Hindawi
Mobile Information Systems
Volume 2018, Article ID 4175298, 15 pages
https://doi.org/10.1155/2018/4175298

Research Article

Vertical Indexing for Moving Objects in Multifloor Environments

Sultan Alamri,[1] **David Taniar**,[2] **and Kinh Nguyen**[3]

[1]*College of Computing and Informatics, SEU, Riyadh, Saudi Arabia*
[2]*Clayton School of Information Technology, Monash University, Melbourne, VIC, Australia*
[3]*Department of Computer Science and Computer Engineering, La Trobe University, Melbourne, VIC, Australia*

Correspondence should be addressed to Sultan Alamri; salamri@seu.edu.sa

Received 15 August 2017; Revised 22 October 2017; Accepted 1 November 2017; Published 28 January 2018

Academic Editor: Laurence T. Yang

The indexing and tracking of objects moving in indoor spaces has increasingly become an important area of research, which presents a fundamentally different challenge. There are two main reasons for why indoor should be treated as cellular space. Firstly, an indoor space has entities, such as rooms and walls, that constrain the movement of the moving objects. Secondly, the relevant notion of locations of an object is cell based rather than an exact Euclidean coordinate. As a solution, in our earlier works, we proposed a cell-based indexing structure, called the C-tree, for indexing objects moving in indoor space. In this paper, we extend the C-tree to solve another interesting problem. It can be observed that many indoor spaces (such as shopping centers) contain wings/sections. For such a space, there are queries for which the wing/section location of an object, rather than the cellular location, is the relevant answer (e.g., "the object is in the east wing"). In this paper, we propose a new index structure, called the GMI-tree ("GMI" stands for "Graph-based Multidimensional Index"). The GMI-tree is based on two notions of distance, or equivalently, two notions of adjacency: one represents horizontal adjacency and the other represents vertical adjacency.

1. Introduction

Indoor environments represent a promising area for the indexing and querying of mobile/moving objects in spatial databases [1–4]. People conduct most of their day-to-day activities in indoor environments, such as entertaining, living, working, and shopping. Studies have shown that people spend around 80% of their lives indoors [2, 5–9].

In addition, indoor environments are rapidly becoming larger and more complex. For example, the Beijing subway is a rapid transit rail networks with 227 stations and passengers numbering in excess of 7 million daily. Consequently, the positioning and monitoring of mobile objects indoors on different floors have become an essential research field with many applications for the location of mobile objects, indoor multifloor wayfinding, and security [10–14].

Until now, a great deal of research has been devoted to the indexing, querying, and tracking of objects moving in outdoor spaces. Unfortunately, such works often cannot be applied to objects moving in indoor spaces [15]. Indoor spaces contain entities such as walls and doors which play an important role in controlling and restricting the movement of an object in the environment. Also, the cells (e.g., rooms) which can contain objects are entities of primary interest in indexing and querying. For example, a query may be performed to locate object O_5; in an indoor space, the logical answer is the room/cell in which O_5 is residing (e.g., room/cell 313). In this case, the exact locations of the moving objects, in terms of the numerical values of their coordinates, are not important. That is, indoor space is not to be treated as Euclidean space or as a spatial road network. It is also relevant to note that indoor spaces are not compatible with GPS tracking because of the inaccuracy of GPS systems in indoor environments [8, 16–18]. Given these factors, the notion of position to be used for indoors is based on the notion of cell-based space (or cellular space). Consequently, the current moving object researches that are based on metric distribution cannot be applied to an indoor situation [19–23].

In light of the starkly different nature of indoor space, compared to outdoor space, a fundamental challenge is how to build a cell-based index that can efficiently serve the purpose of querying and tracking of objects moving in

