Computing and Informatics, Vol. 32, 2013, 679–696

QMBR$^{\mathbf{I}}$: INVERSE QUANTIZATION OF MINIMUM BOUNDING RECTANGLES FOR SPATIAL DATA COMPRESSION

Jongwan Kim, Dukshin Oh

Department of Management Information Systems
Sahmyook University, Seoul, Korea
e-mail: wany@korea.ac.kr, ohds@syu.ac.kr

Keecheon Kim[*]

Department of Computer Science and Engineering
Konkuk University
Seoul, Korea
e-mail: ckkim@konkuk.ac.kr

Abstract. In this paper, we propose QMBR^i, the inverse representation of the quantized minimum bounding rectangles (MBRs) scheme, which compresses a minimum bounding rectangle key into one byte for spatial-data compression. QMBR^i is a novel spatial-data compression scheme that is based on inverse quantization and overcomes the shortcomings of conventional relative coordination or quantization schemes. If a spatial data is far from the starting point of the search region, the relative coordination scheme does not guarantee compression. In a quantization scheme, since the MBRs are expanded, the overlapping of MBRs is increased and the search performance is reduced. The proposed scheme overcomes these shortcomings, and simulation results suggest that it performs better than other schemes.

Keywords: Spatial data, spatial-data compression, MBR, RMBR, HMBR, QMBR

[*] corresponding author

1 INTRODUCTION

The R-tree [1] approximates spatial objects as minimum bounding rectangles(MBRs) such as R_2 in Figure 1. Since an R-tree is a disk-based index, the size of the coordinates, x and y, greatly affects the search performance. Therefore, much research has been undertaken on spatial data compression schemes that reduce the sizes of coordinates in order to improve the performance of geographical information systems (GISs) or location-based services (LBSs) [2, 3, 4, 11].

Existing spatial data compression schemes can be categorized into relative coordinate and quantization schemes. The former comprises the relative MBR (RMBR) and hybrid MBR (HMBR) schemes [5]. The disadvantage of these schemes is that compression is not possible if the spatial object is far from the starting-point of the search region because the relative coordinates are then too large. This disadvantage is inherent in all relative-coordinate-based compression schemes. The compressed sizes of RMBR and HMBR are eight and six bytes, respectively. Another scheme is the quantized MBR (QMBR), which compresses spatial data into four bytes [5]. In the quantization-based compression scheme, the size of real MBRs increases and they overlap. This increases the number of node accesses during the search of spatial data and slows processing performance.

In this paper, we propose a new compression scheme, QMBR^i, which overcomes the disadvantages of the existing compression schemes described above. QMBR^i compresses MBR coordinates in two-dimensional space into one byte, as in the conventional QMBR scheme. Although the division of the search space through quantization is the same as in the QMBR scheme, since the quantization value is not used as a coordinate, the MBRs neither expand nor overlap. QMBR^i is an inverse quantization scheme that creates new coordinates from the quantization value. It is believed that the proposed scheme may be the first attempt to compress spatial data into one byte through inverse quantization.

The main contributions of this paper are as follows.

- If the relative coordinates exceed two bytes, the spatial data are not compressed. This is resolved using the virtual search region in QMBR^i. That is, the coordinates, which are far from the start point of the search region, are compressed.
- QMBR^i expands only two sides of MBR. Therefore, the overlap, a shortcoming of the QMBR scheme, is decreased in QMBR^i and performance improves over that of QMBR.

This paper is organized as follows. In Section 2, we examine the issues that affect existing MBR compression schemes. In Section 3, we explain the details of inverse quantization, which is the basis of the new compression scheme. Section 4 analyzes the structure of indices with the QMBR^i scheme. Section 5 describes the performance evaluation of the QMBR^i scheme and Section 6 presents the conclusions.

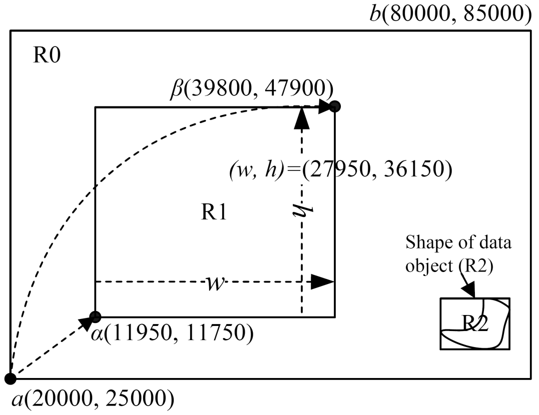


Figure 1. RMBR and HMBR

2 MBR COMPRESSION AND PROBLEMS

In this section, using the following terminology, we discuss the issues that affect existing compression schemes. For the search region, R_0, we select two points, a and b (see Figure 1). The internal MBRs contained in the search region are numbered $R_1, R_2, \ldots, R_n$. The starting- and end-points are generically denoted by rs and $re(s : start; e : end)$, respectively. According to the number of internal MBRs (viz., n), these points are designated as $r1s$, $r1e$, …, rns, and rne. The two compressed points are α and β, and each point comprises the coordinates, (x, y). Generally, one MBR stores four coordinates in two points and has the size of 16 bytes.

As shown in Figure 1, the RMBR calculates the relative coordinates of R_1 from a. The coordinates are compressed from 16 to 8 bytes. However, when the relative coordinates are greater than two bytes due to widening of the search region, there is no compression effect. For example, if the search region is wider and $r1e$ is $(87\,000, 72\,900)$, the relative coordinate of β in the x axis is $67\,000$. Consequently, there is no compression effect because the relative coordinates exceed two bytes. This also applies to α in the same way. In the HMBR scheme, as in the RMBR scheme, the starting-point, $r1s$, is compressed in terms of relative coordinates. However, if R_1 remains far from the vertex, a, of R_0, the coordinates of $r1s$ are not compressed into two bytes. The end-point, β, evaluates the height (h) and the width (w) in relation to α rather than the starting-point of R_0. Therefore, the coordinates of the end-point are compressed into one byte and stored in fewer bytes than in the RMBR scheme [5]. The purpose of this paper is to improve the search performance by compressing spatial data. Spatial data compression schemes that are based on relative coordinates have the following drawback (Observation 1), which we attempt to overcome.

