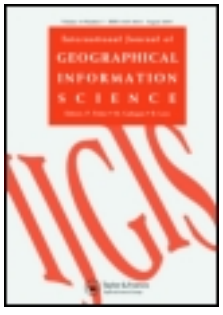

International Journal of Geographical Information Science

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tgis20

A characteristic bitmap coding method for vector elements based on self-adaptive gridding

Yongning Wen [a], Min Chen [b], Guonian Lu [a], Hui Lin [b] & Songshan Yue [a]

[a] Key Laboratory of Virtual Geographic Environment (Ministry of Chinese Education), Nanjing Normal University, Nanjing, Jiangsu, China

[b] Institute of Space and Earth Information Science, The Chinese University of Hong Kong, Hong Kong, China

Version of record first published: 02 Apr 2013.

To cite this article: Yongning Wen , Min Chen , Guonian Lu , Hui Lin & Songshan Yue (2013): A characteristic bitmap coding method for vector elements based on self-adaptive gridding, International Journal of Geographical Information Science, DOI:10.1080/13658816.2013.774006

To link to this article: http://dx.doi.org/10.1080/13658816.2013.774006

International Journal of Geographical Information Science, 2013
http://dx.doi.org/10.1080/13658816.2013.774006

Taylor & Francis
Taylor & Francis Group

A characteristic bitmap coding method for vector elements based on self-adaptive gridding

Yongning Wen[a], Min Chen[b]*, Guonian Lu[a], Hui Lin[b] and Songshan Yue[a]

[a]*Key Laboratory of Virtual Geographic Environment (Ministry of Chinese Education), Nanjing Normal University, Nanjing, Jiangsu, China;* [b]*Institute of Space and Earth Information Science, The Chinese University of Hong Kong, Hong Kong, China*

(*Received 23 September 2012; final version received 4 February 2013*)

Spatial index is a key component of Geographic Information Systems (GISystems). To date, an increasing number of spatial indexes have been developed to enhance the efficiency of spatial analysis and spatial query. Approximate expressions are adopted in the foundation of spatial index construction, to assist data organisation, e.g., bounding box of vector elements can be employed to build R-tree index. However, R-tree index using such bounding box expresses elements approximately, which usually results in redundancy and excessiveness due to inherent roughness of this method. This study proposes a characteristic bitmap coding method, termed the QCODE method, which generates approximate expressions of vector elements based on self-adaptive gridding. Based on the sizes of vector elements, this method selects the grid at an appropriate level in a self-adaptive manner, discretises the vector elements into grids through a rasterisation operation, as well as compresses and encodes this information as the code of a characteristic bitmap, i.e., the QCODE. The 'bit' data type is used in the design of QCODE to restrict the approximate expression of vector elements into finite bytes, providing more precise filtering as compared to the case in which only bounding box is used. With its distinct characteristics, the QCODE is introduced for the improvement of R-tree index. The results of experiments show that, combined with R-tree index, this method can reduce the filtering amount of vector elements as required for spatial analysis, and accelerate the execution efficiency of the entire process of GIS spatial analysis.

Keywords: characteristic bitmap; QCODE method; approximate expression; adaptive gridding; R-tree

1. Introduction

The construction of spatial index is a key issue in the development of GISystems, especially geographic databases (Goodchild *et al.* 1992, Guting 1994, Laurini 1998, Rigaux *et al.* 2002). By support of approximate expression methods, spatial index is used to group the physical items of data, thus non-essential data can be filtered out in a calculation so that spatial analysis can be applied more efficiently (Ooi *et al.* 1993, Colak 2002, Shekhar and Chawla 2003, Banerjee *et al*. 2008).

*Corresponding author. Email: tianxiaxue99@163.com
Yongning Wen is also affiliated with the Postdoctoral Workstation, Suzhou Industrial Park Survey CO.LTD, China.

There are two major approximation methods in the construction of spatial index: one is to produce a relatively simple approximate geometric shape (e.g., a bounding sphere or bounding box) based on the outline of the vector elements (Mamoulis 2011, Zhao *et al.* 2012); the other is to discretise the vector elements into grids and use the approximate shape within the grid cells, instead of the vector elements, to perform the preliminary calculations (Poiani *et al.* 1996, Pilevar and Sukumar 2005, Zheng *et al*. 2006). The bounding box in the first class of methods is one of the most common approximation shapes (Baum and Hanebneck 2010, Brisaboa *et al*. 2010, Chaudhuri *et al*. 2012), and it is often used to construct the R-tree Family index (e.g., R-tree, R^*-tree and R^+-tree) of the vector elements (Guttman 1984, Sellis *et al.* 1987, Beckmann *et al*. 1990, Kuan and Lewis 1997). The bounding box, which is capable of supporting an efficient judgment algorithm, is characterised by its simple structure and low-storage requirements. However, the filtering results are usually redundant and/or excessive due to the inherent roughness of the method (Beckmann *et al*. 2009, Al-Badarneh *et al.* 2010, Siqueira *et al*. 2012). The second class of methods uses divided grids to approximate the vector elements. In fact, rasterisation techniques have been widely studied in the field of computer graphics to approximately express objects in 2D or 3D (e.g., Gibson and Lucas 1982, Lindholm *et al*. 2008, Novak and Dachsbacher 2012), and divided grids are increasingly used to express the geographic elements (e.g., Wang *et al*. 2003, Harley and Gellerstedt 2005). However, the classical grid method requires a large storage space to record the locations of the grid cells. Additionally, the intersection operations of this method require looping through each row (or column), a process that can become computationally intensive (Maheshwari *et al.* 2002, Bivand *et al.* 2008, Zhang *et al*. 2011). Therefore, it is vital to find an effective approximation technique for vector elements to increase the efficiency of GIS spatial index (Pramanik and Li 1999, Berman and Raskhodnikova 2010).

This article presents a characteristic bitmap coding method for approximate expression of vector elements based on self-adaptive gridding, named the Quad-Code (QCODE) method, because it is designed on the basis of multilevel, 8×8, quad-tree grids. It is worth mentioning that the proposed method can be used in combination with a spatial index (in this article, R-tree is employed) to reduce the filtering as required for spatial analysis. Although the querying efficiency will be lower, when the combination method is used, as compared with the case when only R-tree is used, fewer vector elements will be selected for the next step of spatial computation (e.g., intersection), and the overall efficiency of the process will be improved. Based on the sizes of the vector elements, this method first prescribes a self-adaptive method of grid selection. The method then discretises the vector elements into the grid through a rasterisation operation. Finally, the discretised information is compressed and encoded to compose the codes of the characteristic bitmap-QCODE utilising the 'bit' data type of the computer. In the QCODE method, the encoding information for a certain data volume can cover the geometric features of the vector data more accurately than the bounding box method. Moreover, it can filter the vector datasets more precisely and efficiently using bit operations, and can reduce the time consumption in the entire process of spatial analysis.

This article consists of five sections. Section 2 introduces the design strategy for the QCODE method. Section 3 presents the query optimisation algorithm based on the QCODE method, and outlines the resolution requirements for the rapid conversion of QCODE among vector elements at different levels and the rapid intersection operations of the vector elements. Section 4 incorporates the QCODE method into the R-tree index to improve the R-tree filtering results. The results demonstrate that the QCODE method can

improve accuracy in the R-tree index filtering. The final section presents the conclusions of this work and discusses future improvements.

2. Design of the QCODE method

2.1. Multilevel grids in the QCODE method

The QCODE method is based on multilevel grids that are used to determine an approximate expression for the vector elements. The gridding process used in this method is similar to that of the classical gridding index, except for the three rules described below.