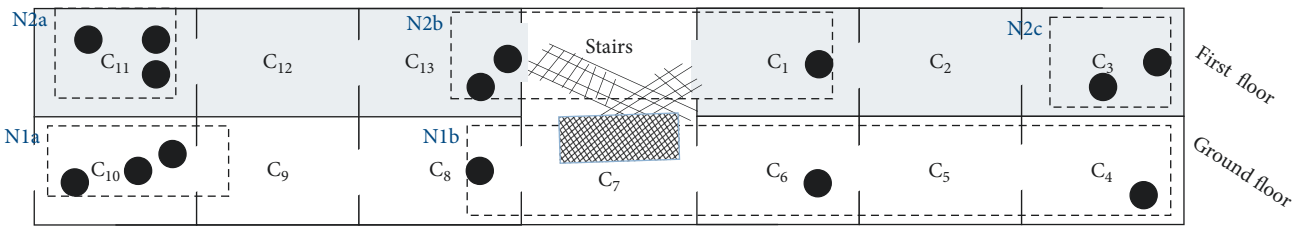

Figure 1: N2b and N1b are crossed-over nodes between the wings.

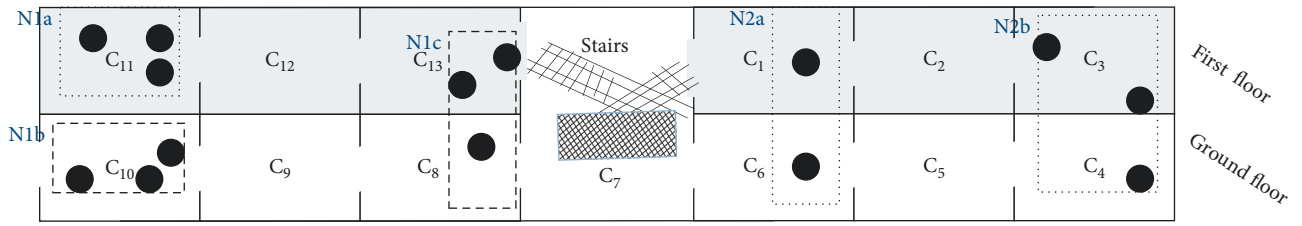

Figure 2: The moving objects are grouped in each wing.

indoor spaces. From a conceptual viewpoint, an indoor space can be regarded as a graph which consists of nodes (representing cells and rooms) and edges (representing connections between cells such as doors and stairways). Based on this model of indoor space, we propose a method for constructing the cell-based index, which is called the C-tree ("C" for "cell"). The work has been reported in [24]. The method for constructing the C-tree, together with associated concepts, will be described in Section 3.

In addition to our proposed solution to the fundamental problem described above, the C-tree can also be extended to provide a solution to another problem, which is the subject matter of this paper. Many queries can be raised in multifloor buildings which can be considered as multifloor queries regardless of the floor. Therefore, some queries can be raised such as "Where is object O_3?"; however, the exact location (in terms of cell) of the object may not be important at all, whereas the expected answer may be "north section/wing." Moreover, queries regarding the objects nearest to a certain lift can be considered as RNN queries in multifloor environments (detail with examples in Section 3). These types of queries raise the question whether grouping the moving objects vertically in addition to the traditional grouping (horizontally) can achieve efficiency of the indexing and querying processing. The challenge now is how to construct an index that is based on the primary location concept of cells while taking into account the secondary location concept of wing/section. Or, put simply, how can we build an index that can efficiently support queries regarding cells, but at the same time can support the wing queries and the multifloor RNN queries and pool queries?

As a solution, we proposed a mobile objects index structure called the GMI-tree ("GMI" stands for "Graph-based Multidimensional Index"), in which we extend the C-tree approach to account for more than one concept of adjacency (in the same or nearby cell and in the same wing/section). If the index structure is based solely on the cell-based distance (as treated in the previous section), then it will require many verification steps in order to reach the targeted answer (in case of wing or multifloor RNN queries) (see Figure 1). This is obvious since, if the moving objects are grouped close to each other according to cell-based distance, then the index structures will need to check all the descending nodes on each floor individually, thereby incurring high access costs. In addition, it will cause many crossover nodes between the indoor areas of a building. That is, nodes might cross a wing to another in order to group the moving objects as in Figure 1, whereas in Figure 2 the objects are grouped in each wing without any crossover.

We call the new index "multidimensional" because, as will be seen in Section 4, our solution is based on two types of links: the usual "horizontal" links between the cells connected by doors (called the primary links) and the newly introduced "vertical" links which involve cells that are directly above or below one another (called the secondary links).

Our major contributions are as follows:

(i) We propose a model for representing indoor spaces.