Observation 1. The relative-coordinate schemes achieve compression only if the offsets between the coordinates of R_0 and the MBRs are contained in either one or two bytes.

Let R_1 and R_2 be the MBRs contained in the search region, R_0, and let them have two sets of points $r1s \,\&\, r2s$ and $r1e \,\&\, r2e$, respectively. We assume that R_{1x} and R_{2x} are adjacent to vertices, a and b, respectively, of R_0. Then R_2, which is far from vertex a of R_0, will have larger relative coordinates than R_1. If the size of the relative coordinates, $(rs_x - a_x)$, exceeds two bytes, the RMBR is not compressed. This also applies to the compression of rs in the HMBR scheme. In the RMBR scheme, the two compressed points, α and β, of R_1 are as follows,

$$\begin{aligned} \alpha_x(R_1) &= |a_x - rs_x|\,, & \alpha_y(R_1) &= |a_y - rs_y| \\ \beta_x(R_1) &= |a_x - re_x|\,, & \beta_y(R_1) &= |a_y - re_y|\,. \end{aligned} \tag{1}$$

QMBR quantizes the search region, as shown in Figure 2. It substitutes the quantization value for the coordinates of an MBR and stores the MBR in four bytes. Compared to the eight- or six-byte compression of existing schemes, the compression ratio of QMBR is relatively high. However, QMBR has the disadvantage described in Observation 2.

Observation 2. MBRs that are compressed by the quantization scheme extend into four directions, and the overlap ratio increases.

Let l be the quantization level and $q_1, q_2, \ldots, q_n$ be the values of quantization that divide the search region R_0 into a grid. In Figure 2, the two points of R_1, ($r1s$ and $r1e$) are substituted by the quantization value and become $\alpha(2,1)$ and $\beta(10,8)$. At that point, R_1 overlaps R3 while it extends. This overlap reduces the search performance in the spatial index.

The following is an example of Observation 2. Assume that R3 contains a data object from Figure 2, and that R_1 and R_3 reside on nodes A and B, respectively. In the R-tree, the search query approaches node B to find the data object. However, in the QMBR, backtracking occurs because R_1 contains a data object with expanded regions. That is, the query approaches node A, moves to the upper node again, and then approaches node B to find the data object. Consequently, the increased number of node accesses reduces the search performance.

3 MBR COMPRESSION USING INVERSE QUANTIZATION

Spatial data compression affects the index size and search performance. In this paper, compression is made on the basis of the QMBR. However, this does not extend to the quantization value on each side. The new coordinates that are compressed into one byte are created on the basis of the nearest quantization value. In other words, QMBR^i compresses the spatial data by using inverse quantization. Since QMBR^i transforms the offsets between the coordinates of the search region and the MBRs into 1 byte, Observation 1 is resolved.

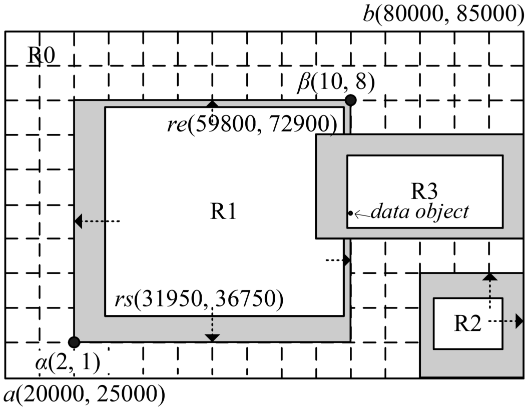


Figure 2. Expansion of MBRs in QMBR

Definition 1 (The virtual search region)**.** Let ds be the dataset, which is comprised of MBRs and let R_0 be the search region. R_0 is the widest MBR in ds and is defined by Equation (2). The two points of R_0 are a and b. To compress the MBR into one byte, the virtual search region (VSR), which is converted so searching can commence from the coordinates of $(0, 0)$, is defined by Equation (3). The virtual search region is vR_0 and the size of the region is the same as that of R_0.

$$\forall_{MBR} \subset R_0,$$

$$\begin{aligned} MBR(R_0) &= \{(a, b) | a = \text{MIN}(ds.rs), b = \text{MAX}(ds.re)\} \\ rs(R_0) &= \{(x_1, y_1) | x_1 = \text{MIN}(ds.x_1), y_1 = \text{MIN}(ds.y_1)\} \\ re(R_0) &= \{(x_2, y_2) | x_2 = \text{MAX}(ds.x_2), y_2 = \text{MAX}(ds.y_2)\}. \end{aligned} \tag{2}$$

$$\begin{aligned} a(vR_0) &= (rs_{x1} - rs_{x1}, rs_{y1} - rs_{y1}) \\ b(vR_0) &= (re_{x2} - rs_{x1}, re_{y2} - rs_{y1}). \end{aligned} \tag{3}$$

Definition 1 converts the starting-point, a, of the search region into the coordinates, $(0, 0)$, of a virtual region, and converts b as well to compress each coordinate into one byte. The reason for converting to a virtual region is to avoid negative numbers when calculating the compressed coordinates based on the quantization value. The MBRs in the dataset are converted into vMBRs (virtual MBRs) and then included in the VSR.

