# ONE RASTERIZATION APPROACH TO SPATIAL PREDICATE

*Hui Dong, Zhenlin Cheng, Yanwei Zhao, Jinyun Fang*

Institute of Computing Technology, Chinese Academy of Sciences
Graduate University of Chinese Academy of Sciences

## 1. INTRODUCTION

The description and representation of spatial relation/spatial predicate, which claims the boundary between GIS and Management Information System (MIS) and Computer Aided Design (CAD), is the kernel of GIS[1]. Spatial predicate is the integrated component of spatial filters specified in Web Feature Service(WFS) protocol.

Computational geometry approach to spatial predicate implementation has the advantage of high precision with the cost of memory usage and CPU time consumption. Slow response time is not desirable. In fact, there are many scenarios and applications where a fast answer with desired accuracy is more preferable than slow exact answer because getting useful data for decision making in a timely manner is becoming crucial for increasing in business competitiveness[2]. Furthermore, Internet map service is proliferating and becoming prominent in spatial decision support applications. But, the time and space complexity of computational geometry approach to spatial predicate is not affordable or scalable for massive concurrent sessions. In fact, in modern internet map server, the image tiles is pre-rendered or dynamically rendered on initial client request. In such scenarios, the cached image tiles are not only the visualization of vector data but also the rasterization approximation the exact geometry. Such images can provide more precise approximation than minimum bounding rectangle (MBR) and can be utilized as the basis of spatial predicate implementation.

Reference [3] presents one kind of filter: the raster approximation of polygons, named four color raster signature (4CRS). This approximation can be used to test if two polygons overlap. The 4CRS is used for representing polygons, and it is a small bit-map using four colors. Each color represents an intersection type between the object and the cell: empty(The cell is not intersected by the polygon)、weak(the cell contains an intersection of 50% or less with the polygon)、strong(the cell contains an intersection of more than 50% and less than 100% with the polygon) and full(the cell contains an intersection of 100% with the polygon). Only strong×strong is determined situation(overlap) and weak ×weak、strong×weak and weak×strong are uncertain situations needing further calculation. In the calculation of the approximate area and confidence interval, it uses mathematical expectation and probability formula to estimate which may not be suitable for the real data.

In this paper, we propose on improved method, combined with the rendering engine, which can enhance the filter accuracy and avoid the above drawbacks of 4CRS. This proposal can record coverage area of border grid cells accurately based on subpixel accuracy, so it can determine whether two polygons overlap through judging coverage area of the corresponding grid cells. For example, given that coverage area of the first layer is 49% and that of the second layer is 52% in the same grid position, it can determine that two layers overlap in this cell because the coverage area sum is more than 100%. But it is uncertain situation if using 4CRS because it can not handle weak×strong condition (not to mention weak×weak situation). At side effects, the rendering engine can preserve feature attribute information in the cell structure which offers more useful hints for map overlay (polygon IDs etc.).

## 2. RASTER SIGNATURE GENERATION

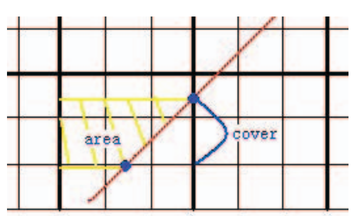
There are two key steps: (1) raster signature generation (2) to implement spatial predicate utilizing raster signature. The core idea for raster signature generation is using the subpixel accuracy of Bresenham algorithm[4],which is based on the error discriminate to generate line. The difference with traditional Bresenham is subpixel accuracy, which divides a pixel into N * N subpixels. In this paper, N is 256 which can meet the most requirements of real world vector data visualization.

First, convert vector points through coordinate conversion pipeline. Vector point coordinates are multiplied by 256 (this operation is equivalent to shift the coordinates of binary representation left eight bits), which has the advantages of considering the influence of fraction part to the pixel weights(cover). It can computed easily with subpixel accuracy[5].

Outline scanning and rendering for filling polygons is the core of algorithm. When scanning, saves the grid cell weights(cover) and area of border grid cells. Fy--one value between 0 and 255, is the fraction part of the origin vector coordinates. The cover and area of grid cell influenced by the line can be calculated easily and the formula is described as following:

$$\mathrm{cov}\,er = fy2 - fy1 \tag{1}$$

$$area = (fx2 + fx1) \times \mathrm{cov}\,er \tag{2}$$




When scanning outline of polygon is completed, it will render filled polygons. Firstly, sort all grid cells of border; Secondly, scan every line from staring position to end position. At the same time, filling the rendering buffer according to the coverage area of grid cell. Each cell span has initial X, length, and an array of bytes that determine the alpha-values for each pixel.