(ii) Based on that model, we propose the C-tree as a fundamental indexing method for moving objects in indoor spaces.

(iii) We carry out experiments to thoroughly evaluate the C-tree approach.

(iv) We extend the basic indoor space model for multidimensional grouping (e.g., based on cells and wings) and identify typical situations where such a model can be usefully applied.

(v) We develop a mobile object index structure for multidimensional grouping, the GMI-tree.

(vi) Under a simulation environment, we conduct experiments in order to evaluate the proposed index structure of the GMI-tree by studying the costs of construction and insertion, and query performances.

The rest of the paper is organized as follows. Section 2 presents works related to the indexing of moving objects. Section 3 describes and illustrates the basic method for indexing objects moving in indoor spaces: the C-tree. Section 4 presents the structure and construction of the GMI-Tree, our proposed solution for indexing objects moving in indoor spaces which involves multidimensional grouping. Section 5 presents the results of a performance evaluation of the proposed GMI-tree. Section 6 concludes the paper and suggests possible directions for future work.

2. Related Work

As mentioned above, since the development of indoor positioning systems, the importance of indexing and querying mobile objects in indoor environments has become more significant. A number of works have been published in this area, although to the best of our knowledge, these are few in number.

The earliest paper on this topic is [7]. They proposed two types of index trees: RTR-tree and TP2R-tree. Both are based on *the positions of RFID readers* that are installed in the indoor environments. The RTR-tree is based on the R-tree method of node organization. The RTR-tree includes the form (MBR and recordID), where MBR is identified by recordID. However, the trajectories are treated as line segments in the indoor space (see Figure 3). The insertion of new index entries into an RTR-tree is carried out similar to the R-tree; however, the calculation of the MBR area will be based on the RFID readers, where the calculation of the area of an MBR in the RTR-tree is Area = $(\text{readerID}_{\max} - \text{readerID}_{\min}) * (t< t>)$ [7]. In order to deal with a range query, the search will be conducted in the same way as for the R-tree; however, instead of searching on Euclidean space, the search will be done on a set of RFID readers.

On the other hand, the TP2R-tree changes the trajectories of data into a set of points in the indoor space which is different from the RTR-tree. The TP2R-tree can obtain fewer node accesses since it handles the case overflow nodes by the time extents. Here, the leaf node in the TP2R-tree includes the form (MBR, Δt, and recordID), where Δt is a time parameter that determines the duration of the continuous reading by the same reader. Note that the TP2R-tree has a better node organization than the RTR-tree does. A TP2R-tree is illustrated in Figure 4.

In [25], Lu et al. proposed a distance-aware indoor space structure which integrates indoor space distances. This structure provides an algorithm for *calculating the indoor space distances*. Indoor space rooms are connected by doors; therefore, a door connects two adjacent rooms (partitions). This distance-aware model proposed topology information mappings which basically map the indoor floor partitions and doors. Here, there are two concepts: first, D2P(D2Pdi) which can be unidirectional and bidirectional, if $|\text{D2P}(di)| = 1$ is considered as unidirectional, whereas $|\text{D2P}(di)| = 2$ is considered as bidirectional.

In our view, the three methods make some use of the characteristics of indoor space and associated tracking technologies. However, there are important characteristics that are not taken into account. An indoor space has rooms, doors, walls, corridors, staircase, floors, etc. At least at a high level of abstraction, it consists of a set of cells (rooms, corridors, etc.) and connections between them (doors and staircases). Our idea is to build index trees that are based on these essential characteristics.

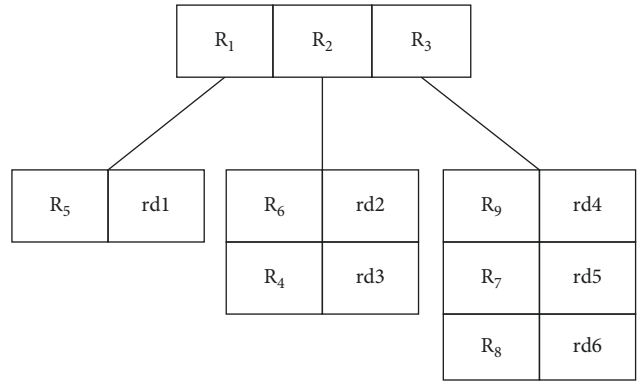

Figure 3: Example of the RTR-tree.