Definition 2 (MBR quantization value)**.** Let vR_0 be the VSR and vR_1 be the MBR contained in vR_0. The quantization levels of the x- and y-axes are l^x and l^y, respectively. Then, vR_0 has two points, a and b, and vR_1 has rs and re. The quantization values, q_{rs}, and q_{re}, are substituted for the two points of vR_1 as follows. The quantization values are applied to inverse quantization.

$$\begin{aligned} q^x_{rs} &= \lceil (rs_x - a_x)/(b_x - a_x) * l^x \rceil, \quad q^y_{rs} = \lceil (rs_y - a_y)/(b_y - a_y) * l^y \rceil \\ q^x_{re} &= \lceil (re_x - a_x)/(b_x - a_x) * l^x \rceil, \quad q^y_{re} = \lceil (re_y - a_y)/(b_y - a_y) * l^y \rceil. \end{aligned} \tag{4}$$

Definition 3 (Inverse quantization)**.** Let vR_1 be the MBR contained in the VSR and *qsize* be the size between q_n and q_{n+1} of quantization value. As shown in (5), the coordinates of the two points of vR_1 ($r1s$ and $r1e$) are inversely quantized as shown in (5) on the basis of the nearest quantization value. The two compressed points are α and β.

$$\begin{aligned} \alpha_x &= q^x_{rs} \times qsize^x - rs_x, \quad \alpha_y = q^y_{rs} \times qsize^y - rs_y \\ \beta_x &= q^x_{re} \times qsize^x - re_x, \quad \beta_y = q^y_{re} \times qsize^y - re_y. \end{aligned} \tag{5}$$

Inverse quantization in the search region is performed as shown in Figure 3. First, quantization occurs after the conversion of R_0 in Figure 2 into vR_0 that starts at $(0,0)$. On the basis of each coordinate of vR_1 and the nearest quantization values, viz., q^x_3, q^x_{10}, q^y_2, and q^y_8, new coordinates are calculated via Equation (5). Then, vR_1 is compressed into one byte from $(11\,950, 11\,750)$–$(39\,800, 47\,900)$ to $(50, 250)$–$(200, 100)$; this is referred to as *inverse quantization.*

QMBR has two characteristics. One is that QMBR compresses the spatial data into 4 bytes from 16 bytes and space is saved. The other is that MBR is expanded and overlaps other MBRs in QMBR. The main contribution of QMBR^i is to prevent MBR expansion and overlap. QMBR is expanded on four sides. However, since QMBR^i is expanded on two sides, such as in Figure 3, the overlap with MBRs decreases. Therefore, inverse quantization minimizes MBR overlap, thereby overcoming the disadvantage of the QMBR scheme. If the overlap decreases, search performance is superior to that of QMBR. This is analyzed in Section 4 and shown by simulation. As we know, it is important to decrease overlap to improve performance.

Figure 4 is a spatial-data compression algorithm that uses inverse quantization and a MATLAB-based [10] pseudo code. In this paper, only the compression algorithm for the x-axis coordinates is shown. The calculation of the y-axis is straightforward. In the algorithm, the search region is vR_0 and represents two points, a and b; rs and re represent the lower-left and upper-right corners of the MBR that is contained in the search region. These two points are converted to the coordinates of vR_1, namely, rs_x_new and re_x_new, and compressed to α_x and β_x, respectively.

Line 7 shows that the dataset is stored in the array from a data file. To create the VSR, the smallest coordinate, a_x, and the largest coordinate, b_x, in the dataset are found (lines 8–10). The x-axis of a and b of the VSR is created by using the maximum and minimum coordinates (line 12). To quantize the VSR, we define the quantization level of the x-axis as ranging from 1 to the maximum value, and then calculate the quantization size (lines 13–14). We convert each coordinate of the MBRs that are to be included in vR_0 and calculate the quantization value of the two points, rs and re (lines 15–19). Then, we obtain α_x and β_x through inverse

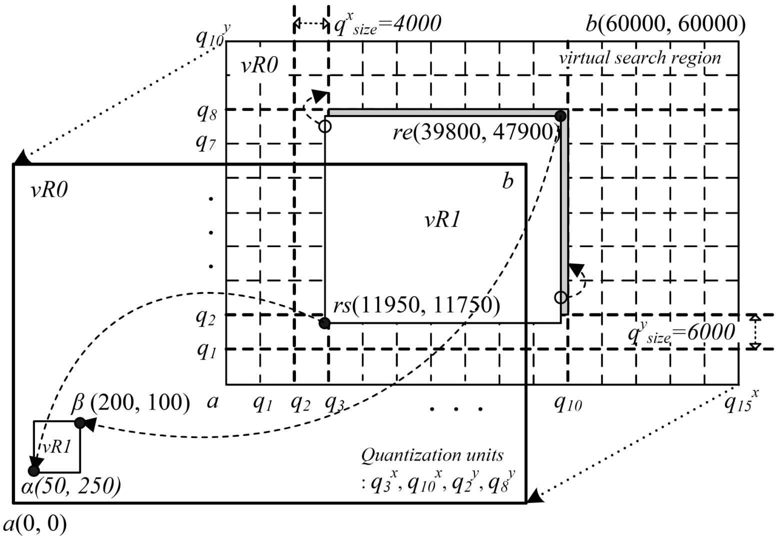


Figure 3. Inverse quantization MBR in VSR

quantization by compressing the coordinates of the x-axis (lines 21–22). Finally, lines 24–28 check to see whether the x-coordinates of α and β are within the range of one byte. The algorithm terminates when all the coordinates of the MBRs are compressed into one byte each.

4 INDEX STRUCTURAL ANALYSIS

4.1 The Index Size

The MBR, RMBR, HMBR, and QMBR compression schemes have stored-key sizes of 16, 8, 6, and 4 bytes, respectively. The proposed QMBR^i also has a four-byte key size. When each scheme is stored on disk, the index size is as shown in Equation (6). In Equation (6), we assume that the size of an MBR is 16 bytes and that the pointers for sub-nodes are four bytes in size. We also assume that the number of data objects, n, is 1 000 000, the node size, N, is 4 096 bytes, and the node usability, u, is 90 %. The value of m is the maximum fan-out of the nodes. According to Equation (6), the index size of the R-tree is 22.43 MB and that of QMBR^i is 19.33 MB. Since the index space that is calculated by the equation represents only a simple structural analysis, QMBR and QMBR^i are identical in the size of the compressed data. However, the search performance of the QMBR scheme is inferior to that of the QMBR^i scheme due to the overlapping MBRs.