(1) For gridding, it is necessary to determine the constitution of a bounding box $(xmin, ymin, xmax, ymax)$, with a side length of $D = xmax - xmin = ymax - ymin$, for the original vector element sets. The width of this bounding box equals the height, and the box covers all elements in the dataset. If the width and height of the bounding box are not equal, then the longer side is taken as the side length of the quadrate bounding box.
(2) The grid used in the QCODE method is made up of a series of multilevel grids. The initial level is called $Level_0$, and has eight rows and eight columns. Thus, the size of the grid cell at $Level_0$ is $d_0 = D/8$.
(3) Among the multilevel grids, the size of a grid cell at a given level is half the size of a grid cell at the next lower level. Therefore, the size of a grid cell at $Level_1$ is $d_1 = d_0/2 = D/16$, and the size of a grid cell at $Level_2$ is $d_2 = d_1/2 = D/32$. The remaining sizes can be calculated in a similar way.

Figure 1 demonstrates the grids within an array ranging from $Level_0$ to $Level_2$ that are used in the QCODE method. A grid cell at one level splits into four cells at a higher level. Consequently, the grid adopted in the QCODE method is a quad-tree grid with initial cells of dimensions 8×8. Thus, the encoding method developed in this article is called the QCODE method.

2.2. Characteristic bitmap of the QCODE method based on rasterisation

The characteristic bitmap of the QCODE method is produced based on the rasterisation of the vector element's geometric shape. Based on the multilevel grids introduced in

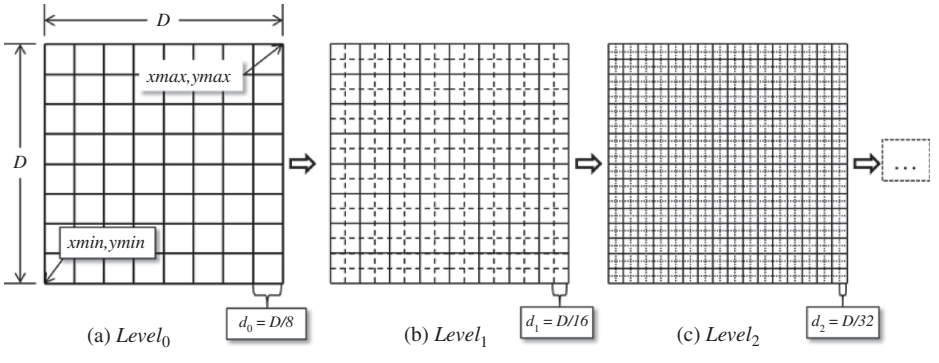

Figure 1. Multilevel grids ($Level_0$ to $Level_2$) used in the QCODE method.

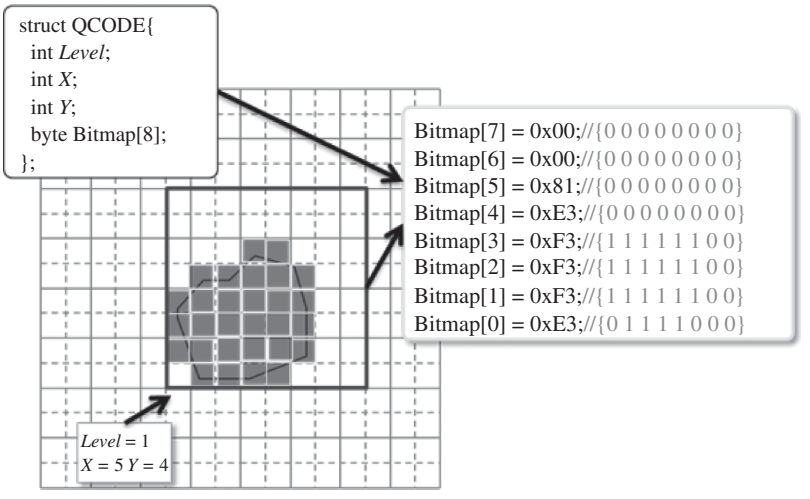

Figure 2. Constitution of the characteristic bitmap.

Section 2.1, the characteristic bitmap of the QCODE method is a binary image with the same pixel size as that of the grid cell of a specific level. The number of grid cells in the image is 8 × 8 (starting from the bottom-left). Figure 2 illustrates basic information about the characteristic bitmap, including the information described below:

(1) The level of the multilevel grid that contains the characteristic bitmap. For example, the grid of the vector element in Figure 2 is at $Level_1$.
(2) The initial position (lower-left coordinates) of the characteristic bitmap within the grid. For example, the initial position of the bitmap in Figure 2 is (5, 4).
(3) Pixel information about the characteristic bitmap. Pixel information about the bitmap is crucial for the QCODE method. After rasterising the vector elements, the grid cells in the quadrate grids covered by the bitmap are either unfilled and marked as 0 or filled and marked as 1. The information on the bitmap pixel can thus be expressed in 8 bytes. Each byte indicates a scanning line of the bitmap, and the position indicates whether the grid cell is filled.

In summary, the grid level (*Level*), initial position (*X, Y*) and pixel information can be expressed by data organisation, as demonstrated by the QCODE structure in the bottom right corner of Figure 2.

2.3. Strategy for selecting self-adaptive grid level

The strategy used to select self-adaptive grid levels for these vector elements requires selecting an appropriate level from the constructed multilevel grids in accordance with the minimum bounding box overlaid with the most precise 8 × 8 grid cells. Based on the quad-tree segmentation of the multilevel grids, the *Level* of a vector element can be determined using the following formula:

$$Level = floor(\log_2 D / \max(w, h))$$

In this formula, D refers to the side length of the bounding box of the vector dataset (Section 2.1); w refers to the width of the minimum bounding box of this vector element, which also fixes the grid level; h is the height; *max* is the function used to calculate the maximum value; and *floor* is the function that takes on the integer value of a floating number.

Based on this, the side length of the characteristic bitmap that corresponds to this vector element at the calculated *Level* is

$$Size_{level} = D/2^{Level}.$$

The size of the pixel of the characteristic bitmap that corresponds to this vector element is

$$d_{Level} = Size_{level}/8$$

After the *Level* is determined, the initial position of the characteristic bitmap that corresponds to this vector element can be calculated as follows:

$$\begin{cases} X = floor(x_0/d_{Level}) \\ Y = floor(y_0/d_{Level}) \end{cases}$$

In this equation, x_0 and y_0 are the minimum numbers of rows and columns of the grid cells.

As shown in Figure 3, the grid level of a vector element is determined by its minimum bounding box. The vector elements in a grid at this level can be covered by an 8×8 characteristic bitmap. As the size of the bounding box decreases, the level of its grid becomes greater and the size of a grid cell decreases.

2.4. The QCODE structure of vector elements and its optimisation

The aforementioned strategy assigns the given vector element to a grid level (*Level*) and an initial position (X, Y), and provides the information about the characteristic bitmap. These pieces of information together constitute the QCODE of the vector elements.

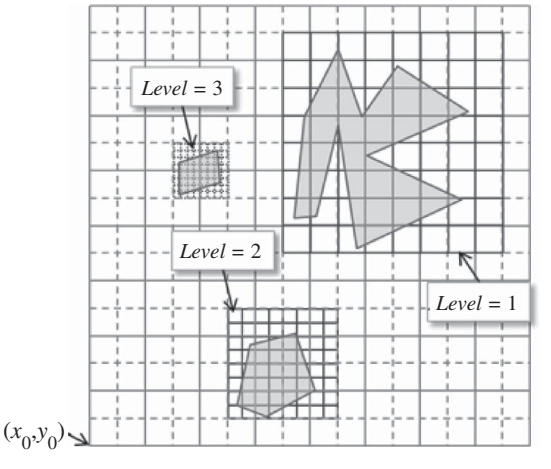

Figure 3. Self-adaptive grids of vector elements.

