Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 4315850, 13 pages
https://doi.org/10.1155/2020/4315850

Research Article

An Optimization Technique of the 3D Indoor Map Data Based on an Improved Octree Structure

Xiaomin Yu,[1,2] **Huiqiang Wang**,[1] **Hongwu Lv**,[1] **and Junqiang Fu**[1]

[1]*College of Computer Science and Technology, Harbin Engineering University, Harbin, Heilongjiang, China*
[2]*College of Computer and Control Engineering, Qiqihar University, Qiqihar, Heilongjiang, China*

Correspondence should be addressed to Huiqiang Wang; wanghuiqiang@hrbeu.edu.cn

Received 15 April 2020; Revised 2 June 2020; Accepted 10 June 2020; Published 10 July 2020

Guest Editor: Chi-Hua Chen

The construction and retrieval of indoor maps are important for indoor positioning and navigation. It is necessary to ensure a good user experience while meeting real-time requirements. Unlike outdoor maps, indoor space is limited, and the relationship between indoor objects is complex which would result in an uneven indoor data distribution and close relationship between the data. A data storage model based on the octree scene segmentation structure was proposed in this paper initially. The traditional octree structure data storage model has been improved so that the data could be backtracked. The proposed method will solve the problem of partition lines within the range of the object data and improve the overall storage efficiency. Moreover, a data retrieval algorithm based on octree storage structure was proposed. The algorithm adopts the idea of "searching for a point, points around the searched point are within the searching range." Combined with the octree neighbor retrieval methods, the closure constraints are added. Experimental results show that using the improved octree storage structure, the retrieval cost is 1/8 of R-tree. However, by using the neighbor retrieval, it improved the search efficiency by about 27% on average. After adding the closure constraint, the retrieval efficiency increases by 25% on average.

1. Introduction

Nowadays, people have higher requirements for using the electronic map due to the popularization of electronic map application and the rapid development of related technologies. The traditional two-dimensional map cannot meet the latest requirements in terms of visual effects. Surveys showed that people are spending more time in indoor environment nowadays, including shopping malls, exhibition halls, libraries, cruise ships, and many other large-scale indoor environments [1]. The existence of three-dimensional indoor map is of practical significance [2]. Data storage efficiency, retrieval consumption, time delay, and other parameters will directly affect the application level of the maps [3]. Consequently, the efficient management of the indoor map data has become a major challenge.

At present, all walks of life, including the fields of map, are faced with the problem that huge data is difficult to store and complex to query [4]. Since the data sets of a map scene are very large, they are not able to be stored directly by the indexes. Therefore, a scene segmentation method is needed as a secondary index to reduce the data storage unit and achieve the lightweight storage. Generally speaking, due to the different environment, maps can be divided into outdoor maps and indoor maps. The regional segmentation technology is preferable as the outdoor scene is large and uneven, and the map data is stored as sheet. Compared with the outdoor map, the indoor map includes numerous objects, and the volume of the objects is small. The situation of uneven distribution often exists. If the sheet framing is used, a large number of objects may be distributed in a sheet. This extreme situation will reduce the regional segmental effect. In order to solve this problem, it is necessary to continue the segmentation, which will increase the cost of scene segmentation. The sheet framing cannot meet the vertical requirement between different floors and cannot preserve the three-dimensional nature of the data.

The technology of the storage of outdoor map data has developed rapidly, while the indoor map data storage is still in the initial stage [5] of development. For the indoor data storage, the scene can be divided by tree structure, and the octree is one of the high-efficiency scene segmentation methods, which can realize the rapid segmentation of 3D map scene and retain the characteristics of map data [6]. However, at present, this method is mainly applied in the field of game scene and 3D image processing, and there are few applications for indoor map. Moreover, indoor map data is updated more frequently, and there are room partitions between the data. Therefore, the traditional scene segmentation methods are not applicable for the indoor map data [7]. One of the key problems is to design a set of storage structure suitable for indoor map features based on fast scene segmentation method.

The efficiency of the octree structure in the scene segmentation is very high. However, its performance in data retrieval is not ideal. After scene segmentation using octree structure, map data will be stored in corresponding nodes. We need to traverse every required data tree, so the efficiency of querying and updating indoor map data is not high. This leads to another key problem, which is improving the efficiency of data retrieval and ensuring high storage efficiency.

In this paper, a set of indoor map data storage model based on the scene segmentation is designed, which can fulfill the scene segmentation, code design, data expression, and storage structure design. And a fast data retrieval method based on the data storing models is also proposed. These results provide an efficient data storage and retrieval scheme.

2. Related Works

At present, map data processing software (such as ArcGIS) or domestic emerging software (including SuperMap) mostly adopt the scene segmentation for data unit division and form a map data index structure. The ArcGIS SDE of Esri supports fixed grid scene segmentation structure, while MapGIS and SuperMap support quad-tree scene segmentation structure. Furthermore, ArcView and MapInfo support R-tree scene segmentation structure, while Oracle Spatial supports both R-tree and quad-tree scene segmentation structure [8].

Fixed grid is a kind of scene segmentation structure which is widely used in two-dimensional space environment. Its specific principle is to use a preset size mesh to segment the scene and form a rectangular pixel of equal size. Each rectangular cell is uniquely annotated and recorded according to the row number and the column number. The contents stored in the rectangle cell area are related to the information and properties of the data. The data in the whole scene find their respective storage locations through uniquely marked rectangular area [9].

And when it is necessary to retrieve the data from the scene, the location of the cell can be located by querying the row number and the column number. Since there may be multiple data in each cell, it is also necessary to compare the data attribute values to determine the retrieved data. According to the research conducted by Qiao [10], data retrieval with fixed grid method can be divided into accurate retrieval and range retrieval. Accurate retrieval can complete the search of row number and column number to uniquely determine the cell location, but it still needs range retrieval to continue to retrieve all units that intersect the range retrieved in the first step. One of the studies conducted by Yang [11] pointed out that due to the variety of map data, when the data attributes were particularly complex, it would lead to the redundancy of the index records and reduce the efficiency of data retrieval. This method is similar to the rectangle framing and longitude latitude framing in the map, and it is more suitable for two-dimensional space.

Originally as an index method, R-tree structure was proposed by Guttman in 1984. This structure is applicable to both two-dimensional environment and three-dimensional environment, and it is an extension of B-tree [12]. Many researches have been carried out on the deformation of R-tree, and specific needs have been met [13]. These studies often expand R-tree node and store some properties similar to topological relationship. However, this method will double the stored information which increases the burden to the storage and cause the problem of data redundancy.

The study carried out by Wang et al. [14] proposed a method combining R-tree and the octree to manage large-scale three-dimensional scene. Its advantage is to implement dynamic, efficient, and real-time management of the change data during the interaction. It makes the rendering picture smoother, resulting in a significant reduction in rendering time. However, due to fusing two types of complex structures, it will consume more time to construct the data structure. The study carried out by Namdari et al. [15] proposed an efficient parallelization task execution method based on R-tree. By designing the storage structure, the problem of limited parallelization effect of R-tree on GPU was solved, and a fast search method of R-tree was proposed. R-tree is one of the commonly used storage methods at present, but the objects in the indoor scene are placed with a large density and uneven distribution, so this method is not much applicable [16].