$$IndexSize = \begin{cases} \frac{nN}{um}, & \text{if leaf node} \\ \frac{nN}{um(um-1)}, & \text{otherwise.} \end{cases} \tag{6}$$

```
1. **Algorithm**: Inverse quantization for the x-axis of MBRs
2. **Procedure** CompressSpatialData (*Dataset*)
3. **Input**: *Dataset*, a set of spatial coordinates
4. **Output**: Compressed coordinates in one byte
5. **Begin**
6.      $min\_a_x$, $max\_b_x$ ← initialize to zero;
7.      [$rs_x$, $re_x$, $rs_y$, $re_y$] ← set *Dataset* to MBR array;
8.      **for** (i = 1 : NumOfDataset)    // *search area R0*
9.      Get minimum $a_x$ ($min\_a_x$) from  MBR array;
10.     Get maximum $b_x$ ($max\_b_x$) from  MBR array;
11.       // *conversion from R0 to virtual search region vR0*
12. $a_x$ ← $min\_a_x$ - $min\_a_x$; $b_x$ ← $max\_b_x$ - $min\_a_x$;
13. **for** (qlevel_x=1 : $max\_b_x$)
14.     $qsize^x$ ← Ceiling(($b_x$ - $a_x$) / qlevel_x) for the size of a unit;
15.     **for** (i = 1 : SizeOfDatasets)
16.         $rs_x\_new$ ← the vMBR's start x-coordinate;
17.         $re_x\_new$ ← the vMBR's end x-coordinate;
18.         $q_{rs}^x$ ← the quantization unit of the starting-point, $rs_x$;
19.         $q_{re}^x$ ← the quantization unit of the end-point, $re_x$;
20.             // *compressed coordinate of vMBR's x-axis*
21.         $\alpha_x[i]$ ← $q_{rs}^x$ x $qsize^x - rs_x\_new$;
22.         $\beta_x[i]$ ← $q_{re}^x$ x $qsize^x - re_x\_new$;
23.             // *confirm* $\alpha_x$ *and* $\beta_x$ *in 1 byte compression*
24.         **if** ((($\alpha_x[i]$ >= 0) & ($\alpha_x[i]$ <= 255))
25.                                     & (($\beta_x[i]$ >= 0) & (($\beta_x[i]$ <= 255)))
26.         **then**
27.             *intOneByteCount*++;
28.             **if** (*intOneByteCount* == *SizeOfDatasets*)
29.             **then** return 0;
30. **End**
```

Figure 4. Inverse quantization algorithm

4.2 The Number of Node Accesses

We measured the number of node accesses by using the equation summarized in [6]. The search performance is associated with the tree height, which depends on the number of entries in a node. If more entries are stored in one node, fewer disk access operations are required to access them. We assume a node size of 4 096 KB, which is the disk allocation unit. In the R-tree, the number of nodes in a height, h, $node_h$, is $\lceil n/fanout^h \rceil$ and n is the total number of data objects. The number of node accesses at a height, h, can be determined on the basis of the probability of meeting the average region and the query region that is occupied by a node of the relevant height. We assume $area_h$ to be the average size of one node of height, h. The region is overlapped by $area_h$ and the query size, qs, is $(\sqrt{qs} + \sqrt{area_h})^2$, according to the Minkowski sum [7]. Figure 5 shows the probability of the Minkowski sum, which is the average size of the node that the query object encounters. Finally, qs, which is overlapped by the nodes at a height, h, is the same as the expression in parentheses in Equation (7). The total number of node accesses from the leaf node to the root node of the R-tree becomes equal to the accumulated number of node accesses at each height, as shown in Equation (7).

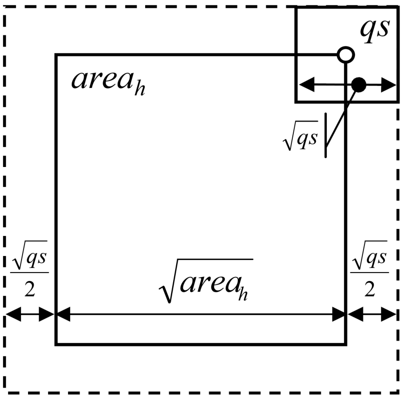


Figure 5. The average area of a node with the Minkowski sum

$$NodeAccess(R\text{-tree}) = 1 + \sum_{h=1}^{\lceil \log_f N \rceil + 1} \left(1 + \sqrt{\left\lceil \frac{N}{f^h} \right\rceil \cdot s}\right)^2. \tag{7}$$

The number of node accesses in the QMBR becomes greater than that of the real MBR. This is because the MBR extends to the quantization value. The probability of the number of node accesses in the QMBR is described below. When the quantization level is q, then each node at a height, h, is divided into q^2 cells. One side of each cell is $\sqrt{area_h}/q$ in length. The number of node accesses depends on how much the query MBR at the upper node and the average region of the node overlap when a node is approached. The number of node accesses of query MBR at height, h, should add the probability of node overlapped with quantization cell at upper node to the probability of node accesses. Therefore, the node accesses in the quantized region are as shown in Equation (8). The query MBR should be quantized when the data are searched.

$$NodeAccess(QMBR) = 1 + \sum_{h=1}^{\lceil \log_f N \rceil + 1} \left(1 + \sqrt{\left\lceil \frac{N}{f^h} \right\rceil \cdot s} + \sqrt{\left\lceil \frac{N}{f^{h+1}} \right\rceil \cdot s/q} \right)^2 . \quad (8)$$

Figure 6 shows the mathematical results for the number of node accesses of each scheme. Those of the MBR, RMBR, and HBMR schemes are calculated by Equation (7), whereas that of the QMBR is as shown in Equation (8). Figure 6 also shows the decrease in the number of node accesses as the node size increases. When the node size is large, the number of entries in a single node increases and the probability of finding an entry in a node also increases. The number of node accesses decreases almost linearly; a similar pattern appears in the synthetic dataset. Another case where there are a large number of entries in a node is when the stored key is small. This is the compressed case. When the key is small, the number of node accesses decreases as more entries can be stored in the same node. The QMBR and QMBR^i schemes appear almost identical in this respect due to the simple calculation via the equation. However, the results from the simulation are different. This is explained in Section 5.

5 PERFORMANCE EVALUATION

We evaluated the performance of the RMBR, HMBR, QMBR, and QMBR^i schemes in two parts. The first was to implement an index and the second was to convert the dataset to compressed coordinates. The index for each compression scheme was programmed in C++ as a modification of the *R*-tree, and coordinate compression was performed in MATLAB. The experiment was performed on a PC with 1 GB of memory and running on Windows XP.