Using the structure designed in Section 2.2, if the values of *Level*, X and Y are all integers in a 32-bit computer (each integer requires 4 bytes), then the QCODE occupies $sizeof(Level) + sizeo(X) + sizeof(Y) + sizeof(Bitmap) = 20$ bytes in total, where sizeof is a function that returns the number of bytes used by the computer system to store the given argument. Notably, the *Level*, X and Y values are related to the grid level so that the length of a QCODE can be optimised based on the specific vector elements, which can be calculated by $Length = ceiling(((\log_2^{Level}) + 2 \times Level + 6)/8) + 8$. For example, if the highest vector element level in a vector element set is *Level* 7, then *Level* ($2^3 - 1 = 7$) can be expressed in 3 bits, and the values of the maximum row and column of the grid can be expressed as $2^{3+7} - 1$. Therefore, X and Y require 10 bits each. If the highest vector element level is *Level* 8, then *Level* can be expressed in 4 bits, and X and Y would require 11 bits each. Consequently, the data structure of a QCODE can be further optimised. The highest grid level of the vector elements in the vector element set can be standardised to determine the values of *Level*, X and Y, and this unnecessary memory overhead can be reduced. Figure 4 demonstrates the model of a memory layout of a QCODE expressed in 12 bytes.

In conclusion, the approximation of the vector elements using the QCODE method has the following salient features: (1) it can be easily accomplished by the classical rasterisation of vector elements, (2) a QCODE includes information on sizes, locations and outlines of the vector elements, and can be treated as a hash code of the vector elements, and (3) vector elements can be filtered to a high approximation by the bit operation based on the characteristic bitmap as dictated by the QCODEs. Consequently, the efficiency of the spatial query and spatial analysis can be improved.

3. Calculating the intersection of the vector element based on the QCODE method

3.1. Principle for calculating the intersection of vector elements based on QCODE

Calculation of the spatial relationships of the vector elements in a dataset constitutes the foundation of GIS studies, including spatial analysis, spatial query and spatial statistics (Schneider and Eberly 2003, Molet *et al.* 2010). In large vector datasets for spatial processing, it is critical to have an efficient algorithm for calculating spatial relationships. However, an accurate calculation of the spatial relationships is often so complicated that it consumes a great deal of time. Therefore, establishing a spatial index of the vector elements and using their approximate expressions to filter non-essential vector elements for

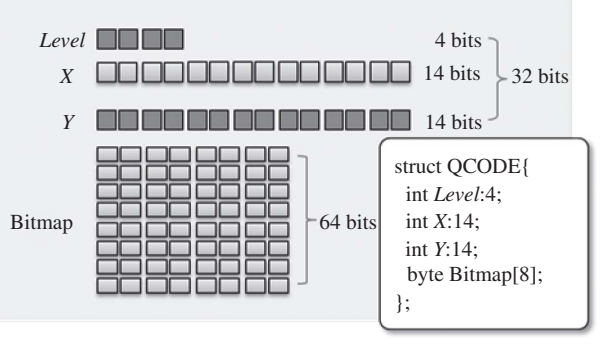

Figure 4. A 12-byte QCODE.

an accurate spatial calculation can be more time-efficient. Based on the QCODE method, the optimisation process can be carried out as follows:

(1) The QCODE generation process also designates the grid that completely contains the vector element. This relationship can be defined as:

$$G \in QCODE(G)$$

In the formula, G represents the vector element.

Accordingly, if the two $QCODE(G)$ sets of two different vector elements do not intersect, we can argue that the vector elements do not intersect each other, i.e.,

```
If QCODE(G1) ∩ QCODE(G2) = ∅

then

G1 ∩ G2 = ∅
```

Here, *G1* and *G2* refer to the two vector elements.

The evaluation of whether spatial elements intersect is one of the primary issues in determining spatial relationships and spatial queries (Gold 2009). Therefore, to calculate the precise spatial relationship between two or more vector elements, it is firstly necessary to determine whether the related $QCODE(G)$.formulations intersect. For non-intersecting cases, the following calculation steps can be omitted to increase efficiency.

(2) All data in the QCODE are integers. The characteristic bitmap is expressed using the 'bit' to indicate whether the grid cells are occupied or not. As the efficiency of an integer operation and bit arithmetic in a computer instruction is much higher than that of a floating-point operation, the evaluation time for a possible intersection can be reduced using the 'bitwise AND' operation.

3.2. Intersection judgment of QCODEs at the same level

The features of QCODEs are such that if QCODEs of two vector elements are marked with the same grid level and have the same initial positions (X, Y), the 'bitwise AND' operation can be directly engaged using the characteristic bitmap information, and the results can then be used to evaluate whether the two vector elements intersect. If their initial positions differ, however, then the alignment is carried out through the characteristic bitmaps of two QCODEs, and the 'bitwise AND' operation can be used to evaluate the nature of the intersection according to the following steps.

(1) First, align the 8×8 grid in the X direction. The information on each row in the characteristic bitmap is stored as 1 byte (8 bit). In the X direction, the rows can be offset, resulting in changes to the starting point of X in the QCODE. Because the coordinate direction in the bitmap is in the opposite direction of the bytes' ranking direction, the left offset is operated on in the order of the right displacement of the characteristic bitmap information in the QCODE. Figure 5 shows two QCODEs: Q_1 and Q_2 of the two vector elements (the vector element of Q_2 is not shown). Q_1 starts at (5, 4) and Q_2 at (8, 7). The intersection between the vector

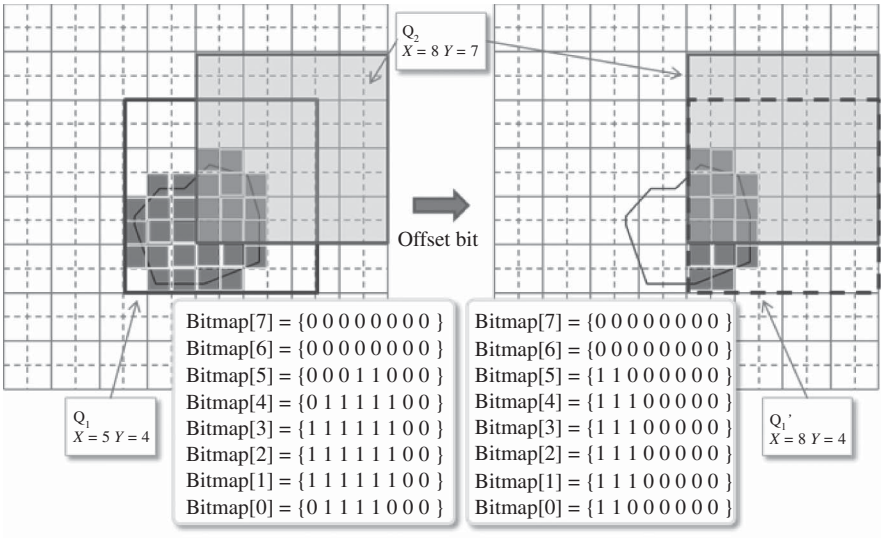

Figure 5. Alignment in the X direction of the characteristic bitmap of a QCODE.

elements expressed by Q_1 and Q_2 is performed in the following way. The characteristic bitmap of Q_1 is aligned with that of Q_2. All the rows of the characteristic bitmap of Q_1 are moved to the right by 3 cells, and the new position of Q_1 is given by the coordinates (8, 4). Thus, the information in each row of the characteristic bitmap should be offset by 3 bits to the left, and Q_1' is produced.

(2) Second, execute the intersection operation of the rows. After the alignment in the X direction, it is necessary to compare the information in the rows of Q_1' and Q_2. The comparison process requires iterative operations, leading to lower execution efficiency (Clifford 2011). To improve the efficiency, the following two methods are suggested.