It is easy to figure out that the area which is obtained in the process of outline scanning is not the final area of cell because we don't know whether it is the interior or exterior of polygon when scanning the outline, so the actual area can only be obtained when rendering. Computing the actual area of each grid cell is: accumulating cover and area in the same grid cell and, recording the sum as alpha. Below is the pseudo code:

```
calculate_alpha
    {
        poly_base_shift = 8;            aa_shift = 8;
        aa_num   = 1 << aa_shift;       aa_mask  = aa_num - 1;
        aa_2num  = aa_num * 2;          aa_2mask = aa_2num - 1;
        int alpha = ((cover << (poly_base_shift + 1)) - area)>> (poly_base_shift*2 + 1 - aa_shift);
        if(alpha < 0)  alpha = - alpha;
        alpha &= aa_2mask;
        if(alpha > aa_num)  alpha = aa_2num - alpha;
        if(alpha > aa_mask)  alpha = aa_mask;
      return alpha;
    }
```

## 3. IMPLEMENTING SPATIAL PREDICATE USING RASTER SIGNATURES

After the above steps the raster signature is saved in the form of matrix. Each grid cell structure includes polygon ID this cell belongs to and coverage area. Location is inherent in the storage structure, namely, implied by row and column number of the grid cell rather than through the use of explicit spatial coordinates. When implementing spatial predicate, we only need to find the specified grid cell and examine the polygon ID this cell belongs to and coverage area to get judgment and return qualified polygon IDs. We will illustrate this method by taking the predicate of overlap and contain for example.

### 3.1. overlap

Given that there are two input feature sets-A and B, judge whether the two layers overlap or not. If it is true(regarding the overlap possibility as filtering condition, overlap possibility is more than 60% is considered overlapping), return the qualified polygon IDs. The pseudo code is listed as followings..

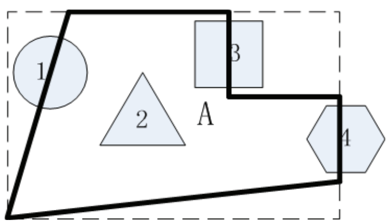
```
for(loop y-scanline in the envelope){
        for(loop x-scanline in the envelope){
                if(the (x,y) grid is not empty in two layers){
                        cellarea1、id1 are coverage area and polygon ID of layer A respectively;
                        cellarea2、id2 are coverage area and polygon ID of layer B respectively;
                        result set maintains the overlap pairs of polygon IDs（id1 and id2）and overlap possibility
（pvalue=undetermined value/ undetermined cellCount）
                        if(overlap pairs are in the result set && it is not sure whether overlap or not){
                                if(cellarea1+cellarea2 >= 100%)
                                        it shows two polygons overlap in the current cell，update result set
                                else{
                                        value += cellarea1*cellarea2;   count++;
                                }
                        }else{   // overlap pairs are not in the result set
                                if(cellarea1+cellarea2 >= 100%)
```

```
                                        it shows two polygons overlap in the current cell，update result set
                                  else{
                                      value = cellarea1*cellarea2;  count = 1;   insert to result set（id1 , id2 and pvalue）
                                  }
                        }
              }
      }
}
```

### 3.2. contain

Given that there are two input feature sets-A and B, judge whether any polygon in A(named pa) contains any polygon in B(named pb) or not. If it is true, return polygon ID pair(pa, pb). As illustrated in the figure below, layer A is in bold lines and layer B consists of four polygons.

There are three steps: (1) obtain the candidate set result1. Every polygon in this set belongs to B and the coverage area of each grid cell is less or equal to the counterpart of A. In this example, polygon 2、3 and 4 in layer B is qualified while polygon 1 in B is excluded because coverage area. (2)compare the envelope of polygons of result1 to that of layer A, remove the polygons whose envelope is bigger than that of layer A and insert the remain to candidate set result2(polygon 2 and 3 in this example). (3) compare the envelope of polygons of result2 to that of polygon(named pa) in A, remove the polygons whose envelope is bigger than pa and insert the remain to result set result3(polygon 2 in this example).Result3 is the final result set and that is layer A contain polygon 2 of B.




## 4. REFERENCES

[1] Longley P, Goodchild M F, Maguire D, et al. Geographical Information Systems: Principles, Techniques, Management, and Applications. 2nd ed. New York: Wiley. 2005. 97-99.
[2] Leonardo Guerreiro Azevedo, Geraldo Zimbrão, and Jano Moreira de Souza, “Approximate Query Processing in Spatial Databases Using Raster Signatures,” VIII Brazilian Symposium on GeoInformatics, Campos do Jordão, Brazil, November 19-22, 2006, INPE, p. 3-17.
[3] T. Brinkhoff, H. P. Kriegel, R. Schneider, and B. Seeger, “Multi-step Processing of Spatial Joins,” In Proceedings of the 1994 ACMSIGMOD Conference, Minneapolis, USA, May 1994.
[4] Bresenham J E .Algorithms for computer control of a digital plotter[J]. IBM Systems Journal,1965,4(1):25—30.
[5] http://www.antigrain.com/doc/index.html