We simulated two datasets, one synthetic and the other real. The synthetic dataset of 100 000 MBRs was generated by DaVisualCode [8] (Figure 7). For the synthetic dataset, the MBRs were generated randomly from 100 000 × 100 000 regions. The distribution of the sample is skewed as in a real dataset. With regard to the real dataset, 62 556 real *Sequoia-in-California* MBRs [9] were used to evaluate the compression schemes (Figure 8). The dataset details are shown in Table 1.

The key of performance is the number of node accesses in the index. The search performance becomes faster than before compression when the number of entries in a node increases and the height of the tree decreases. This is a common characteristic of each compression scheme. However, the performance depends on the particular characteristics of each scheme.

The simulation involved accumulating the results of 30 query workloads for a point query in both the synthetic and real datasets. The extensive results for the

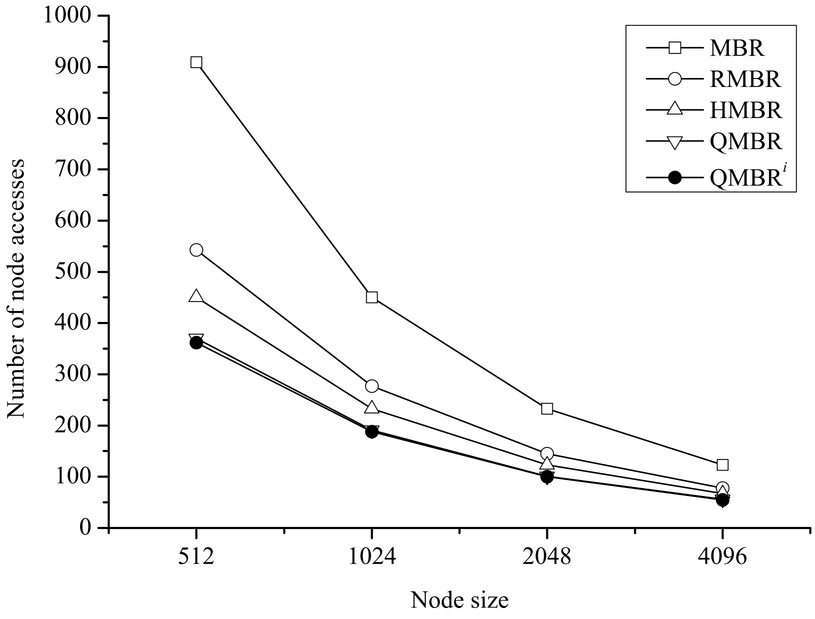


Figure 6. The number of node accesses as per mathematical analysis

Category	Synthetic dataset	Real dataset
# of MBRs	100 000	62 556
Distribution	Skewed	Skewed
Region	100 K × 100 K pixel	1 400 K × 2 100 K
Type	Synthetic	Sequoia groove
Compression size (Bytes)	MBR(16B), RMBR(8B), HMBR(6B), QMBR(4B), QMBR^i(4B)	
Node size	512, 1 024, 2 048, 4 096 bytes	

Table 1. Summary of datasets and compression schemes

synthetic dataset are shown in Figure 9. *Visit* represents the current entry and the node visited to search for the query data. *Leaf node* is an MBR that is found at the final level of the index via intermediate nodes. *Ent, NodeAcc*, and *BTrack* are the entry number that is visited; the number of node accesses; and the number of backtracks, respectively. In Figure 9, the MBR search starts with a visit to the first entry of the root node. At that point, the size of each node is 4 096 bytes and the levels from the leaf nodes to the root node are 0, 1, and 2. As shown in line 17, there are 201 entries in the leaf node. In Figure 9, the number of node accesses is 124 and the number of backtracks is 17. Figures 10 and 11 show the results. *Backtracking* in line 14 is the result of moving to the upper node again to search target data at other nodes. We do not consider the cache.