The first method can be employed without the necessity of meeting additional conditions. As the number of rows in the characteristic bitmap is fixed, only eight possibilities exist for the intersection in the Y direction, and this can be encoded to avoid the loss in efficiency due to recurrent executions. Figure 6 shows the encoding condition used to calculate the intersection when the value at the Y coordinate differs by 0 or 1. In Figure 6, the symbol

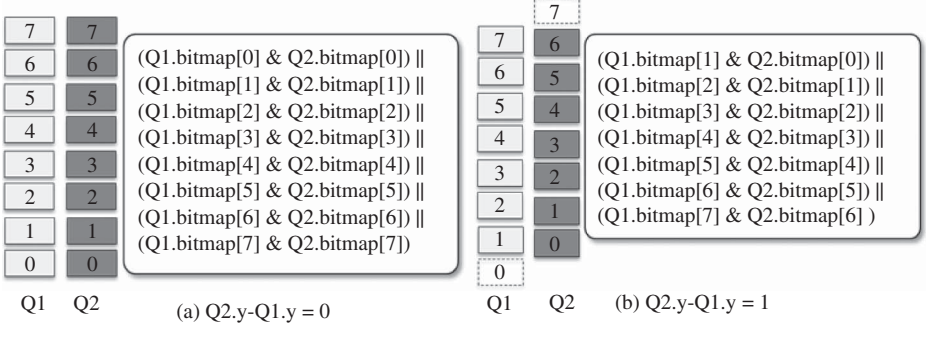

Figure 6. Intersection judgment of bitmap information.

'&' refers to the 'bitwise AND' operation, and the symbol '||' refers to the 'OR' operation. Similar rules can be applied when the value at the Y coordinate differs by 2, 3, 4, 5, 6 or 7.

In the second method, in cases in which this algorithm is implemented in the C/C++ language and the compiler is in support of 64-bit integers, the following data structure can be designed to record the information of the characteristic bitmap:

```
union BitmapUnion {
            unsigned char Bitmap[8];
            INT64 Bitmap64;
};
```

INT64 is the 64-bit integer type supported by the compiler, e.g., the __int64 of the MSVC compiler (Microsoft 2012). Bitmap [8] and Bitmap64 share the same memory address space. Similar to the X alignment operation, with Big Endian as the byte order, moving the Bitmap64 to the left by 8 bits is equivalent to a $Y + 1$ displacement in the QCODE, while a movement to the right by 8 bits is equivalent to a $Y - 1$ displacement. The displacements are executed in accordance with the differentials of Y between Q_1 and Q_2. Subsequently, the 'bitwise AND' operation is executed using the Bitmap64 value. The result can be used to validate the intersection of the two vector elements.

3.3. Intersection judgment of the QCODEs at different levels

If the QCODEs of two vector elements are marked with different grid levels, then the sizes of the grid cells of these two vector elements will differ. Therefore, the alignment and 'bit' operations cannot be executed directly to evaluate the intersection relationship of the vector elements. In this case, the characteristic bitmap information of the QCODE at one level should be transferred according to the other level. As the size of a characteristic bitmap at a lower level is larger than that at a higher level, a transfer from a lower level to a higher one would involve several characteristic bitmaps at the higher level. Therefore, it is more convenient to transfer from a higher level to a lower one to avoid repeated comparisons.

The strategy designed to transfer a characteristic bitmap from *Level* to *Level* –1 is as follows. Suppose that Q represents the characteristic bitmap of a vector element and the self-adaptive grid level is at *Level*. The initial position of the characteristic bitmap in the newly generated self-adaptive grid is *floor (Q.X/2)* and *floor (Q.Y/2)*. The grid cells in *Level* that belong to the same cell in *Level –1* are merged.

The initial positions of the new characteristic bitmap and the original bitmap may be misaligned, as shown in Figure 7, where the red one represents the high-level characteristic bitmap and the grey one represents the low-level bitmap. Figure 7a demonstrates an overlap of the initial positions of two characteristic bitmaps, and Figure 7b shows the misalignment situation. The value of *Q.X MOD 2* can be taken as the alignment offset, and *MOD* is the mode operation.

First, the merging situation in the X direction is considered. When the aligning offset is 0, as shown in Figure 7a, the two grid cells (No. 0 and No. 1) in the same row will merge into a single grid cell (No. 0) at the next lower level. The other transformations can be calculated in a similar manner. When the alignment offset is 1, as shown in Figure 7b, the original grid cell (No. 0) corresponds to the grid cell No. 0 at the next lower level. The grid

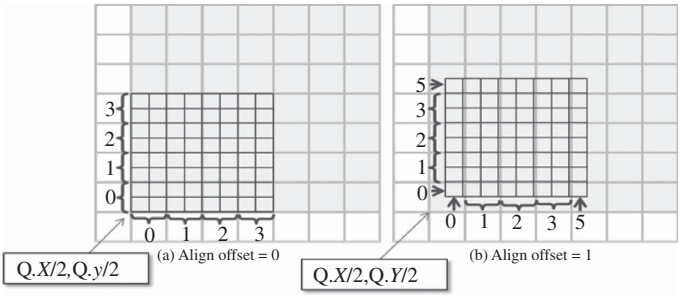

Figure 7. Alignment strategies for transferring the characteristic bitmaps at different levels.

cells (No. 1 and No. 2) merge into the grid cell No. 1, and the grid cell (No. 7) corresponds to No. 5 at the next lower level.

Based on the above method, the row information can be transferred from a high-level characteristic bitmap to a lower-level one. In the current research, a mapping table has been designed for the transferal process. Information for each row of the bitmap in the QCODE is expressed using 8 bits (1 Byte), which can represent any integer within the set [0, 255], corresponding to the different filling modes of the grid cells. Thus, it is appropriate to establish a mapping table to record each value using the row information at the high level and that at the corresponding lower level. Every row of bitmap information of the QCODE can be transferred from the high level to the corresponding lower level, with the mapping table being queried. Figure 8 shows the partial mapping relationships when the corresponding offset is 0 or 1.

The same procedure is repeated for the Y direction. The alignment displacement must be considered in this procedure. The merging process can be realised using the 'OR' operation.

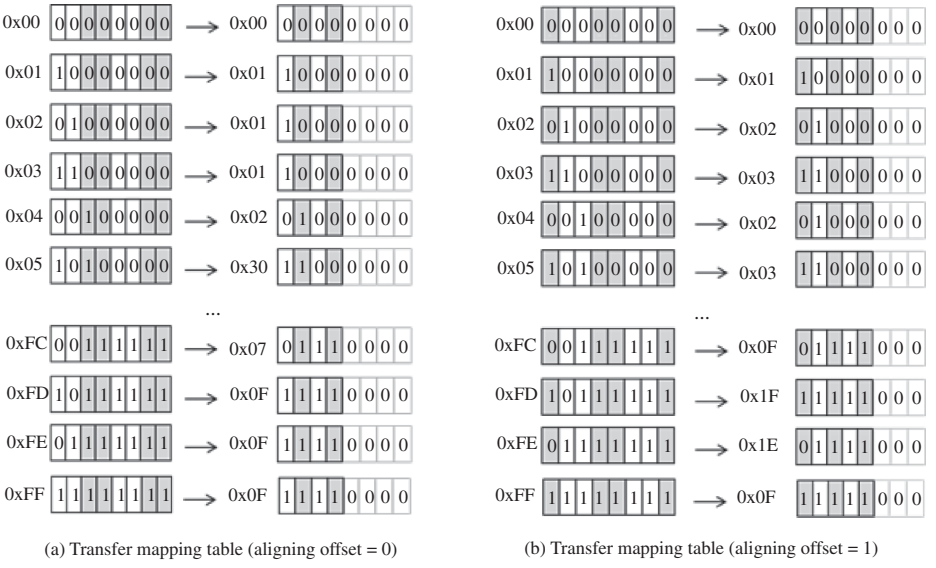

Figure 8. Mapping table of row information for transferring the characteristic bitmap to different levels.

