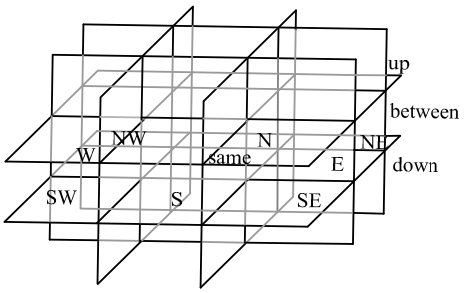

Fig.1. Spaces divided by planes

3. Definition of direction relation predicates

The two direction operations of *reverse* and *isBewteen* are used to define direction predicates. Here, there are four vectors: $\vec{d}$, $\vec{d}_1$, $\vec{d}_2$ and $\vec{d}_3$. Then the *reverse* operation can be defined as: $(-1) \times \vec{d}_1$; The *isBetween* operation can be defined as: $\vec{d}$ *isBetween*($\vec{d}_1$, $\vec{d}_2$, $\vec{d}_3$)is true iff $\exists\ c_1>0, \exists\ c_2>0, \exists\ c_3>0$, s.t. $\vec{d} = c_1 \vec{d}_1 +$

$c_2\vec{d}_2 + c_3\vec{d}_3$. A predicate between target region A and reference region B can be defined in terms of the directional relationship between one point inside A and one or two endpoints of B. $\vec{r}$, $\vec{f}$ and $\vec{a}$ represent the directions of three coordinates axes respectively in the new coordinate system.

Now three kinds of shapes are derived from Fig.1 and are used to represent the 27 parts, as shown in Fig.2.

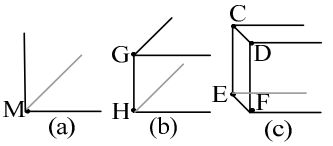

Fig.2. Three kinds of shapes derived

4. Object-orientation-based open shape models

4.1. Definition of open shape

Open shapes refer to the geometries whose boundaries are partially defined. Open shape objects have infinite interiors and extend beyond the boundary of the embedding world. Now we provide the definition of new ADT OpenShape, which is defined as a base class hierarchy to represent open geometries whose boundaries are not closed. We focus on the data types that are needed to model open direction region. The function interfaces for OpenShape related to the processing of direction relation queries can be defined in C++ notation. Fig.3. shows some examples of open shapes that are useful for the processing of direction relation queries. Therefore, new data types for open boxes can be defined as derived classes of OpenShape, and the attributes can be described using open rectangles, directions and points. We only show the attributes and constructors of each open shape class, as shown in Table 1:

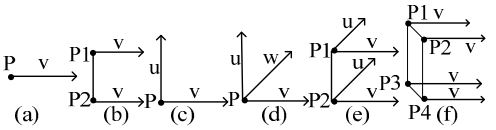

Fig.3. Examples of open shapes

Table 1 Abstract Data Types for Open Shapes

ADT	attributes	constructors
OpenLine1	startPoint: Point; dir: Direction	OpenLine1(Point, Direction);
OpenRect1	Vertex1:Point;vertex2:Point; dir: Direction	OpenRect1(Point, Point, Direction)
OpenRect2	startPoint: Point; dir1:Direction;dir2:Direction	OpenRect2(Point, Direction, Direction)
OpenBox1	openrect:OpenRect2;dir: Direction	OpenBox1(Direction ,OpenRect2(Point, Direction, Direction))
OpenBox2	openrect1:OpenRect1;dir: Direction	OpenBox2(Direction,OpenRect1(Point, Point, Direction))
OpenBox3	openrect1:OpenRect1; openrect2:OpenRect1	OpenBox3(OpenRect1(Point,Point, Direction), OpenRect1(Point, Point, Direction))