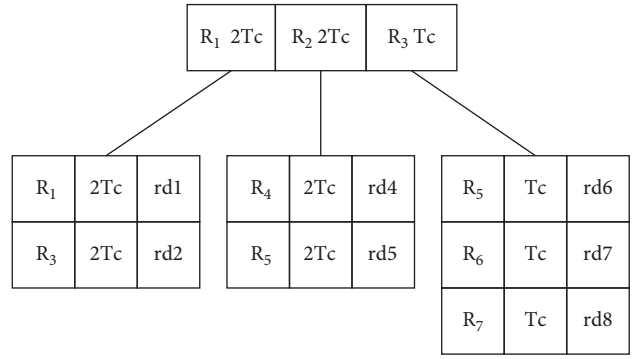

Figure 4: Example of the TP2R-tree.

Therefore, we proposed the C-tree which is efficient for calculating the moving objects in 2D space [24]. The C-tree considers the uniqueness of the indoor space which contains partitions, walls, doors, and corridors. The C-tree represents indoor space as a connectivity graph. It also proposes an indexing method that is based on the concepts of cells and connections between cells. We call it a C-tree, "C" for "cell." The basic methodology for the C-tree is to construct the moving objects in the indoor floor as a horizontal grouping, whereby moving objects will be grouped with its adjacent moving objects in the same cell or in adjacent cells.

However, the methodology used for the C-tree can be efficient in some indoor queries such as positioning queries or Knn. However, in the case of some buildings, the importance of the position of mobile objects depends on the structure of the building. In some buildings (such as shopping centers), the positioning queries can be obtained by wings/sections of the building. In these cases, the locationing of any moving objects based on their wing location might be more useful to the user. Therefore, we proposed the GMI-tree in order to group moving objects both horizontally and vertically. Hence, we have a multidimensional

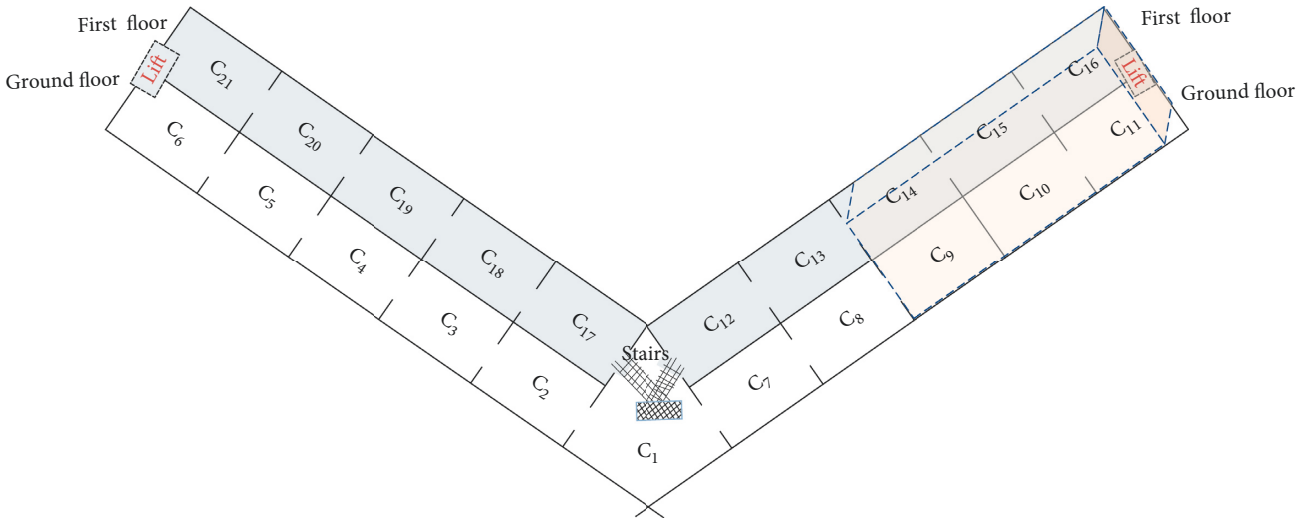

Figure 5: Example of an indoor space.

connectivity graph which considers the moving objects construction in a multidimensional way. Note that the concepts and techniques for C-Tree will be essential for the method/approach which is the subject of this work. Therefore, we devote Section 3 below to describe the method applied for the construction of the C-tree.