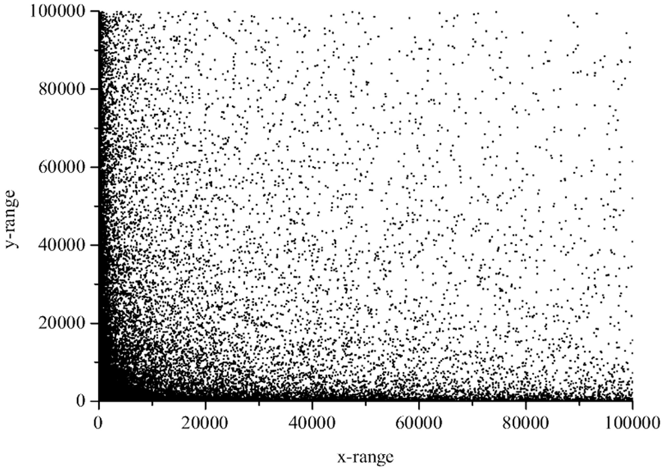


Figure 7. A synthetic dataset

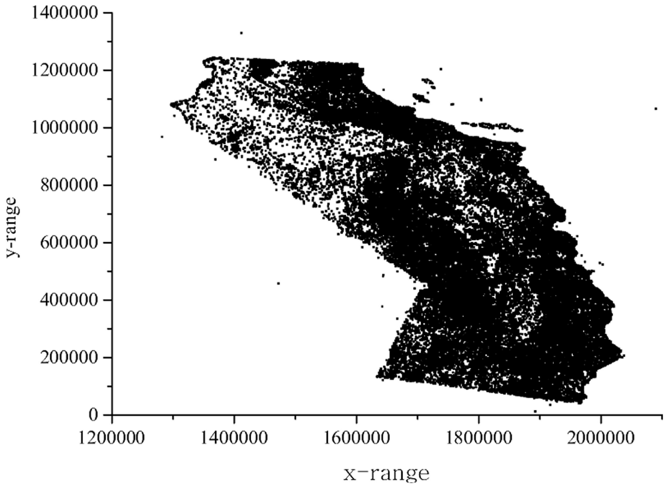
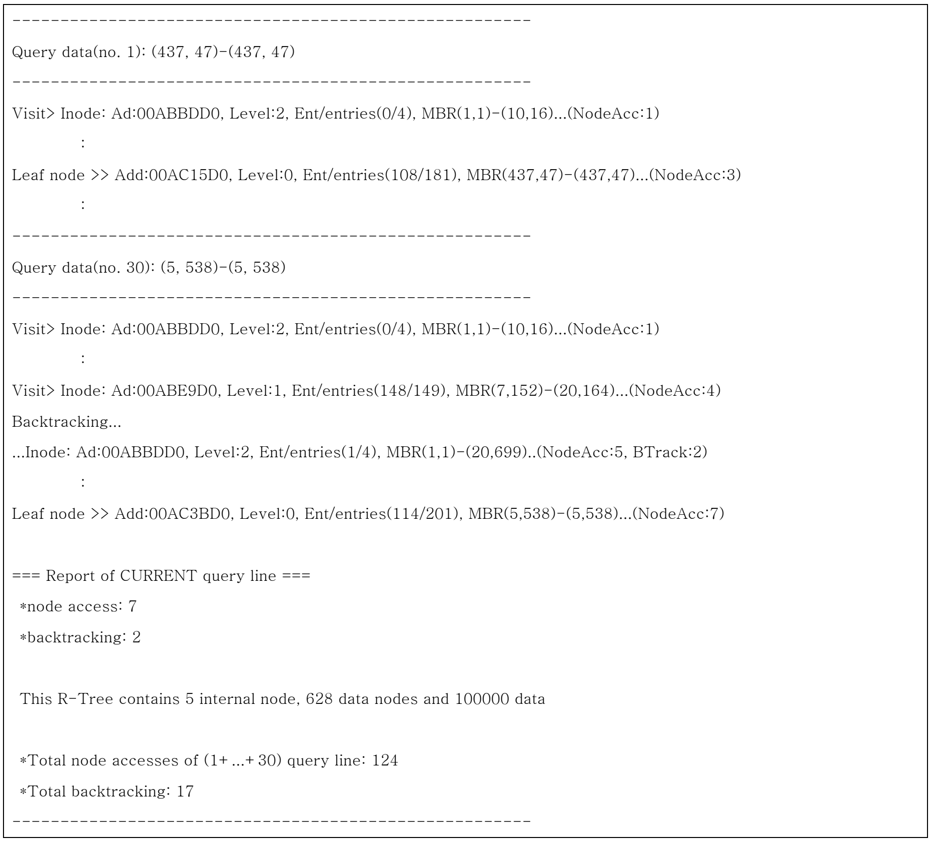


Figure 8. A real dataset (California Sequoia)

Figure 10 shows the number of node accesses when the compression scheme is applied to the randomly-created synthetic dataset. Since the number of node entries is increased by compression, the compression schemes have fewer node accesses than uncompressed MBRs of the same node-size. The 4 096-byte node shows fewer accesses than the 512-byte node because of the increased node-size and the improved node-usability. However, even though the node usability improves with QMBR compression, there are more node accesses as node-backtracking increases because the MBRs extend and overlap. Therefore, the QMBR scheme does not



```
----------------------------------------------------
Query data(no. 1): (437, 47)-(437, 47)
----------------------------------------------------
Visit> Inode: Ad:00ABBDD0, Level:2, Ent/entries(0/4), MBR(1,1)-(10,16)...(NodeAcc:1)
          :
Leaf node >> Add:00AC15D0, Level:0, Ent/entries(108/181), MBR(437,47)-(437,47)...(NodeAcc:3)
          :
----------------------------------------------------
Query data(no. 30): (5, 538)-(5, 538)
----------------------------------------------------
Visit> Inode: Ad:00ABBDD0, Level:2, Ent/entries(0/4), MBR(1,1)-(10,16)...(NodeAcc:1)
          :
Visit> Inode: Ad:00ABE9D0, Level:1, Ent/entries(148/149), MBR(7,152)-(20,164)...(NodeAcc:4)
Backtracking...
...Inode: Ad:00ABBDD0, Level:2, Ent/entries(1/4), MBR(1,1)-(20,699)..(NodeAcc:5, BTrack:2)
          :
Leaf node >> Add:00AC3BD0, Level:0, Ent/entries(114/201), MBR(5,538)-(5,538)...(NodeAcc:7)


=== Report of CURRENT query line ===
 *node access: 7
 *backtracking: 2


 This R-Tree contains 5 internal node, 628 data nodes and 100000 data


 *Total node accesses of (1+...+30) query line: 124
 *Total backtracking: 17
----------------------------------------------------
```

Figure 9. Results of the point query

guarantee compression performance. On the contrary, since the $QMBR^i$ scheme minimizes the overlap of MBRs and compresses the MBRs into one byte, it results in far fewer node accesses while maintaining the same node-usability as the QMBR scheme.

Figure 11 shows that node backtracking occurs most often in the QMBR scheme. In other words, the search of the data requires moving to upper nodes because of the overlap of MBRs. The RMBR and HMBR schemes entail more backtracking than the MBR scheme. The reason is that the offsets are big from the starting-point of the search region and the overlapping increases. In Figure 11, backtracking increases in all schemes at a node size of 1 024. We also observe an increase in the RMBR and HMBR schemes to 2 048. This means that even though the node size increases, the uncompressed MBR is stored in other nodes. However, when the node size increases to 4 096 and the uncompressed key is stored in the same node, backtracking decreases.