As a scene segmentation method of tree structure, the quad-tree is mainly applied to two-dimensional scene. It uses the fixed grid structure to divide each space into four rectangular areas. It can achieve more efficient space segmentation compared with the grid structure. Compared with R tree, quad-tree structure is relatively simple, so its construction time is shorter. Due to the regularity of each recursion, each quad-tree node can be encoded. The data retrieval can be completed by encoding. This structure is more suitable for two-dimensional space scenes similar to the fixed grid structure [17].

The octree structure was first proposed by Meagher in 1982 [18]. It is an extension of the quad-tree in three-dimensional environment [19], and it has the characteristic of flexibility and extensibility [20]. It is a popular method in the field of 3D image processing and spatial data index [21] and has a better effect than R-tree [22]. The octree has been applied in the storage of 3D objects data such as iron and

steel industry, etc. [23]. Riegler et al. [24] and Tatarchenko et al. [25] proposed a fast scene reconstruction method by taking advantage of the regularity of the octree segmentation. Deng et al. [26] proposed a method to add an update memo in the octree, which improves the update efficiency and significantly reduces the update cost of 3D objects.

These methods can initially realize the basic storage of 3D spatial data, but due to the large density and uneven distribution of indoor map data, these storage processes have many disadvantages in some aspects.

When the map scene is segmented by the octree, the objects' data is stored in the nodes. In this way, when retrieving objects, it needs to go through the tree. So, it is very tedious for the data with large retrieval requirements. Many scholars have studied the retrieval method, and the current aspect is the retrieval of neighbor nodes. The neighbor query is divided into four classes [27]: (i) all objects are also the query points. This is known as an all-kNN query (akNN); (ii) only selected objects are the query points. This is known as a Group Nearest Neighbor (GNN); (iii) there is a variant of GNN, which is called Group Nearest Group (GNG); and (iv) only object locations within a given rectangle are the query points. This is known as a Range-kNN query. Optimization techniques used include machine learning [28–30], deep learning [31, 32], and optimization technologies for transportation [33–35]. The study carried out by Pan et al. [31] proposed a fine-grained classification model named RMA (ResNet-Multiscale-Attention) based on deep learning to analyze the subtle and local differences among navigation mark types for the recognition of navigation marks. The study carried out by Pan et al. [32] proposed a CNN-GRU model combined with the network structures of gated recurrent unit (GRU) and convolutional neural network (CNN). The GRU part learned the changing trend of water level, and the CNN part learned the spatial correlation among water level data observed from adjacent water stations.

The study carried out by Cho et al. [36] proposed a neighbor retrieval method, which took advantage of the regularity of the octree segmentation and could find out the 26 neighboring nodes by calculating the offset in three directions of x, y, and z. However, the neighbors search was done after the tree is built again. This approach is simple and does not add any time complexity. Furthermore, Namdari et al. improved on this basis [15] and proposed an improved neighbor retrieval method, which stored the neighbors of all leaf nodes in the process of building the tree. It has the advantage of conventional octree partitioning. Storing neighbors when building a tree will not cause damage to the complexity of building tree. They proved that the sum of time for retrieval, storing neighbors, and building tree is not much different from the time it takes to create the tree alone.

Although the neighbor node search method is feasible, it does not need all 26 neighbor nodes for the indoor scene of closed room. There is still some unnecessary consumption, and the efficiency needs to be further improved. In this study, we proposed an octree-based data storage model with backtracking and an octree neighbor retrieval algorithm with closed constraints.

3. Materials and Methods

3.1. Map Scene Segmentation. The octree storage model usually consists of three parts: scene segmentation, node coding, and data representation. The basic principle to fulfill the function is using the octree to segment 3D scene regularly, using octree code to encode the route of each node [37]. The cube boxes are further used to express the objects approximately, and the codes are stored in the corresponding nodes according to the size and location of the boxes [38–40].

3.1.1. Octree Scene Segmentation Technology. The octree is a regular segmentation data structure. It approximately takes the overall scene of a three-dimensional indoor map as a cube and takes it as the root node of the tree. Moreover, it divides the scene into 8 or 0 pieces at a time during scene segmentation. Its schematic diagram is shown in Figure 1, and its definition is as follows:

$$\text{octree} = \begin{cases} v_i = \frac{1}{8} v_{i-1},\ d_i = \frac{1}{2} d_{i-1}, h_i = h_{i-1} + 1, & n > 2, \\ \text{end of separation}, & n \le 2 \text{ or } d_i = d_{\min}, \end{cases} \tag{1}$$

where v_i, d_i, and h_i are the volume, side length, and depth of the cube obtained by the ith division, respectively, n is the number of objects contained in the cube, and $d_{\min}$ is the minimum side length that the cube can be separated into.

There are three types of nodes in an octree. In order to distinguish them, three colors are used in this paper. The white nodes indicate that no data has been stored, whereas gray nodes indicate that more than two objects' data have been stored and black nodes indicate that one- or two-object data have been stored. White and black nodes can be regarded as leaf nodes, and gray nodes need to be further segmented. This segmentation is a recursive process, whose termination condition is that all nodes are leaf nodes or segmentation reaches the specified minimum segmentation area. Figure 1 shows a twice-segmented tree structure, in which (a) is a schematic diagram of three-dimensional cube, whereas white circles represent the objects, and (b) is a schematic diagram of two-dimensional tree structure.

After a three-dimensional map is recursively segmented by octree algorithm, many subregions will be formed, which are distributed in all the corners of the three-dimensional space.

3.1.2. Octree Node Addressing Encoding. In order to make the data management organized in the octree data storage models, it is necessary to make clear position of each data corresponding to the tree. In this paper, the position of each subregion is marked by encoding. The node addressing code uses digits to mark the 8 child-nodes which were extracted from each node, so that the 3D data can be represented by one-dimensional digits. Among which the number of encoded bits is the same as the number of layers, and each encodes unique marks of the node region while recording

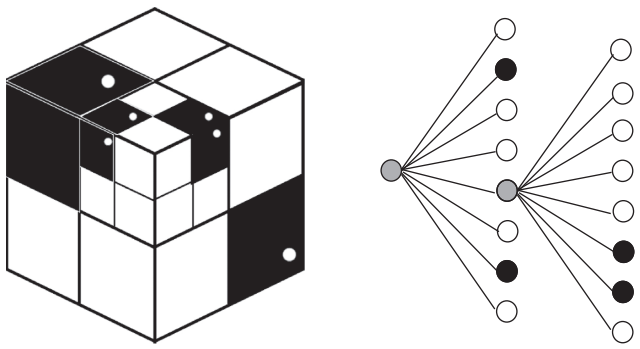


Figure 1: The octree scene segmentation schematic diagram.