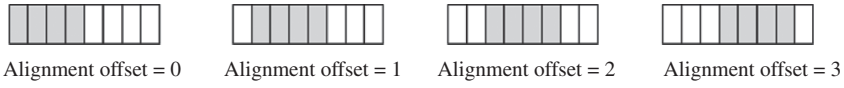

Figure 9. Alignment situations where the self-adaptive grid differs by two levels.

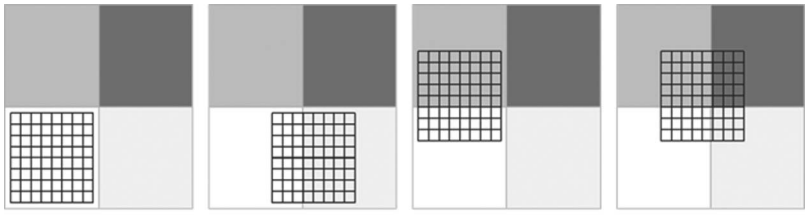

Figure 10. Relationships where the differential of self-adaptive grid levels is greater than 2.

Similarly, a transfer mapping table can be established, in which the self-adaptive grid differs by two levels. In this situation, four offsets, i.e., 0, 1, 2 and 3, were considered during the design of the mapping table. Figure 9 illustrates these four conditions.

When the differential of the self-adaptive grid levels is greater than 2, the grid cell at a lower level can cover the length of a minimum of eight high-level grid cells. Therefore, the bitmap information of the original QCODE may have four relations and eight offset conditions with the low-level grid, as shown in Figure 10. The corresponding transfer mapping tables can be designed accordingly.

In conclusion, the situations, where the differentials of the self-adaptive grid levels are 1, 2 and greater than 2, correspond to 2, 4 and 8 mapping tables, respectively. Based on 14 sets of mapping tables, a complete transfer algorithm can be designed, and a rapid transfer of the QCODE at different levels can therefore be accomplished.

4. Experiments—Optimisation of the R-tree index based on QCODE

An experimental system has been developed to validate the QCODE method. The system was realised using C++ language with Microsoft ACCESS as the database. To generate QCODEs, the scanline algorithm (Lane and Carpenter 1979) was employed. An open source C++ graphics library titled Anti-Grain Geometry (AGG) was used to produce pixel images in the memory from vector data. The 8×8 bitmap with white colour was generated first, and then if the grid cell was occupied, it was filled with black. Using the bitmap, related information was extracted and QCODE was recorded. The QCODE is stored based on a BINARY type, and the geometric field is stored using the Binary Large OBject (BLOB) type with a Well Known Binary (WKB) format (see Figure 11).

4.1. R-tree improvement based on QCODE

R-tree is a method of spatial data indexing proposed by Guttman (1984). A series of variants, e.g., R^+-tree and R^*-tree, have been created since the emergence of R-tree to improve the search efficiency and precision (Guttman 1984, Sellis *et al.* 1987, Beckmann *et al*. 1990, Zhu *et al*. 2003, Arge *et al.* 2005, Zhu *et al.* 2007, Arge *et al*. 2008, Arge *et al.* 2009). R-tree is usually developed based on the bounding box of vector elements, usually

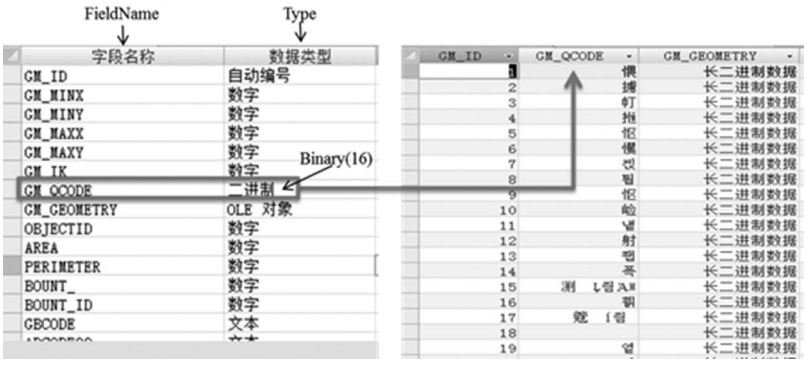

Figure 11. ACCESS Database for the storage of QCODEs.

taking the IDs of the vector elements and the bounding box as index entries. When a spatial query is executed, the bounding boxes are considered as the search conditions for data retrieval.

The optimisation of an R-tree index based on QCODE involves two steps:

(1) Put the QCODE of the vector elements into the index entries of R-tree. Except for the ID and bounding box, the improved index entries also record the QCODE information of the vector elements.
(2) The bounding box was used as the preferred search criterion with the QCODE acting as the auxiliary criterion. In this way, more accurate filtered results can be acquired during the retrieval process.

The whole process of generating and using QCODE can be illustrated in Figure 12.

4.2. Empirical evaluations

The validation data used for the experiment is the Chinese 1:1 Million Topographic Map, which includes three basic layers: the regional boundaries of the counties (3234 polygonal elements), rivers in the third level (14,096 line elements), and roads (49,454 line elements). All the data are located in China and the boundary is fixed, so we use (60°E, 0°N) as the original minimum position and (140°E, 80°N) as the original maximum position. Figure 13 displays the characteristic bitmaps of the regional boundaries of the counties using the QCODE method in the prototype system.

4.2.1. Experiment 1

This experiment is designed to evaluate the approximation ability of a QCODE characteristic bitmap. The proposed $t = Area_{QCODE}/Area_{BOX}$ is used as the indicator, where $Area_{QCODE}$ represents the area of the QCODE characteristic bitmap of the vector elements, and $Area_{BOX}$ represents that of the bounding box. The area ratio t is adopted because both the bounding box of the vector elements and the QCODE characteristic bitmaps completely cover the vector elements. The number of bytes in the data structure as occupied by the bounding box approximates that of the QCODE. Therefore, smaller areas approximate the original vector elements more precisely in an absolute sense. Figure 14 displays the

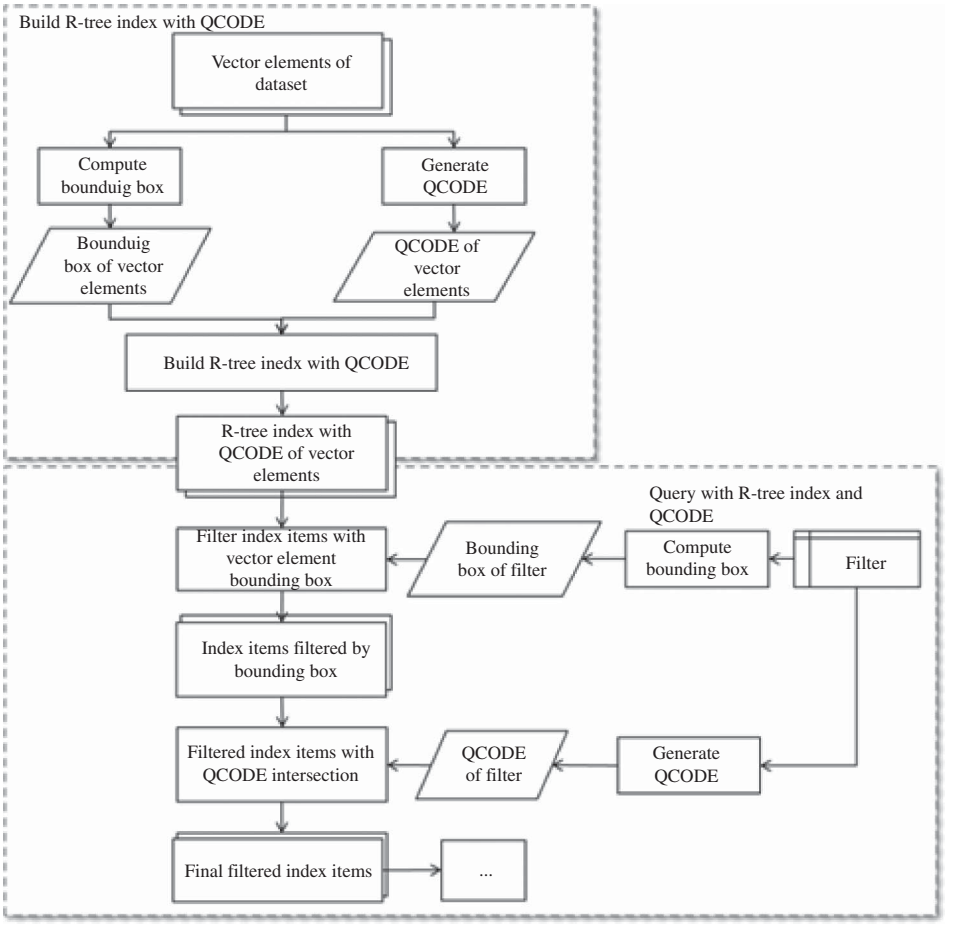

Figure 12. The process of the generation and usage of QCODE.

comparative results. In Figure 14, the X axis represents $Area_{QCODE}/Area_{BOX,}$ while the Y axis represents total record count (record count under the condition of $t < x$).