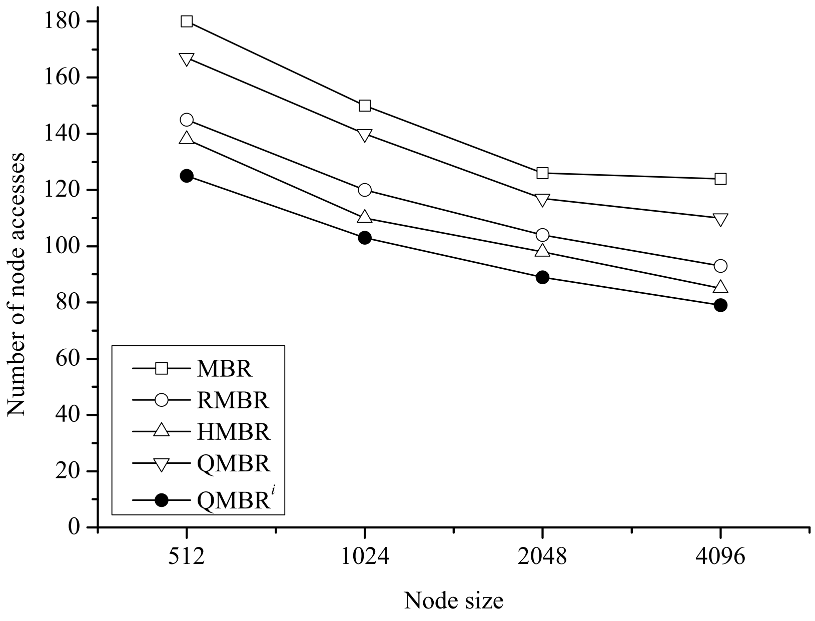


Figure 10. The number of node accesses with the synthetic dataset

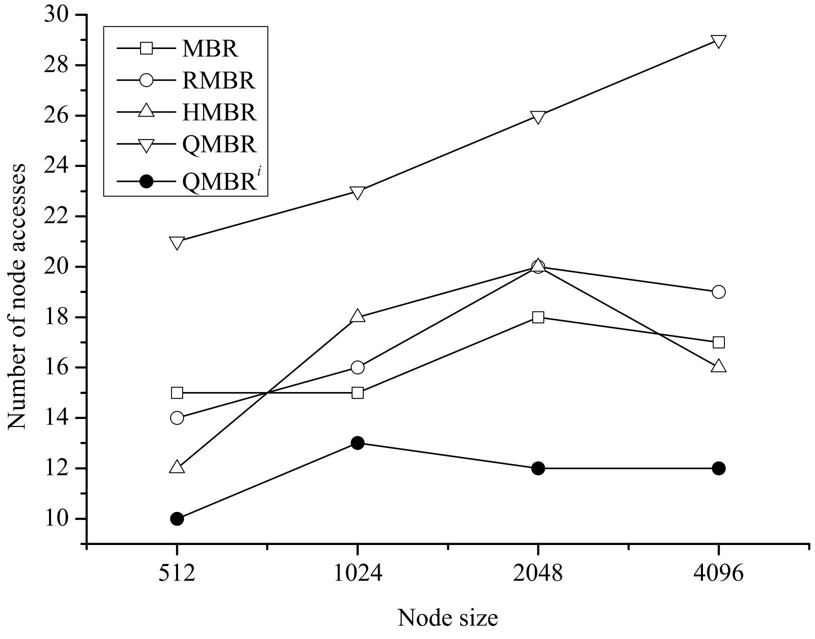


Figure 11. The number of backtracks with the synthetic dataset

In Figure 12, for the real dataset, the number of node accesses decreases as the node size increases in all schemes. The QMBR scheme entails the most node accesses and the QMBR^i scheme requires the least. As shown in Figure 13, the QMBR^i scheme also has the least backtracking. Since many entries are stored in one node in the QMBR^i scheme, the performance is better than that of other schemes. When the node size is 4 096, the backtracking of the RMBR scheme is zero, the same as for the QMBR^i scheme.

The simulation results with the synthetic and real datasets differ slightly according to differences in the data distribution. All of the graphs show that the QMBR^i scheme requires fewer node accesses than the QMBR scheme, despite differences in data distribution. Node backtracking, which is the major cause of performance deterioration in two-dimensional spatial indexes, is lowest in the QMBR^i scheme. This is due to a significant decrease in MBR overlap in comparison with the QMBR scheme, as each axis is compressed into one byte during spatial-data compression.

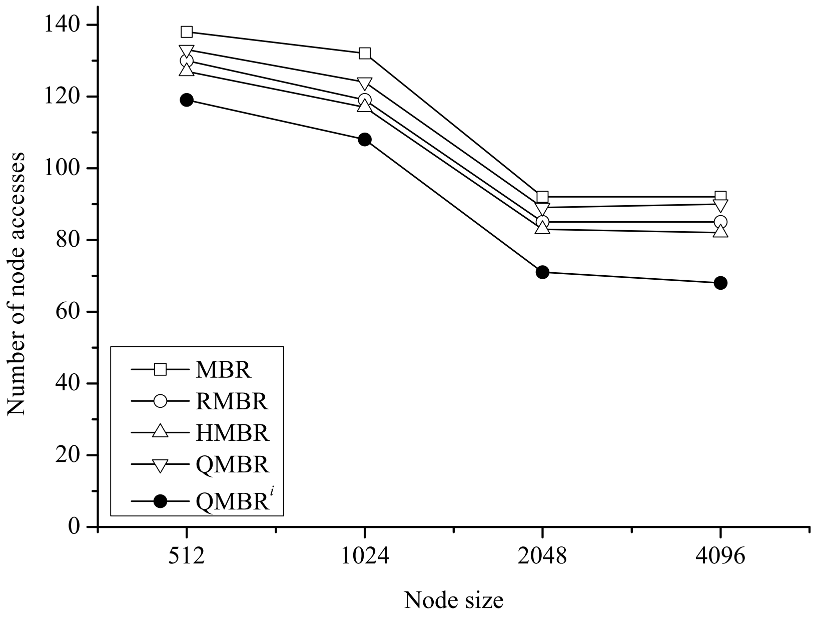


Figure 12. The number of node accesses with the real dataset

6 CONCLUSIONS

We have proposed the QMBR^i scheme, a new spatial-data compression scheme that overcomes the problems related to the size of relative coordinates and MBR overlap that are caused by quantization. Even though the QMBR^i scheme quantizes the search region in the same manner as the QMBR scheme, it minimizes the overlaps