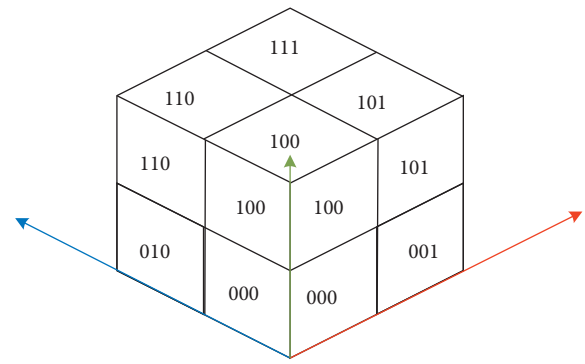


Figure 2: The octree nodes addressing encoding.

the path of this node to the root node. The method of "push back" is used in encoding. The first segmented code is taken as the first digit, and the result of each recursive segmenting is added to the last several digits in turn. The segmented 8 subregions can be distinguished by the deviation degree from the three direction axes, and this method can mark the spatial information in the 8 directions with only 3 binary bits. The following formula can be used to calculate the node addressing encoding, where p is the current number of layers and n is the number of layers to be segmented:

$$\text{coding} = \left(x_p y_p z_p\right), \left(x_{p+1} y_{p+1} z_{p+1}\right), \ldots, \left(x_{p+n} y_{p+n} z_{p+n}\right). \tag{2}$$

The first bit of the three-bit binary code indicates the deviation degree from the x-axis, 0 indicates the subregion has no deviation in the x-axis direction, and 1 indicates the subregion has deviation in the x-axis direction. Similarly, the second bit and the third bit represent the deviation degree of the subregion in the y-axis and z-axis direction, respectively, which is exhibited in Figure 2.

The maximum number of layers for node segmentation is determined by the system word length. m is used to indicate the maximum number of layers that the current system can be divided into, and *Bits* represents the number of bits in the system word length:

$$m = \frac{\text{bits}}{3}. \tag{3}$$

A 64-bit system can be divided up into 21 layers. Assuming that the required minimum partition area is $1\,\text{m}^2$, the maximum coverage area is $4^{21}\,\text{m}^2$.

According to the addressing encoding method, the storage structure of the octree can be encoded. In order to easily record, the code can be converted to decimal representation as shown in Figure 3, which is a twice-segmented 3D scene. The specific steps of segmentation and encoding are as follows:

(1) The 3D map scene is represented in the form of cube approximately.

(2) By treating this cube as the root node of the tree, according to its deviation from the three axes, it is marked with three binary numbers, and then the binary numbers are converted into corresponding decimal numbers. The first segmentation and coding are completed.

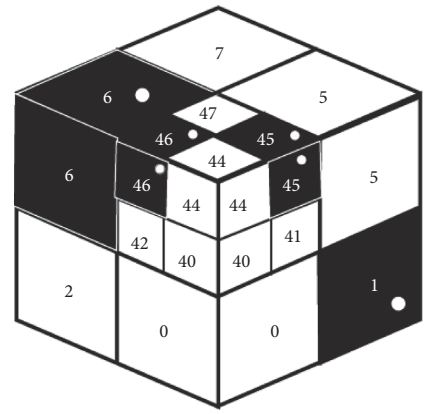


Figure 3: The decimal coded representation of the octree nodes.

(3) After the first segmentation, the regions numbered 0, 2, 3, 5, and 7 do not contain data, so they are white nodes and can be used as leaf nodes without further segmentation. The regions numbered 1 and 6 contain one data, respectively, so they are black nodes and can be regarded as leaf nodes without further segmentation. The region numbered 4 contains 4 data, so it is a gray node. It cannot be regarded as a leaf node, and the segmentation should continue; repeat step (2) to divide and encode.

(4) After the second segmentation, the region originally numbered 4 is divided into 8 subregions, among which the nodes numbered 40, 41, 42, 43, 44, and 47 do not contain data, so they are white. The nodes numbered 45 contain two data, and the node numbered 46 contains two data, so the two nodes are black. At this time, all nodes can be regarded as leaf nodes, so there is no need to segment, and the recursive operation of scene segmentation is finished.

3.1.3. Data Expression. In three-dimensional indoor maps, most of the shapes of objects are very irregular, and the objects with cube shape occupy a much larger proportion than that with spherical shape. Therefore, after statistical

analysis, Axially Aligned Bounding Box (AABB) method was considered to be more appropriate. The enclosing shape is a cuboid, and each side of the cuboid needs to be consistent with the corresponding coordinate axis direction. The benefit of normalizing the cube is that the AABB can be easily calculated based on the maximum and minimum coordinates of each vertex. The size and coordinates are as follows:

$$pos = \left\{(x, y, z) \middle| x_{\min} \le x \le x_{\max}, y_{\min} \le y \le y_{\max}, z_{\min} \le z \le z_{\max}. \right. \tag{4}$$

The minimum coordinate of point A in the lower-left corner and the maximum coordinate of point B in the upper-right corner can be selected and wrapped into a range. Two points A and B can be used to represent the bounding box as shown in Figure 4.

By combining the above octree scene segmentation, node addressing encoding, and data approximate expression AABB, a storage model based on traditional octree can be formed. However, due to its approximate expression, some error rates still exist and are acceptable.

3.2. Design of Map Data Storage Method. In the process of saving the approximate representation data into the octree, few practical problems appeared such as map data conflict, partition line and object conflict, etc. Therefore, an improved octree structure with backtracking characteristics is proposed in this paper.

3.2.1. Resolution to Map Data Conflict. In the three-dimensional indoor map scene, the distribution of objects is irregular. Many objects are too close to each other and even overlap. Therefore, when using bounding boxes to represent the objects approximately, the bounding boxes may intersect or overlap. As AABB is cube and cube box sides, respectively, on the axis, when a minimum coordinate and a maximum coordinate are used to represent the cube, we should check whether the coordinates of the two objects overlap or not. Suppose the coordinates of A are $\{(x_{A_{\min}}, y_{A_{\min}}, z_{A_{\min}})\}$, $\{(x_{A_{\max}}, y_{A_{\max}}, z_{A_{\max}})\}$, and the coordinates of B are $\{(x_{B_{\min}}, y_{B_{\min}}, z_{B_{\min}})\}$, $\{(x_{B_{\max}}, y_{B_{\max}}, z_{B_{\max}})\}$; then, the judgment conditions are as follows:

(1) If $x_{A_{\max}} < x_{B_{\min}}$ or $x_{B_{\max}} < x_{A_{\min}}$, the bounding box cubes of two objects do not conflict

(2) If $y_{A_{\max}} < y_{B_{\min}}$ or $y_{B_{\max}} < y_{A_{\min}}$, the bounding box cubes of two objects do not conflict

(3) If $z_{A_{\max}} < z_{B_{\min}}$ or $z_{B_{\max}} < z_{A_{\min}}$, the bounding box cubes of two objects do not conflict

If none of the above three conditions is satisfied, the AABB surrounded cubes of two objects are to be in conflict. In the three-dimensional map of indoor scenarios, due to the relatively small volume of a single object, for the conflicted data, we should use a bigger bounding box for approximate expression, that is, obtaining the maximum and minimum coordinates of the four coordinates used by two objects to represent the coordinates of the new bounding box. Because this kind of conflict is not a lot in the actual indoor environment and the new bounding box volume is not large, the cost is acceptable. Another advantage of taking the method is to help solve the difficulty caused by the unexpected situation of the dividing line at the intersection of two objects of the AABB bounding boxes.

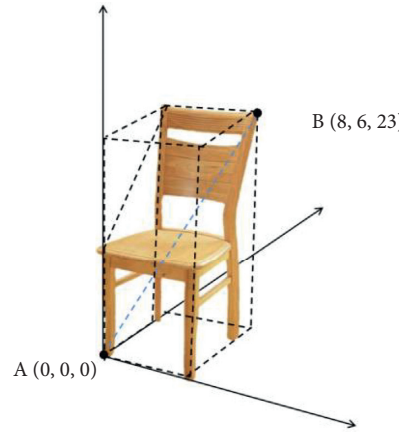


Figure 4: The sample of AABB object.