3. Basic Method for Indexing Object Moving in Indoor Space

In this section, we describe and illustrate our proposed method for indexing objects moving in indoor spaces. The outcome of the method is the C-tree, the indexing structure. Taking an example of an indoor space, we show

(i) how to convert the given indoor space into a connectivity graph;

(ii) how to derive the connectivity tree from the connectivity graph;

(iii) how to construct the C-tree.

3.1. Connectivity Graph. Consider the indoor space (a building) shown in Figure 5. An indoor space typically contains elements such as *rooms, doors, corridors, floors, stairs, elevators,* and *pathways* between buildings.

Our first step is to map the domain concepts (such as room, door, etc.) into the modeling concepts of cells and connections. The way to perform the mapping is summarized in Table 1. The result of the mapping is a *connectivity graph*, which represents the given indoor space (see Figure 6). More specifically, the graph is an *undirected graph* of *cells (nodes)* and *edges (connections)*, which, in addition, is a *connected graph* (i.e., there is a path between every pair of nodes).

3.2. Connectivity Tree. Given a connectivity graph representing an indoor space, the next step is to construct the connectivity tree, which is done as follows:

Table 1: Mapping domain concepts to modeling concepts.

Domain concepts	Modeling concepts
Room	A cell
Door	An edge
Corridor	One or more cells with one or more edges
Stair	One or more cells with one or more edges
Elevator	One cell with several edges
Pathway	One or more cells with several edges

(1) First we select one cell as the default cell of the connectivity graph. For our example, we choose cell C_1 in Figure 6.

(2) We then construct a spanning tree by performing a breadth-first search on the connectivity graph, starting from the default cell.

(3) We then order the siblings (nodes at the same level) of the spanning tree by the number of descendants. Those with more descendants are said to be higher. For siblings with the same number of descendants, we select an arbitrary order. In this way, all the nodes in the trees are ordered.

We refer to the tree, thus ordered, as the connectivity tree.

Example: for the indoor connectivity in Figure 6, we get the connectivity tree shown in Figure 7. C_1 is on level 1: it is the highest node according to our definition. On level 2, we have cells C_{12}, C_{17}, C_2, and C_7. The connectivity tree will be used as the major *input* for the construction of our C-tree.

Note that the connectivity tree not only represents the connections between the cells but also establishes an ordering of the cells. This ordering of cells is of fundamental importance to the construction of the C-tree. Formally, given a connectivity graph G and a default cell C, a connectivity tree T is a tree such that

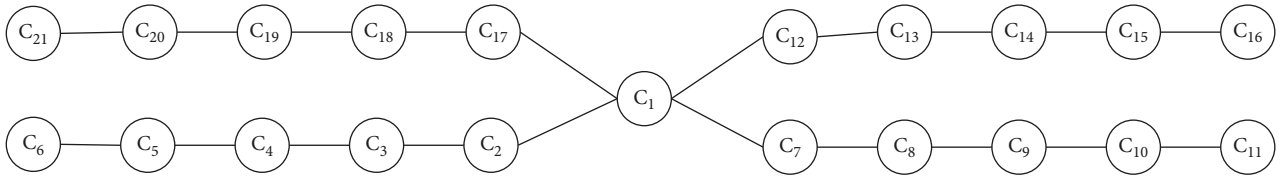

Figure 6: Connectivity graph for the indoor space of Figure 5.