The result of the comparison indicates that the number of records, for $t < 1$, is greater than two-thirds of the total number of records. The above data demonstrate that the QCODE encoding improves the approximation of vector elements when the memory data are relatively similar in their amount.

4.2.2. Experiment 2

This experiment aims to simply compare the filtering accuracy rates of the bounding box with the optimisation by the QCODE method. Ten arbitrary lines and polygons that can act as filters are constructed by user interaction.

Table 1 demonstrates the filter results with lines, and Table 2 illustrates the filter results with polygons. As rivers in the third level and roads are all line elements, roads are omitted from this calculation. The area of the filter represents the area of the geometric object that

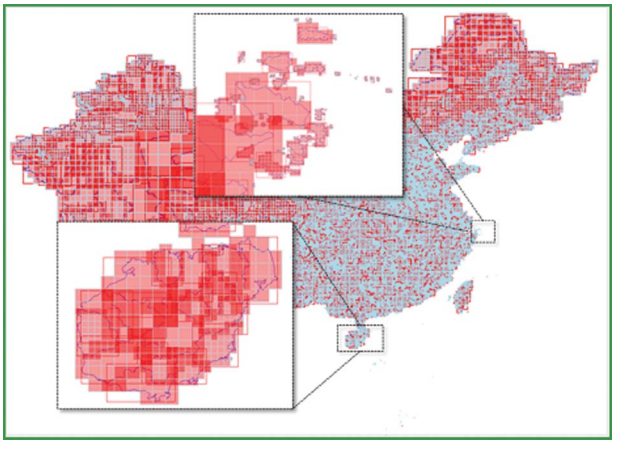

Figure 13. Visualisation of the characteristic bitmap of the regional boundaries of the counties.

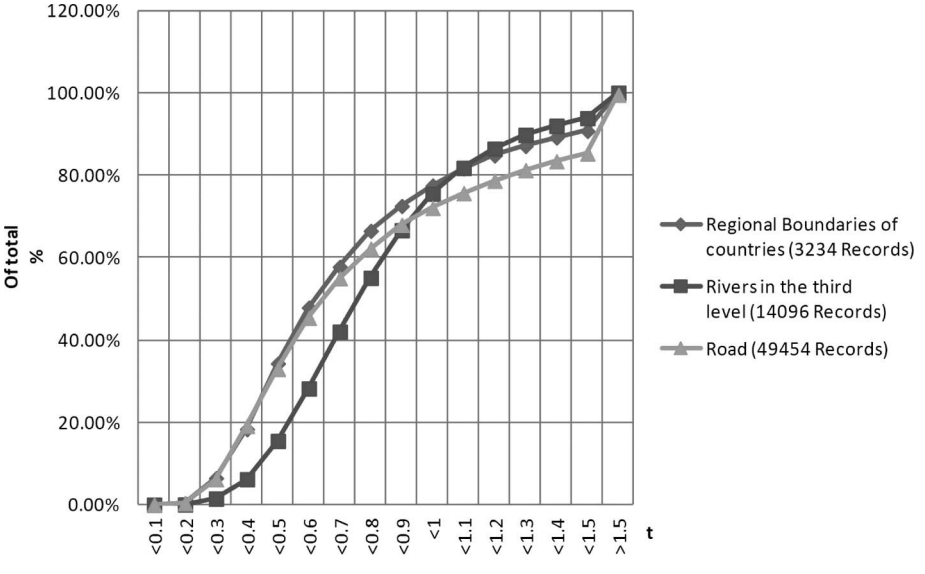

Figure 14. Comparison between $Area_{QCODE}$ and $Area_{BOX}$ ($Area_{QCODE}/Area_{Box}$).

was used as the filter, and there are two types: one is the area of the bounding box, and the other is the area of the QCODE bitmap. The former can be used for the judgment of intersections with bounding boxes of other vector elements, and the latter can be used for judgment of intersections with QCODE bitmaps of other vector elements. The record filter count A is calculated with the bounding box as the filter method, and the record filter count B has the values obtained from the optimisation using the QCODE method.

The results are ordered according to the filter area. The comparison shows that the optimisation using the QCODE method can significantly reduce the number of vector elements

Table 1. Filter results using the line elements.

	Filter area		With regional boundaries of counties			With rivers in the third level		
No.	ABB	AQB	RFCA	RFCB	B/A*100%	RFCA	RFCB	B/A*100%
1	0.3	0.1	7	5	71%	13	7	54%
2	1.1	0.6	16	11	69%	26	20	77%
3	2.1	1.7	20	17	85%	45	35	78%
4	4.3	2.3	37	23	62%	75	42	56%
5	11.9	4.3	84	37	44%	756	387	51%
6	16.3	10.9	112	50	45%	329	95	29%
7	19.1	14.1	109	60	55%	724	264	36%
8	31.9	28.1	211	133	63%	652	300	46%
9	36.6	14.1	220	82	37%	719	178	25%
10	55.5	31.3	331	198	60%	1777	907	51%
11	67.6	18.8	332	131	39%	1007	392	39%
12	84.6	32.8	391	208	53%	1116	445	40%
13	105.3	81.3	692	416	60%	2714	1134	42%
14	118.5	106.3	711	476	67%	2762	2199	80%
15	163.5	68.8	949	355	37%	4123	1327	32%
16	189.7	118.8	1085	606	56%	3872	2383	62%
17	203.0	93.8	1494	642	43%	5630	1848	33%
18	225.5	75.0	1288	454	35%	4731	1395	29%
19	257.4	87.5	1328	418	31%	5033	1506	30%
20	277.3	225.0	1486	831	56%	5412	2894	53%
Total (avg.)			10903	5153	53%	41516	17758	47%

Notes: ABB = Area of bounding box; AQB = Area of QCODE bitmap; RFCA = Record filter count A; RFCB = Record filter count B.

in the spatial relationship calculations, with reductions of approximately 50% for line elements and 20% for polygon elements, compared to the bounding box method. It is worth mentioning that different filter areas may lead to the grid cells of different sizes. The tables suggest that smaller grid cells will not necessarily enhance the filtering precision, while the characteristics of vector data might be the most important reason, i.e., the complexity of vector shape and variation of scale.

4.2.3. Experiment 3

This experiment aims to compare the execution time costs. T1 is used to record the time cost for executing the QCODE filter process, T2 is employed to record the time cost of the intersection operation without resorting to the QCODE method, while T3 is used to record the time cost of the intersection operation with QCODE method. The intersection operation is performed by importing a Geometry Engine - Open Source (GEOS)(http://geos.osgeo.org).

As seen from Tables 3 and 4, after being filtered by QCODE, the counts of selected vector elements are clearly reduced, and the time cost of the whole process is also reduced. For intersection operation with a line, the time cost is reduced by nearly 40%, and for intersection operation with a polygon, it is reduced by about 20%.