3.2.2. Resolution to Conflict between Dividing Line and Object. The linear way can replace the storage of pointers by addressing codes, which only need to record the addressing codes of leaf nodes, which can greatly reduce the storage pressure. In addition, the linear octree can be transformed into the corresponding linear list for representation. The elements in the linear list correspond to the nodes, which can increase the convenience of calculation.

Linear mode is one of the most widely used storage methods. All data of this method are only stored in the leaf nodes, and the middle nodes do not store any data. However, in the three-dimensional indoor map scene, the location of indoor data is very close. It often exists in the situation that the dividing line is on the indoor object. If the object data is segmented at this time, it will terminate the object data and need to be reconstructed again. This is a complex process and requires a large amount of calculation. Therefore, this method is not applicable to the data storage of 3D indoor map.

According to the characteristics of the close distribution of indoor map data, a new data storage model is proposed in this paper. It improves the linear octree and makes it to have backtracking ability. Originally, the map data is still stored at the leaf nodes, when the dividing line occurs in the object, the object data is not divided and instead it gets stored in the nearest parent node that can fully contain the object data. Since the object data storage has been completed at this time, the scene segmentation can be continued.

3.2.3. Algorithm Implementation

(1) Definition of relevant data structure

(I) Leaf Tree Node structure

```
Struct LeafTreeNode
{
Coordinate [4]//Coordinates of the cube
NodeCoding//Location code
NodeDepth//The level of node
ObjectNum//The number of objects contained in nodes
ObjectID[objectNum]//Object code
}
(II) Non-Leaf Tree Node structure
Struct Non-LeafTreeNode
{
Coordinate [4]//Coordinates of the cube
NodeCoding//Location code
NodeDepth//The level of node
Children [8]//the pointers point to the node's children
ObjectNum//The number of objects contained in nodes
ObjectID[objectNum]//Object code
}
(III) Object structure
Struct Object
{
ObjectID//Object code
ObjectSize//Object size (AABB)
RoomID//Room code that the object is in
}
(IV) Room structure
Struct Room
{
ObjectID//Object code
ObjectSize//Object size (AABB)
RoomID//Room code the object in
}
```

The three structures defined above have different functions, but they are also related to each other. The algorithm of creating an octree is as follows Algorithm 1.

3.3. Closed-Orientation 3D Indoor Map Data Retrieval Method. Based on the data storage model which is proposed in this paper, a data retrieval method with closed constraints was proposed according to the characteristics of more indoor partition to improve the efficiency of data retrieval.

3.3.1. Analysis of Neighbor Retrieval Algorithm Based on the Octree. In octree structure, there are at most 26 neighbor nodes of a node. According to the type of critical location between the target node and neighbor node, three cases exist: surface-neighbor, edge-neighbor, and point-neighbor, as shown in Figure 5. There are 6 surface-neighbors, 12 edge-neighbors, and 8 point-neighbors for one node.

In the octree, the calculation of neighbor nodes can be realized by finding the nearest common ancestor of target node and neighbor nodes. According to the direction and area information of the nodes, this method completes the calculation of address coding of neighbor nodes. This calculation is based on the modifying of the target node addressing code. The process is from the rightmost coding in turn to the left, updating the value of the target node at that position with the numerical results obtained by each bit of computation.

In the octree environment, if we want to find the 26 direction's information, we can only record the surface-neighbor data in orthogonal direction {D, U, R, L, B, F}. The 12 edge-neighbors and 8 point-neighbors' data can achieve through orthogonal direction data twice or thrice operation. This can reduce the amount of data table.

When the target node and the neighbor node in the given direction are brothers, the calculation method executes the "stop" command and directly modifies the rightmost digit value. Otherwise, the nearest common ancestor of the two nodes should be located, and then relevant calculation can be performed. When the layer of neighbor nodes is deeper than the target nodes, which is due to the neighbor node calculation method can only calculate the same size nodes with the target nodes, and the nodes calculated are not leaf nodes. That is, the nodes adjacent to the target nodes can also continue to refine. Although it is difficult to solve this problem, the research background of this paper is 3D indoor maps. Most of the map data volume is small, and therefore we can regard the calculated neighbor nodes as the target node's neighbor nodes, and it does not need any further detailed analysis. The effects of these errors are negligible.

3.3.2. Neighborhood Retrieval Optimization Algorithm in Indoor Map Scene. In the traditional octree neighbor node retrieval methods, the results of each retrieval are the target node's 26 neighbor nodes in all directions. As we knew from the analysis mentioned earlier, there is no need of all 26 neighbor nodes in the 3D indoor scenarios. We can take the indoor room partition into consideration. The neighbor nodes in the same room with the target nodes are needed with the data. The optimization of the traditional octree neighbor retrieval method can realize the filtrating of neighbor nodes, reduce the number of neighbor nodes, reduce the workload for the later neighbor data analysis, thus reduce the cost of retrieval, and improve the stability of the algorithm.

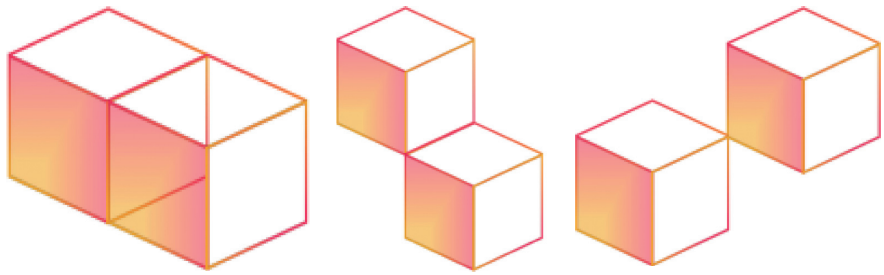
In the actual retrieval process of indoor map, according to the area where the target data is located, it can be divided into the following situations:

(1) The retrieval of data on the room ground: the retrieval in this environment can improve the efficiency by using the closure neighbor retrieval method.

(2) Corridor data retrieval: unlike the data on the room ground, the objects in corridor are limited, the environment is relatively open, and the enclosed space

```
Input: TreeNode//Map scene is as root node
Output: TreeNode//The root node of the segmented data storage model
Begin:
  For i = 0: MAX//Scene segmentation
    newNodeDepth = nodeDepth + 1
    newNodeParents = node
    newNodeAABB = nodeAABB/8
    If nodeAABB.value > 2 and node.AABB > nodeMin then
      If Line on nodeAABB then
        Search for the smallest ancestor
        Data is stored in the ancestor node
        Delete original data
      End If
      Divide node
      Node coding
    End If
  End For
  Store leaf node data
  Return TreeNode
End
```

Algorithm 1: Octree Map Data Storage.



(a) (b) (c)

Figure 5: The neighbor nodes of a node. (a) The surface-neighbor. (b) The edge-neighbor. (c) The point-neighbor.

is few. Therefore, using neighbor search algorithm does not improve the efficiency, and it will lead to unnecessary retrieval costs. So, in this case, we can retrieve directly.

(3) The retrieval of data inside the room of nonground: the data in the area mainly refers to the space among walls and ceiling; it is similar to the data in the corridor. The purpose of retrieving it is not to move it, such as lamps on the ceiling. The purpose of retrieval is to repair, not to change the position. So even though it is closure, it does not meet the usage scenarios of closure neighbor search algorithm; this case also can retrieve directly.

These three cases can contain most of the data, and direct retrieval is sufficient for extreme cases.