4.2. Topological Operations on Open Shapes

In this section, we will focus on defining a topological operation, namely, *interiorIntersects*, since this operation is needed to solve direction predicates. The *interiorIntersects* (open shape O, closed shape C) operation is defined, where O could be an OpenBox1, OpenBox2 or OpenBox3 and closed shape C could be a box. The implementation of *interiorIntersects* uses the two topological operations of *contains* and *crosses*. For any pair of open shape O and closed shape C, the *interiorIntersects* relation between them is defined as:

$$O.interiorIntersects(C)=TRUE \Leftrightarrow interior(C) \cap interior(O) \neq \phi$$

In other words, O.*interiorIntersects*(C) is true if C is contained in O or C *overlaps* O. Since a closed object C cannot contain an open object O due to infinite extent, we do not have to check for C contains O. We will discuss the implementation of the OpenBox1.*interiorIntersects*(Box), OpenBox2. *interiorIntersects*(Box) and OpenBox3.*interiorIntersects*(Box) respectively.

4.2.1The interiorIntersects operation for OpenBox1

An OpenBox1 *interiorIntersects* a rectangle if there exists a set of points which belong to both the interior of the OpenBox1 and the interior of the box. Fig.4 shows several situations that an OpenBox1 *interiorIntersects* a box. The boolean value of *interiorIntersects* operation can be determined by enumerating all possibilities of the interiorIntersecting relationship between an OpenBox1 and a box. O.*interiorIntersects*(C) is TRUE if and only if at least one of the following three conditions is true:

(1)At least one of the endpoints of C is contained in O (e.g. Fig.4 (a));

(2)All endpoints of C are inside O;(e.g. Fig. 4(b));

(3)At least one endpoint of C is outside O and at least one OpenLine1 of the three OpenLine1s boundary of O *crosses* C (e.g. Fig. 4(c));

The first case can be tested by using OpenBox1. *contains*(Point) operation, the second case is checked by OpenBox1.*contains*(box) operation, and the third case is checked by using OpenLine1.*crosses*(box). If any of the endpoints of O is within C, the *interiorIntersects* operation can be determined to be true without further tests as shown in Fig. 4(d). The pseudo-code for the OpenBox1.*interiorIntersects*(Box) operation is described as Algorithm 1.

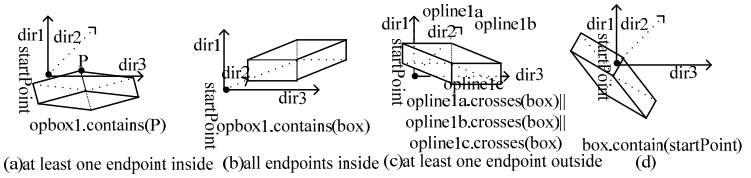

Fig.4. An OpenBox1 interiorIntersects a Box

Algorithm 1:OpenBox1 *interiorIntersects* a Box

```
Input: The box needs to be checked;
       The current OpenBox1 object, which has attributes (dir1, OpenRect2 (startPoint, dir2,dir3));
OpenBox1::interiorIntersects(Box, box){
 /*case1, Fig.4(a)*/
  for each ep ∈ endpoints of box
    if(contains(ep))
      return TRUE;
 /*case2,Fig.4(b)*/
  if(contains(box))
    return TRUE;
```

```
    /*case3,Fig.4(c)*/
     opline1a=Openline1(startPoint, dir1);
     opline1b=Openline1(startPoint, dir2);
     opline1c=Openline1(startPoint, dir3);
           if(opline1a.crosses(box)||
              opline1b.crosses(box)||
              opline1c.crosses(box))
            return TURE;
          return FALSE;
}
```

4.2.2 The interiorIntersects operation for OpenBox2

The situation of OpenBox2.*interiorIntersects*(Box) is similar with Open Box1.*interior- Intersects*(Box) as shown in Fig.5. The pseudo-code for the OpenBox2.*interiorIntersects*(Box) operation is described as Algorithm 2.