Table 2. Filter results using the polygon elements.

	Filter area		With regional boundaries of counties			With rivers in the third level		
No.	ABB	AQB	RFCA	RFCB	B/A*100%	RFCA	RFCB	B/A*100%
1	0.3	0.4	11	8	73%	11	10	91%
2	0.6	0.6	17	14	82%	38	33	87%
3	3.5	3.9	28	26	93%	48	47	98%
4	5.1	5.5	42	28	67%	119	78	66%
5	6.3	7.0	51	45	88%	155	129	83%
6	7.1	7.4	69	63	91%	238	189	79%
7	9.5	9.8	49	44	90%	82	75	91%
8	11.2	9.4	63	48	76%	140	62	44%
9	14.6	9.8	61	38	62%	154	110	71%
10	20.6	18.8	118	84	71%	986	877	89%
11	21.9	18.8	132	96	73%	378	257	68%
12	30.1	34.4	213	195	92%	690	624	90%
13	34.6	35.9	223	203	91%	727	634	87%
14	43.7	28.1	270	150	56%	832	314	38%
15	52.8	51.6	309	279	90%	1686	1576	93%
16	66.4	43.8	393	256	65%	2214	1427	64%
17	70.6	46.9	415	277	67%	2295	1491	65%
18	77.7	100.0	455	389	85%	2187	1927	88%
19	80.4	93.8	439	346	79%	1943	1444	74%
20	119.7	125.0	626	557	89%	2693	2397	89%
Total (avg.)			3984	3146	79%	17616	13701	78%

Notes: ABB = Area of bounding box; AQB = Area of QCODE bitmap; RFCA = Record filter count A; RFCB = Record filter count B.

5. Conclusions

This research designs a characteristic bitmap coding method for vector elements based on the self-adaptive gridding, i.e., the QCODE method. Comparison experiments demonstrate that the QCODE method is capable of spatial approximation, and QCODE uses less memory space to store the relevant information on vector elements. The R-tree method can be optimised taking the QCODE into consideration. Both methods can be effectively combined without damaging the R-tree structure, and the R-tree filtering precision also improves as a result.

However, to further refine this method, the following points must be addressed:

(1) A comparison indicates that the approximation ability of the vector elements using the QCODE method is affected by the complexity of shapes of the vector elements. The more complicated shapes of a vector element is approximated better by the QCODE method than by the bounding box method. Therefore, further studies are required to focus on searching for a critical balance between the QCODE method and the bounding box one, to explore a method of self-adaptive selection for an approximation method suitable for vector elements.

(2) The QCODE method involves approximate information, including the sizes and locations of the vector elements. In addition to evaluating the nature of simple

Table 3. Time cost comparison of the intersection operation with lines.

			With regional boundaries of counties						With rivers in the third level					
	Filter area		Vector element count		Time cost (ms)				Vector element count		Time cost (ms)			
No	ABB	AQB	C1	C2	T1	T2	T3	(T3 + T1)/T2*%	C1	C2	T1	T2	T3	(T3 + T1)/T2*%
1	1.5	0.8	18	11	0	71	48	67.6%	71	29	1	161	89	55.9%
2	1.8	0.8	18	9	0	61	34	55.7%	11	5	0	44	28	63.6%
3	2.3	1.0	28	18	0	74	49	66.2%	30	14	1	81	50	63.0%
4	6.8	4.3	61	38	1	209	141	67.9%	248	140	1	527	316	60.2%
5	10.7	4.7	88	48	0	317	180	56.8%	381	177	2	780	366	47.2%
6	13.9	4.7	114	44	0	436	193	44.3%	521	122	3	958	268	28.3%
7	19.3	17.2	127	98	1	474	402	85.0%	241	177	1	581	423	73.0%
8	26.6	17.2	177	97	1	577	354	61.5%	622	384	4	1284	838	65.6%
9	47.9	21.9	332	174	3	1205	625	52.1%	1153	507	8	2339	1184	51.0%
10	109.8	62.5	694	298	5	2302	1120	48.9%	2850	790	20	5273	1782	34.2%
Avg.								60.5%						54.2%

Notes: ABB = Area of bounding box; AQB = Area of QCODE bitmap; C1 = Vector Element Count Selected By Filter Box; C2 = Vector Element Count Selected By Filter QCODE; T1 = Time Cost of executing QCODE filter process; T2 = Time Cost By Intersection Operation Without QCODE Method; T3 = Time Cost By Intersection Operation With QCODE Method.

Table 4. Time cost comparison of the intersection operation with polygons.

			With regional boundaries of counties						With rivers in the third level					
	Filter area		Vector element count		Time cost (ms)				Vector element count		Time cost (ms)			
No	ABB	AQB	C1	C2	T1	T2	T3	(T3 + T1)/T2*%	C1	C2	T1	T2	T3	(T3 + T1)/T2*%
1	5.4	3.9	59	47	< 1	211	175	82.9%	165	111	1	315	231	73.7%
2	3.6	3.6	48	39	< 1	218	149	68.3%	162	132	2	342	279	82.2%
3	11.6	10.2	112	95	1	517	451	87.4%	306	277	2	597	533	89.6%
4	18.5	23.4	128	110	1	492	442	90.0%	255	204	1	657	536	81.7%
5	21.8	10.2	157	74	1	585	330	56.6%	465	202	3	1023	527	51.8%
6	16.3	16.0	92	82	1	485	451	93.2%	616	546	4	1278	1151	90.4%
7	30.9	23.4	182	125	1	689	513	74.6%	1126	991	8	2187	1867	85.7%
8	67.2	50.0	395	281	3	1621	1181	73.0%	2149	1813	15	4444	3693	83.4%
9	71.5	59.4	447	339	3	1897	1602	84.6%	2096	1397	15	4693	3274	70.1%
10	85.2	87.5	548	470	4	2252	1972	87.7%	2868	2058	21	6003	4388	73.4%
Avg.								79.9%						78.2%

Notes: ABB = Area of bounding box; AQB = Area of QCODE bitmap; C1 = Vector Element Count Selected By Filter Box; C2 = Vector Element Count Selected By Filter QCODE; T1 = Time Cost of executing QCODE filter process; T2 = Time Cost By Intersection Operation Without QCODE Method; T3 = Time Cost By Intersection Operation With QCODE Method.

intersections, it can be expanded, i.e., to record the complete 9-I model information or the efficiency of other topological operations. Future studies will also be required to focus on improving the data storage efficiency of the QCODE method.

(3) More experiments of empirical evaluations shall be required to further explore the relationship between the rasterisation resolution and the selectivity improvements. For example, different grid cell sizes of QCODE (e.g., 4×4, 16×16) will be employed to evaluate the essential effect of the rasterisation resolution to the precision and efficiency of filtering.

Acknowledgements

The authors appreciate the detailed suggestions and comments from the editor and the anonymous reviewers. The work described in this article was supported by The National Key Technology R&D Program of China (grant no. 2012BAH35B02), the National Natural Science Foundation of China (grant no. 41001223).

References

Al-Badarneh, A.F., Yaseen, Q., and Hmeidi, I., 2010. A new enhancement to the R-tree node splitting. *Journal of Information Science*, 36 (1), 3–18.

Arge, L., Berg, M.D., and Haverkort, J., 2005. Cache-Oblivious R-Trees [online]. Available from: http://www.win.tue.nl/~hermanh/stack/cache-oblivious-rtree.pdf [Accessed 1 February 2013].

Arge, L., Berg, M.D., and Haverkort, H.J., 2009. Cache-oblivious R-trees. *Algorithmica*, 53 (1), 50–68.