Because different retrieval algorithms are used in different scenarios, the logical relationship of the object position parameters is as follows: x represents object data, $A(x)$ represents data on the floor of the room, $B(x)$ represents data on the corridor, and $C(x)$ represents data in the room that is not on the floor:

$$A(x) \longrightarrow \exists x(B(x) \wedge C(x)). \tag{5}$$

N represents neighbor search:

$$\exists x(B(x) \wedge C(x)) \wedge A(x) \longrightarrow N. \tag{6}$$

Object data on the floor of the room requires neighbor retrieval. $N(x, y)$ is the data that x and y are neighbor. $R(x, y)$ is the data that x and y are in the same room:

$$\exists y(N(x, y) \wedge R(x, y)) \wedge \forall y(N(x, y) \longrightarrow R(x, y)). \tag{7}$$

Although there are cases in which neighbor data is in the same room, not all the neighbor nodes are in the same room.

3.3.3. Algorithm Design. Considering the closure of data, the data retrieval method is the combination of the traditional octree neighbor nodes retrieval method and the closure of indoor rooms. The calculated neighbor data is filtered again by adding the constraint condition of room closure:

(1) Compare the ID of the target data with the node data to find the node where the target data is located and

record the addressing code of the node, which is called the target node.

(2) According to the addressing code of the target node, retrieve the location of the target node in the octree and compare the data in the node with the target data ID to obtain the target data.

(3) If the target data do not conform to the application scope of neighbor retrieval, the retrieval shall be stopped; otherwise, the next step shall be carried out.

(4) Calculate the neighbor nodes of the target node and consider the room constraints, no more than 26. Parse the data in the qualified neighbor node. The two algorithms of octree retrieval and octree neighbor retrieval are shown in Algorithms 2 and 3, respectively.

4. Results and Discussion

4.1. Creation of Experimental Scenarios. In order to ensure the standardization and convenience of 3D indoor map scene creation, SketchUp and ArcGIS were used to create the map scene. ArcMap was used to create the two-dimensional scene base map, in which the surface data structure was used as the wall, and the point data structure was used as the marker point of each specific object in the scene. Finally, in the ArcScene platform, we added a height value for each surface type data structure and made it with a 3D data including height value. We changed the symbol expression mode of each point data to the specific object form which it represented, such as computer table, sofa, cabinet, etc. These specific objects needed to be done by SketchUp specific components. For the purpose of beauty, wall, ground, windows, and doors can be decorated with concrete material. The final scene is shown in Figure 6.

Map scenario consists of two floors; there is a corridor in the middle section of each floor. On both sides, there are some rooms of different sizes. The basic object types include computer chair, table, and bookcase inside the room, a total of 87 indoor objects, and the specific parameters of floor 1 and floor 2 as shown in Tables 1 and 2, not considering the dining regions and stairwells.

4.2. Experiments and Analysis of Neighbor Retrieval Algorithm. For the map data retrieval, the time required is an important index to evaluate the efficiency of a retrieval method. Therefore, in the experiment, based on the proposed storage model, the octree data retrieval method, R-tree data retrieval method, and octree neighbor data retrieval method are tested to analyze the time cost by these three methods in retrieving 3D indoor map data. In order to avoid the impact on the experimental results due to the lack of experimental data, the experiment is conducted under two simulation scenarios. The first experiment scene A is the map scene created in Section 4.1, and the second experiment scene B is shown in Figure 7. The modeling method of this scenario is adopted in Section 4.1.

Both experimental scene A and experimental scene B are three-dimensional indoor maps, and the details of them are shown in Table 3. Now in the second experimental scenarios, three retrieval methods are completed. It is now the analysis of the time used complexity of data retrieval under different conditions.

When 3D indoor map data is directly retrieved without any data structure and retrieval method, the time complexity of retrieval is (n^p), where n is the number of data and p is the number of data to be searched. When only using the proposed data storage model for the experimental data storage, without using any search method, as the octree, it cannot directly link to the other required node address based on a node address. Therefore, it needs to do tree traversal for each retrieval; time complexity of the search is $O(2^{3(L-1)})$, where L is the depth of the tree. R-tree data retrieval method is like octree data retrieval method, but due to the overlapping area occurring in the R-tree, it takes extra time to detect if there is a collision of object data in the overlapping area. In an indoor environment, objects are packed more frequently, so retrieval takes more time. If the concept of "searching for a point, points around the searched point are within the searching range" is used at this time, the neighbor data information is retrieved by the traditional neighbor retrieval method of octree. Because of the regularity of the octree partitioning, it can calculate up to 26 neighbors map data around the target data through the formula. Therefore, the time complexity of neighbor retrieval method is $O(1)$.

This experiment simulated five data retrievals and took scene A as an example for analysis. The data retrieved each time were distributed in multiple rooms. The specific retrieval parameters are shown in Table 4.

Figures 8 and 9 show the comparison of retrieval time among R-tree retrieval, the octree retrieval, and the octree neighbor retrieval under two scenarios, respectively. Figures 10 and 11 show the comparison of the octree retrieval and neighbor retrieval methods in terms of frequency times. From the experimental results, it can be concluded that the octree requires less time consumption in data retrieval operation than R-tree, and its stability is better. It indicates that the octree structure is not only better in construction time than R-tree structure, but also has advantages in data retrieval. For the octree structure, the use of neighbor retrieval method is better than not using it completely, which proves that the idea of "searching for a point, points around the searched point are within the searching range" is feasible. This suggests that neighbor nodes data of the target node is significantly useful. It is verified that the octree neighborhood retrieval can improve the overall efficiency without increasing the time efficiency of single retrieval.

4.3. Experiments and Analysis of Data Retrieval Methods Oriented to Closure Constraints. The experiments in this section will analyze from the aspect of retrieval cost, which refers to the number of neighbor nodes other than the target node each time the neighbor retrieval method is adopted. As the retrieved data needs to be parsed and represented each time, there will be more neighbors and the cost will be higher. The experiments still adopt the simulated retrieval data in the previous section to conduct in the experimental scene A and the experimental scene B.

```
Input: objectID, nodeCoding
Output: target data
Begin: target = root
   While (objectID in CurNode and nodeCoding<>targetNode. nodeCoding) do
      targetNode = CurNode.children[the i-th bit of the nodeCoding]
   While length of node objectNum > 0 do
       /* objectNum is array, does not reach the segmentation threshold */
      collect objectData from targetNode
      if collect objectData = targetObjectData then
         call function OctreeMapNeighborDataRetrieval method
      end while
End
```

Algorithm 2: Octree Map Target Data Retrieval.

```
Input: objectID, nodeCoding, objectPosition
Output: Neighbor data
Begin:
   If objectPosition ≠ roomNoGround and objectPosition ≠ corrido then
   /* Retrieval scenario judgment */
      call NeighborRetrieval method
   End If
End
Function NeighborRetrieval
   For i = 1 : 26
      If targetObject.roomID = neighborObject.roomID then/* Room ID judgment */
         Return True
      End If
   End For
End
```

Algorithm 3: Octree Map Neighbor Data Retrieval.