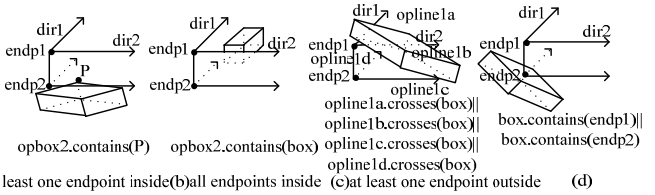

Fig.5. An OpenBox2 interiorIntersects a Box

Algorithm 2.:OpenBox2 *interiorIntersects* a Box

```
  Input: The box needs to be checked;
      The current OpenBox2 object, which has attributes(dir1,OpenRect1(endp1,endp2,dir2));
  OpenBox2::interiorIntersects(Box, box){
    /*case1, Fig5(a)*/
     for each ep ∈ endpoints of box
      if(contains(ep))
        return TRUE;
    /*case2,Fig.5(b)*/
     if(contains(box))
       return TRUE;
   /*case3,Fig.5(c)*/
    opline1a=Openline1(endp1, dir1);
    opline1b=Openline1(endp1, dir2);
    opline1c=Openline1(endp2, dir2);
        opline1d=Openline1(endp2, dir1);
       if(opline1a.crosses(box)||opline1b.crosses(box)||opline1c.crosses(box)|| opline1d.crosses(box))
        return TURE;
     return FALSE;
       }
```

Lemma 2: Algorithm 1and Algorithm 2 are complete and correct.

Proof: Algorithm 1 tests three cases and it returns TRUE if and only if one of these three conditions is true. This guarantees the correctness of the algorithm, i.e., the algorithm returns TRUE only if the OpenBox1 *interiorIntersects* the box.

Let's consider a pair of disjoint OpenBox1 and box. Move the box horizontally and then the box will

touch the OpenBox1. When continuously moving the box horizontally, OpenBox1 *interiorIntersects* the box, and the relationship between OpenBox1 and box falls into the three cases as described in algorithm 1.This guarantees the completeness of the algorithm, i.e., the algorithm returns FALSE only when there is no intersection between the interior of OpenBox1 and the interior of the box. The arguments of completeness and correctness for the algorithm 2 are similar.

5. Conclusions

In this paper, the description model of object-orientation-based direction relation in three-dimensional is put forward based on that in two dimensional space. Then the base class of OpenShape in two-dimensional space is extended, and an object-orientation-based open shape model based on the description model for direction relations in three-dimensional space is proposed. The basic idea is that the actual direction region is modeled as an open shape by constructing a new ADT OpenShape, the computation related to the world boundary of spatial direction region is eliminated, and the processing of the direction predicates is converted into the processing of topological operations between open shapes and closed geometry objects. The I/O and CPU cost of the processing of direction relation queries based on the new open shape model in three-dimensional space, which will be presented in the next paper, is reduced by improving the filtering effectiveness. In future work, we will continue to extend the base class of OpenShape, and study the processing of direction relation queries involving viewer-based direction predicates by deriving more class and operations for ADT OpenShape. The processing of object-orientation-based direction relation queries in a mobile environment is also a direction of future work.

References

[1] Shashi Shekhar,Xuan Liu,Sanjay Chawla. An Object Model of Direction and Its Implications[J] GeoInformatica, 2009,13,(4):357~379

[2] X.Liu,S.Shekhar,S.Chawla. Object-Based Directional Query Processing in Spatial Databases[J]. IEEE TKDE, 2008,20(2):295-304

[3] Y.Theodoridis, D.Papadias. Range Queries Involving Spatial Relations: A performance Analysis [C]. Proceedings of the 2nd Conference on Spatial Information Theory (COSIT), 2010:176-180

[4] Yi Zhenshan.Simplex-based 3D-GIS Data Model and Its Preliminary Design.[J].Bulletin of Surveying and Mapping,2009,12(11):10-13

[5] Qing Zhu et al. An efficient 3D R-tree spatial index method for virtual geographic environments[J]ISPRS Journal of Photogrammetry and Remote Sensing,2007,62(3):217-224