Arge, L., *et al*., 2008. *The priority R-tree: A practically efficient and worst-case optimal R-tree* [online]. Available from: http://delivery.acm.org/10.1145/1330000/1328920/a9-arge.pdf?ip=137.189.162.186&acc=ACTIVE%20SERVICE&CFID=114912840&CFTOKEN=59475025&acm__=1347592247_dc572e1bada1b41164c80381832cb292 [Accessed 1 September 2012].

Banerjee, S., *et al*., 2008. Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70 (4), 825–848.

Baum, M. and Hanebneck, U.D., 2010.Tracking a minimum bounding rectangle based on extreme value theory. *In*: *Proceedings of 2010 IEEE conference on multisensor fusion and integration for intelligent systems*. Salt Lake City, USA, 56–61.

Beckmann, N., *et al*., 1990. *The R*-tree: an efficient and robust access method for points and rectangles* [online]. Available from: http://delivery.acm.org/10.1145/100000/98741/p322-beckmann.pdf?ip=137.189.162.186&acc=ACTIVE%20SERVICE&CFID=114617608&CFTOKEN=10167590&__acm__=1347538389_9f1a794cec37a8ef1d4a57b637920e93 [Accessed 1 September 2012].

Beckmann, N., *et al*., 2009. A revised r*-tree in comparison with related index structures. *In*: *Proceedings of the 2009 ACM SIGMOD international conference on management of data*. New York, USA, 799–812.

Berman, P. and Raskhodnikova, S., 2010. Approximation algorithms for min-max generalization problems. *Lecture Notes in Computer Science*, 6302, 53–66.

Bivand, R.S., Pebesma, E.J., and Gómez-Rubio, V., 2008. *Applied spatial data analysis with R*. New York, NY: Springer Science+Business Media, LLC.

Brisaboa, N.R., *et al*., 2010. Range queries over a compact representation of minimum bounding rectangles. *Advances in Conceptual Modeling – Applications and Challenges*, Lecture Notes in Computer Science, 6413, 33–42.

Chaudhuri, D., *et al*., 2012. Finding best-fitted rectangle for regions using a bisection method. *Machine Vision and Applications*, 23 (6), 1263–1271.

Clifford, A.S., 2011. *Data structures and algorithm analysis. Edition 3.2 (C++ Version)*. Blacksburg: Department of Computer Science Virginia Tech.

Colak, E., 2002. *Portable high-performance indexing for vector product format spatial databases*, M.S. Thesis. Airforce Institute of Technology.

Gibson, L. and Lucas, D., 1982. Vectorization of raster images using hierarchical methods. *Computer Graphics and Image Processing*, 20 (1), 82–89.
Gold, C., 2009. A common spatial model for GIS. *Lecture Notes in Geoinformation and Cartography*, 2, 79–94.
Goodchild, M.F., Haining, R., and Wise, S., 1992. Integrating GIS and spatial data analysis: problems and possibilities. *International Journal of Geographical Information Systems*, 6 (5), 407–423.
Guting, R.H., 1994. An introduction to Spatial Database Systems. *VLDB Journal*, 3, 357–399.
Guttman, A., 1984. R-Trees: A dynamic index structure for spatial searching. *In*: *Proceedings of the Special Interest Group on Management of Data*. Boston, USA, 47–57.
Harley, M. and Gellerstedt, B., 2005. *The role of grid size optimization in ArcSDE performance tuning* [online]. Available from: http://proceedings.esri.com/library/userconf/proc05/papers/pap1858.pdf [Accessed 20 November 2012].
Kuan, J. and Lewis, P. 1997. Fast K nearest neighbour search for R-tree family. *In*: *Proceedings of international conference on information, communications and signal processing*. Singapore, 924–928.
Lane, J. and Carpenter, L., 1979. A generalized scan linealgorithm for the computer display of parametrically defined surfaces. *Computer Graphics and Image Processing*, 11 (3), 290–297.
Laurini, R., 1998. Spatial multi-database topological continuity and indexing: a step towards seamless GIS data interoperability. *International Journal of Geographical Information Systems*, 12 (4), 373–402.
Lindholm, E., *et al*., 2008. Vectorization of raster images using hierarchical methods. *IEEE Micro*, 28 (2), 39–55.
Maheshwari, A., Vahrenhold, J., and Zeh, N. 2002. *On reverse nearest neighbor queries* [online]. Available from: http://people.scs.carleton.ca:8008/~maheshwa/papers/CCCG-rev-nn-2002.pdf [Accessed 30 August 2012].
Mamoulis, N., 2011. Spatial data management. *Synthesis Lectures on Data Management*, 3 (6), 1–149.
Microsoft. 2012. *Microsoft MSDN* [online]. Available from: http://msdn.microsoft.com/en-us/library/29dh1w7z(v=vs.80).aspx [Accessed 1 September 2012].
Molet, M., Jozefowiez, J., and Miller, R.R., 2010. Integration of spatial relationships and temporal relationships in humans. *Learning & behavior*, 38 (1), 27–34.
Novak, J. and Dachsbacher, C., 2012. *Rasterized bounding volume hierarchies* [online]. Available from: http://cg.ibds.kit.edu/publications/p2012/RBVH/RBVH.pdf [Accessed on 20th November 2012].
Ooi, B.C., Davis, R.S., and Han, J.W., 1993. *Indexing in spatial databases* [online]. Available from: http://www.comp.nus.edu.sg/~ooibc/spatialsurvey.pdf [Accessed on 13th January 2013].
Pilevar, A.H. and Sukumar, M., 2005. GCHL: A grid-clustering algorithm for high-dimensional very large spatial data bases. *Pattern Recognition Letters*, 26 (7), 999–1010.
Poiani, K.A., Bedford, B.L., and Merrill, M.D., 1996. A GIS-based index for relating landscape characteristics to potential nitrogen leaching to wetlands. *Landscape Ecology*, 11 (4), 237–255.
Pramanik, S. and Li, J., 1999. *Fast approximate search algorithm for nearest neighbor queries in high dimensions* [online]. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00754931 [Accessed 15 August 2012].
Rigaux, P., Scholl, M., and Voisard, A., 2002. *Spatial Databases: With application to GIS*. San Francisco, CA: Morgan Kaufmann Publishers.
Schneider, P.J. and Eberly, D.H., 2003. *Geometric tools for computer graphics*. San Francisco, CA: Morgan Kaufmann Publishers.
Sellis, T., Roussopoulos, N., and Faloutsos, C., 1987. *The R + −tree: A dynamic index for multi-dimensional objects* [online]. Available from: http://repository.cmu.edu/cgi/viewcontent.cgi?article=1563&context=compsci [Accessed 1 September 2012].
Shekhar, S. and Chawla, S., 2003. *Spatial databases*. Upper Saddle River, NJ: Prentice Hall.
Siqueira, T.L.L., *et al*., 2012. The SB-index and the HSB-index: efficient indices for spatial data warehouses. *GeoInformatica*, 16 (1), 165–205.
Wang, P.T., *et al*., 2003. A resolution-driven generalization approach for linear and areal objects. *In*: *Proceedings of 2003 IEEE international conference of geoscience and remote sensing symposium*. France, 2329–2331.

Zhang, Y., *et al*., 2011. *A grid-aided and STR-Tree-based algorithm for partitioning vector data* [online]. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5980718 [Accessed 11 August 2012].

Zhao, L.L., Liu, S.A., and Li, J.S., 2012. The index storage model for urbanization data combined query. *Advanced Materials Research*, 1186, 532–533.

Zheng, B., *et al*., 2006. Grid-partition index: a hybrid method for nearest-neighbor queries in wireless location-based services. *The International Journal on Very Large Data Bases*, 15 (1), 21–39.

Zhu, Q., *et al*., 2003. An oracle-based data management method for large database in CyberCity GIS. *Geo-Spatial Information Science*, 6 (4), 39–43.

Zhu, Q., Gong, J., and Zhang, Y.T., 2007. An efficient 3D R-treespatialindex method for virtual geographic environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62 (3), 217–224.