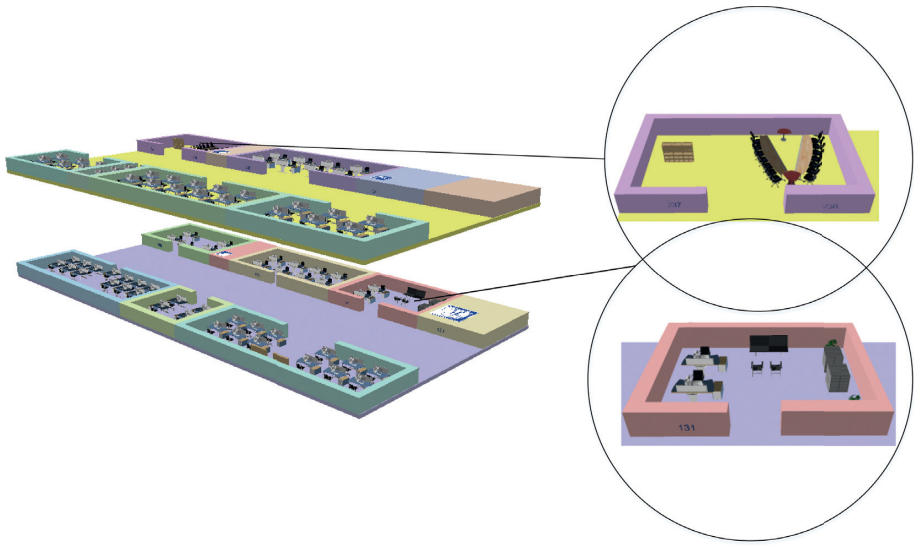


Figure 6: The experimental scenario A.

Table 1: Floor 1 parameters.

Room ID	134	132	131	122	123	124
Object number	5	8	6	13	4	15
Object type	1	1	5	1	1	2
Room length (m)	23	30	28	31	4	55

Table 2: Floor 2 parameters.

Room ID	237	235	225	226	227	228
Object number	2	8	6	3	10	6
Object type	2	1	1	1	1	2
Room length (m)	31	33	30	5	29	26

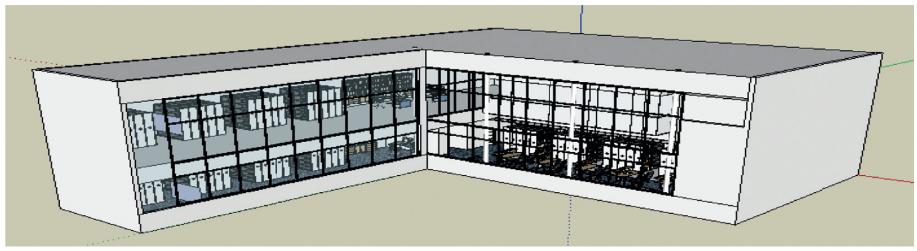


Figure 7: The experimental scene B.

Table 3: Experimental scene parameters.

	Name	Size (kB)	Data format	Object number	Scene length (m)
Scene A	Lab building	16691	sxd	87	90
Scene B	Library	17347	skp	103	60, 46, 90

Table 4: Simulated retrieval operation parameters.

Times	1st	2nd	3rd	4th	5th
Data number	10	15	8	18	20
Room 1 (ID, objects number)	132, 4	122, 8	225, 3	124, 10	227, 5
Room2	131, 4	225, 4	228, 5	123, 4	124, 9
Room3	237, 2	132, 3	—	227, 4	228, 6

From Figures 12 and 13, we can observe that the calculation method of neighbor nodes after adding the closure constraint condition has obviously reduced most cases in the number of neighbor nodes. Compared with the search method without constraint conditions, the search cost is reduced by 25 percent on average, which proves that the room closure condition can reduce the cost of data retrieval and then improve the efficiency of data retrieval on the premise of guaranteeing the efficiency of neighbor retrieval.

5. Conclusions

In this study, we evaluated the problems related to the storage consumption of 3D indoor maps. In order to reduce storage consumption and improve storage efficiency, a 3D data storage model based on the octree structure was proposed. The octree structure is used to segment the indoor map data. The addressing code was used to uniquely mark the generated nodes without retaining the pointers to the parent and child nodes. Compared with pointer storage, this storage method saved 60% of storage space. Considering the complexity caused by irregular shape of indoor objects, AABB method was used to approximately express the object data in the map scene. The advantage of this storage model based on scene segmentation lied in the fact that it could reduce the amount of storage units and improve the storage efficiency. Moreover, the connection between each storage unit was convenient for future management and use. Then, according to the characteristics of close distribution of indoor map data, the situation of octree scene segmentation

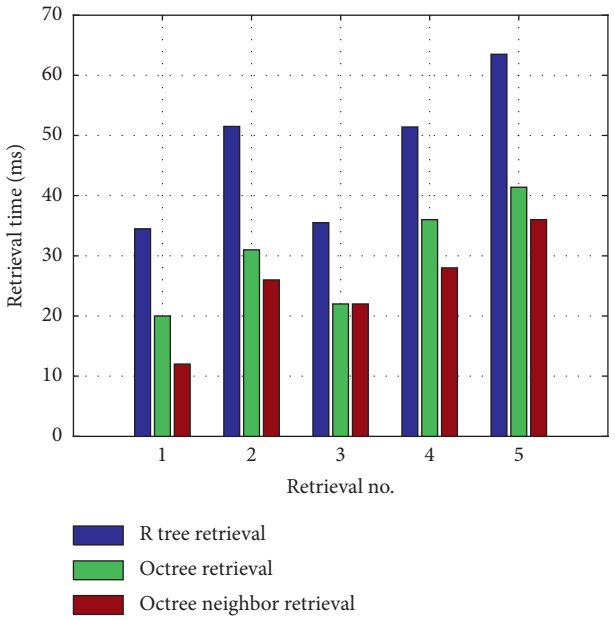


Figure 8: The simulated data retrieval in scenario A.

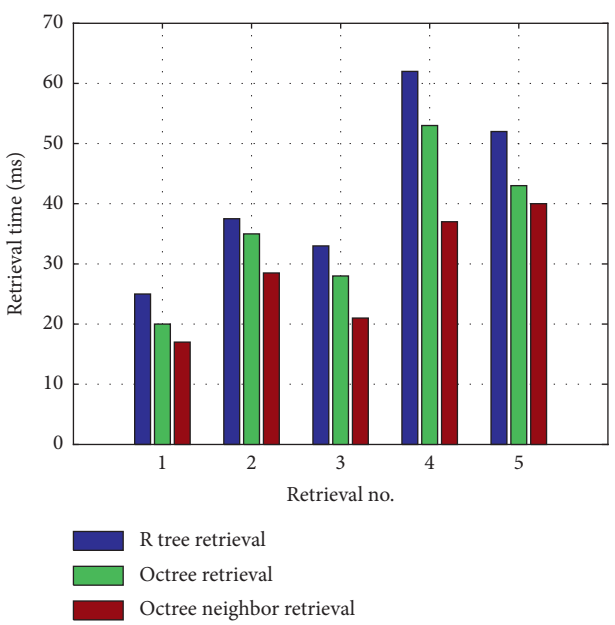


Figure 9: The simulated data retrieval in scenario B.