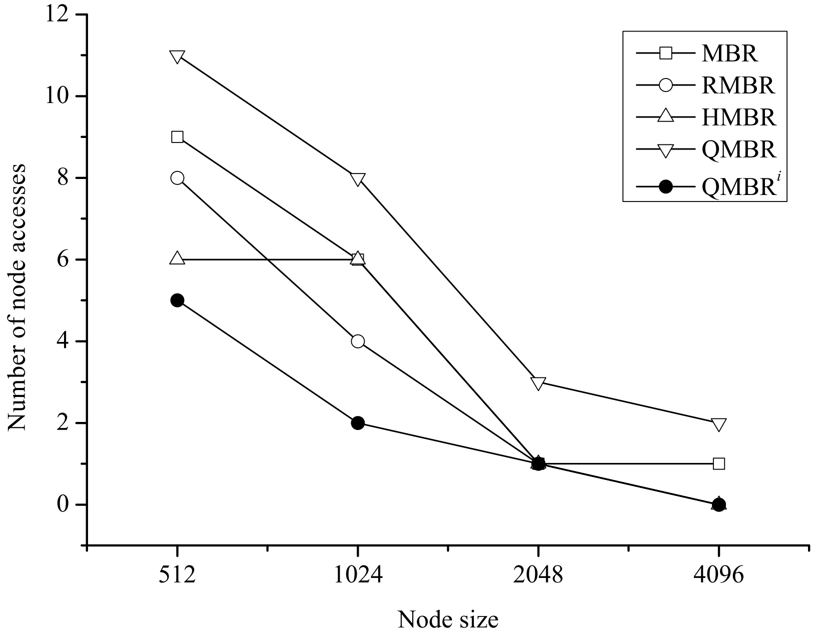


Figure 13. The number of backtracks with the real dataset

between MBRs that occur in the QMBR scheme. The QMBR^i scheme creates new coordinates that are compressed into one byte on the basis of the quantization value of each axis. Since the offsets are decreased into one byte, the shortcomings of RMBR and HMBR are resolved. We refer to this as an inverse quantization scheme. Since the QMBR^i scheme maximizes the compression ratio and the node-usability, it has better performance in terms of the number of node accesses and node backtracking. The simulation results show that the QMBR^i scheme has a smaller index and faster search performance than the existing schemes. We will continue our research to provide faster location-based services by applying the QMBR^i scheme to wireless data broadcasting services that rely on spatial data.

Acknowledgement

This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2009-351-D00075) and IT R & D program of MKE/KEIT (10041910, Development of global cloud delivery platform that can reduce video traffic up to 50 %).

REFERENCES

[1] Guttman, A.: *R*-trees: A Dynamic Index Structure for Spatial Searching. In Proceedings of the 1984 ACM SIGMOD international conference on Management of data SIGMOD, Vol. 14, 1984, pp. 47–57.

[2] Qian, Y.—Zhang, K.—Huynh, D. T.: PatZip: Pattern-Preserved Spatial Data Compression. In Proceedings of the 9th Pacic-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Lecture Notes in Artificial Intelligence (LNAI) 3518, 2005, pp. 726–736.

[3] Berchtold, S.—Bäohm, C.—Jagadish, H. V.—Kriegel, H. P.—Sander, J.: Independent Quantization: An Index Compression Technique for High-Dimensional Data Spaces. In Proceedings of the 16th International Conference on Data Engineering (ICDE), 2000, pp. 577–588.

[4] Kim, J.—Im, S. J.—Kang, S. W.—Hwang, C. S.—Lee, S. K.: SQR-Tree: A Spatial Index Using Semi-Quantized mbr Compression Scheme in *R*-Tree. Journal of Information Science and Engineering (JISE), Vol. 23, 2007, No. 5, pp. 1541–1563.

[5] Kim, J. D.—Moon, S. H.—Choi, J. O.: A Spatial Index Using MBR Compression and Hashing Technique for Mobile map Service. In Proceedings of International Conference on Database Systems for Advanced Applications (DASFFA), LNCS3453, 2005, pp. 625–636.

[6] Kim, K. H.—Cha, S. K.—Kwon, K. J.: Optimizing Multidimensional Index Trees for Main Memory Access. In Proceedings of the 2001 ACM SIGMOD international conference on Management of data SIGMOD, Vol. 30, 2001, pp. 139–150.

[7] Berg, M.—Kreveld, M.—Overmars, M.: Computational Geometry – Algorithm and Applications. 2nd Ed. Springer, 2000, pp. 267–290.

[8] Spatial Data Generator, DaVisual Code1.0. Available on: `http://isl.cs.unipi.gr`.

[9] Sequoia Dataset. Available on: `http://www.rtreeportal.org/spatial.htm`.

[10] MATLAB. Available on: `http://www.mathworks.com`.

[11] Lin, H. Y.: High Index Compression without the Dependencies of Data Orders and Data Skewness for Spatial Databases. Journal of Information Science and Engineering, Vol. 27, 2011, No. 2, pp. 561–576.

Jongwan Kim received the Ph. D. degree in Computer Science and Engineering from Korea University, South Korea, in 2007, B. Sc. degree in Business Administration, and M. Sc. in Computer Science and Engineering from Sahmyook University, Soongsil University, respectively. He has led a project for spatial data compression and involved in RFID project, which are supported by the National Research Foundation of Korea Grant funded by Korean Government. He worked at Konkuk University and Sahmyook University as a research professor and he researched at University New South Wales in Australia as a visiting fellow from 2011 to 2013. His research interests include Mobile & Streaming Data Management, Location-based Services, Sensor/RFID data management and Skyline query.

Dukshin Oh received the Ph. D. degree in Management Information Systems from Sangm-yung University in Korea. He is a professor of Management Information Systems at Sahmyook University in Korea. He has published many papers in several journals such as International Journal of Computer Science and Network Security, The KIPS Transactions. His research interests include Management/Computer Information Systems, System Analysis and Design, e-Business and e-Learning systems.

Keecheon Kim received the B. S. degree from Seoul National University, Korea in 1988, the Ph. D. degree in Computer Science from Northwestern University, Illinois, USA. He has been a senior researcher in Korea Telecom from 1992 to 1996. He joined the current SKT in 1996 as a deputy director for the research center. He has been served as a professor of computer science department since 1998. He also served as a director of the NEXTEL (currently SPRINT Nextel) Communications in US from 2004 to early 2006 in the area of Core network Development. He was a member of US NSTAC Presidential advisory committee. His research interests include Wireless Data Communication, Mobile Computing, Ad-hoc Network, Mobile Computing Security.