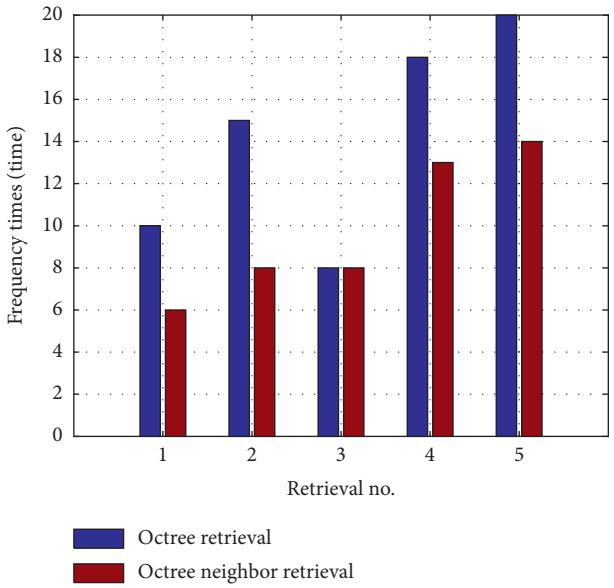


Figure 10: The number of data retrievals in scene A.

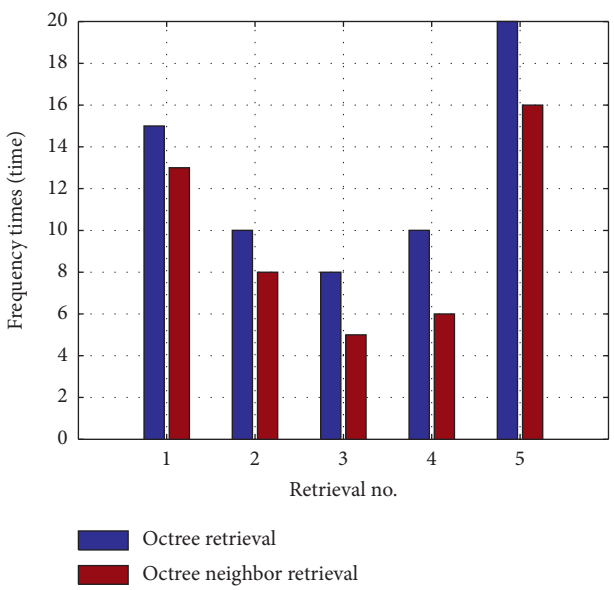


Figure 11: The number of data retrievals in scene B.

line within the object data range was analyzed. The linear octree was improved to have a backtracking ability. When the dividing line occurred in the object, the object data was not divided and instead it got stored in the nearest parent node that could fully contain the object data, not just in leaf nodes. Ultimately, it led to strengthening the stability of data storage and improving the efficiency of storage. After experimental verification, the data storage model of the octree map did not consume much time compared with the traditional octree, which is 1/8 of the time consumed by R-tree structure. Moreover, considering the problem of high cost and low efficiency of data retrieval, a data retrieval method based on the octree storage model was proposed in this paper. The method mainly analyzed from two aspects. Initially, according to the feature of "searching for a point, points around the searched point are within the searching range" of map data retrieval, an octree neighbor nodes retrieval method was proposed to reduce the number of retrieval operations. Then, by adding closure constraint conditions, the neighbor nodes retrieval method was improved to decrease the cost of data retrieval. Then, we could improve the efficiency of data retrieval. Experimental results showed that the cost of retrieval was reduced by about 25%.

Due to the huge amount of indoor map data, the consumption of storage is crucial to the promotion and application of the system. The index linear storage based on octree

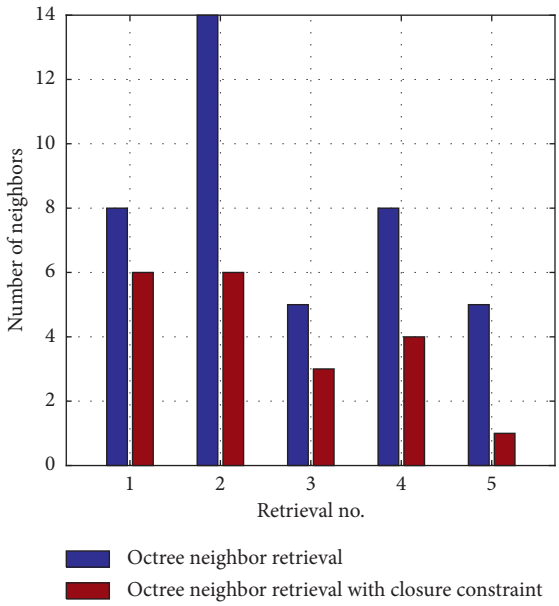


Figure 12: The number of neighbors in scene A.

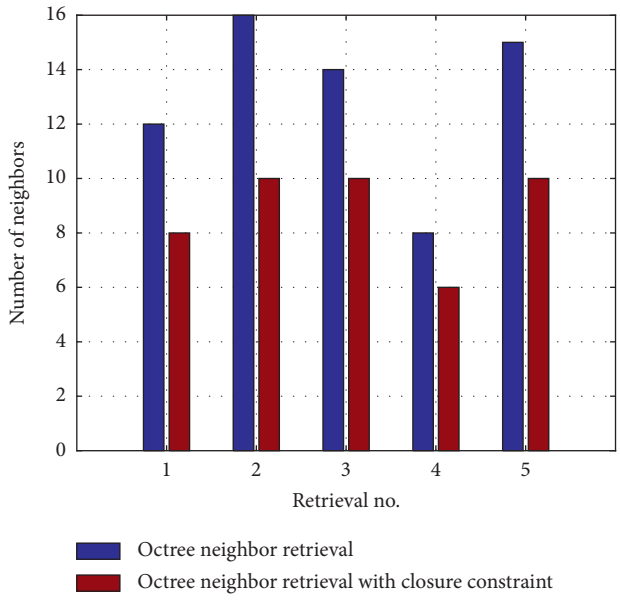


Figure 13: The number of neighbors in scene B.

can save a lot of storage space, reduce storage costs by 60%, and save system development costs. Meanwhile, the neighbor retrieval method proposed by adding closed constraints improved the retrieval efficiency by about 25%. The method of map data storage and retrieval proposed in this paper saved storage space and improved retrieval speed in practical applications. The indoor map was used in indoor localization to display the positioning results in indoor scenes, so that users would have a good positioning experience. With the advent of the 5G era, indoor localization has developed rapidly, and the search efficiency of indoor maps will be increasingly demanded. This research can achieve the purpose of real-time display of positioning results in map scene.

There are many potential future directions existing for this type of work. First, in terms of map data expression, AABB packaging box approximate expression method was adopted in this paper, which can meet current needs. However, for objects with special shapes, some segmentation methods based on point-cloud data can be adopted to improve the accuracy of data expression. Second, there are vast amounts of data and many concurrent operations such as data retrieval in the indoor map. Therefore, cloud storage method and distributed system architecture can be combined to improve the robustness of data storage and retrieval, which could enhance the application experience of users.

Data Availability

The datasets used and analyzed during the current study are available from the first author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (no. 61872104), the National Science and Technology Major Project of China (no. 2016ZX03001023-005), the China Postdoctoral Science Foundation (no. 2019M651264), the Natural Science Foundation of Heilongjiang Province of China (no. 2015023), and the Basic Business Project in Education Department of Heilongjiang Province of China (nos. 135109243 and 135209237).

References

[1] L. Jiang, L. D. Xu, H. Cai et al., "An IoT-oriented data storage framework in cloud computing platform," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1443–1451, 2014.

[2] S. Bemel-Benrud, T. C. MacWright, E. Halperin et al., "Providing visual selection of map data for a digital map," U. S. Patent Application: US20180052593A1, 2018.

[3] S. Har-Noy, L. Barrington, and N. Ricklin, "System and method for large scale crowdsourcing of map data cleanup and correction," U. S. Patent Application: US20160004724A1, 2016.

[4] H. Lu and M. A. Cheema, "Indoor data management," in *Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering*, IEEE, Helsinki, Finland, pp. 1414–1417, May 2016.

[5] S. Malhotra, M. N. Doja, B. Alam, and M. Alam, "Skipnet-octree based indexing technique for cloud database management system," *International Journal of Information Technology and Web Engineering*, vol. 13, no. 3, pp. 1–13, 2018.

[6] Y. Guo, "Method and device for constructing spatial index of massive point cloud data," U.S. Patent Application: US20180081034A1, 2018.

[7] Y. Wanqiang, Z. Junliang, C. Peng et al., "An octree-based mesh simplification algorithms for 3-dimension cloud data," *Science of Surveying and Mapping*, vol. 41, pp. 18–22, 2016.

[8] R. Kothuri and S. Ravada, "Oracle spatial, geometries," *Encyclopedia of GIS*, pp. 1–9, Springer, Berlin, Germany, 2015.
[9] L. I. Zhen-Ju, L. I. Xue-Jun, Y. Sheng et al., "Spatial index built in cloud computing environment," *Geomatics & Spatial Information Technology*, vol. 38, no. 10, pp. 13–17, 2015.
[10] Z. X. Qiao, "Application and research on spatial index technology," *China Place Name*, vol. 6, pp. 72-73, 2014.
[11] C. Yang, "Research on fast index technology of spatial data," *Electronic Technology & Software Engineering*, vol. 19, pp. 207-208, 2018.
[12] Y. Xiao, J. T. Li, W. J. Zhang et al., "A spatial index structure of natural neighbor relationship query," *Geomatics World*, vol. 25, no. 1, pp. 32–38, 2018.
[13] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, "Authenticated indexing for outsourced spatial databases," *The VLDB Journal*, vol. 18, no. 3, pp. 631–648, 2009.
[14] W. Wang, Z. Xuan, L. Sun, Z. Jiang, and J. Shang, "BRLO-tree: a data structure used for 3D gis dynamic scene rendering," *Cybernetics and Information Technologies*, vol. 15, no. 4, pp. 124–137, 2015.
[15] M. H. Namdari, S. R. Hejazi, and M. Palhang, "MCPN, octree neighbor finding during tree model construction using parental neighboring rule," *3D Research*, vol. 6, no. 3, p. 29, 2015.
[16] W. Solihin, C. Eastman, and Y. C. Lee, "Multiple representation approach to achieve high-performance spatial queries of 3D BIM data using a relational database," *Automation in Construction*, vol. 81, pp. 369–388, Article ID S0926580517302388, 2017.
[17] P. F. Wang, Z. Y. Ding, A. G. Teng et al., "Research on the application of quadtree index in Jiangsu power grid GIS platform," *Power Information and Communication Technology*, vol. 15, no. 7, pp. 87–91, 2017.
[18] D. J. R. Meagher, "The octree encoding method for efficient solid modeling," Technical Report IPL-TR-032, Image Processing Lab. Rensselaer Polytechnic Institute, New York, NY, USA, 1982.
[19] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, 1982.
[20] S. Sengupta and P. Sturgess, "Semantic octree: unifying recognition, reconstruction and representation via an octree constrained higher order MRF," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Seattle, WA, USA, pp. 1874–1879, May 2015.
[21] Y.-T. Su, J. Bethel, and S. Hu, "Octree-based segmentation for terrestrial LiDAR point cloud data in industrial applications," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 113, pp. 59–74, 2016.
[22] P. Beno, V. Pavelka, F. Duchon et al., "Using octree maps and RGBD cameras to perform mapping and A* navigation," in *Proceedings of the 2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, pp. 66–72, IEEE, Ostrava, Czech Republic, September 2016.
[23] J. Yun, Y. S. Lee, and Y. Jung, "Development of a gap searching program for automotive body assemblies based on a decomposition model representation," *Advances in Engineering Software*, vol. 81, pp. 7–16, 2015.
[24] G. Riegler, A. O. Ulusoy, and A. Geiger, "Octnet: learning deep 3D representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 3, Honolulu, HI, USA, July 2017.
[25] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: efficient convolutional architectures for high-resolution 3d outputs," 2017, https://arxiv.org/abs/1703.09438.
[26] Z. Deng, L. Wang, W. Han et al., "G-ML-octree: an update-efficient index structure for simulating 3D moving objects across GPUs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 5, 2017.
[27] D. Taniar and W. Rahayu, "A taxonomy for nearest neighbour queries in spatial databases," *Journal of Computer and System Sciences*, vol. 79, no. 7, pp. 1017–1039, 2013.
[28] C. H. Chen, "A cell probe-based method for vehicle speed estimation," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. 1, no. E130A, pp. 265–267, 2020.
[29] X. Chen, N. Liu, Y. Su, and G. Chen, "A survey of swarm intelligence techniques in VLSI routing problems," *IEEE Access*, vol. 8, no. 8, pp. 26266–26292, 2020.
[30] G. G. Liu, Z. S. Chen, Z. Zhuang et al., "A unified algorithm based on HTS and self-adapting PSO for the construction of octagonal and rectilinear SMT," *Soft Computing*, vol. 24, pp. 3943–3961, 2020.
[31] M. Pan, Y. Liu, J. Cao et al., "Visual recognition based on deep learning for navigation mark classification," *IEEE Access*, vol. 99, p. 1, 2020.
[32] M. Pan, J. H. Zhou, H. Cao et al., "Water level prediction model based on GRU and CNN," *IEEE Access*, vol. 8, no. 8, pp. 60090–60100, 2020.
[33] W. X. Li, C. Y. Zhang, G. G. Liu et al., "Extraversion measure for crowd trajectories," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 15, pp. 6334–6343, 2019.
[34] C. H. Chen, F. J. Hwang, H. Y. Kung et al., "Travel time prediction system based on data clustering for waste collection vehicles," *IEICE Transactions on Information and Systems*, vol. 7, no. E102D, pp. 1374–1383, 2019.
[35] C.-H. Chen, "An arrival time prediction method for bus system," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4231-4232, 2018.
[36] S. Cho, S. Park, G. Cha, and T. Oh, "Development of image processing for crack detection on concrete structures through terrestrial laser scanning associated with the octree structure," *Applied Sciences*, vol. 8, no. 12, p. 2373, 2018.
[37] B. Ummenhofer and T. Brox, "Global, dense multiscale reconstruction for a billion points," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1341–1349, Santiago, Chile, December 2015.
[38] F. Steinbrücker, J. Sturm, and D. Cremers, "Volumetric 3D mapping in real-time on a CPU," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Hong Kong, China, pp. 2021–2028, May 2014.
[39] W. Jinqiu, Q. Gang, and K. Pengbin, "Emerging 5G multi-carrier chaotic sequence spread spectrum technology for underwater acoustic communication," *Complexity*, vol. 2018, Article ID 3790529, 7 pages, 2018.
[40] C. H. Chen, F. Y. Song, F. J. Hwang et al., "A probability density function generator based on neural networks," *Physical A: Statistical Mechanics and Its Applications*, vol. 3, no. 541, Article ID 123344, 2020.