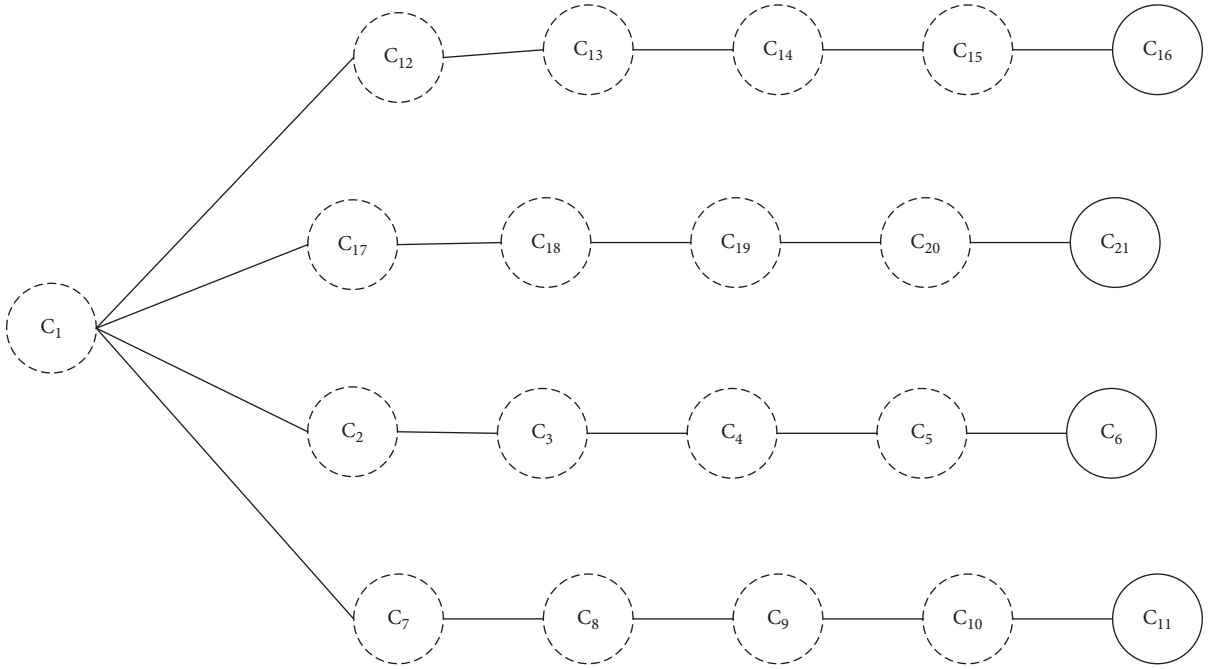

Figure 7: Connectivity tree for the connectivity graph of Figure 6.

(1) T is obtained as a spanning tree by performing a breadth-first search on G from default cell C;

(2) there is a function position that maps each cell (node) in the tree to a number such that

$$\forall C_i, C_j \in \text{Cells}, \tag{1}$$

$$\text{position}(C_i) > \text{position}(C_j) \Rightarrow \text{level}(C_i) < \text{level}(C_j) \vee, \tag{2}$$

$$\left(\text{level}(C_i) = \text{level}(C_j) \wedge \text{descendants}(C_i) \geq \text{descendants}(C_j)\right), \tag{3}$$

where level is the level of the cell on the tree and descendants is the number of descendants of the cell. Figure 7 shows an example of the connectivity tree.

Before describing the index construction algorithm, we define three terms that, though simple in their definitions, are very useful in showing how the index construction algorithm works.

3.2.1. Definition 1: Given a Connectivity Tree

(a) A node is an *expanding node* if it is a nonleaf node.

(b) A node is a *nonexpanding node* if it is a leaf node.

(c) The *expanding node of a node* C is C itself if C is an expansion node; otherwise, it is the parent of C.

Example: In Figure 7, all the expanding nodes are drawn as broken circles. The rest of the nodes are nonexpanding circles.

3.3. C-Tree. The construction of the C-tree uses as inputs: (a) a connectivity tree and (b) a set of objects which are distributed into the cells. The output is a C-tree, which has nonleaf nodes and leaf nodes, whose structures are described below.

A *nonleaf node* has two parts: (a) PTRs, which are pointers to the children of the node, and (b) the range of expanding cells, denoted by RC. The RC of a nonleaf node are the highest and the lowest expanding cells in that nonleaf node. Consider, for example, the N2 nonleaf node of the C-tree in Figure 8. Its RC is determined as follows:

(1) This node, regarded as a subtree, contains four objects O_6, O_7, O_8, and O_9.

(2) They are in cells C_9 and C_{11}.

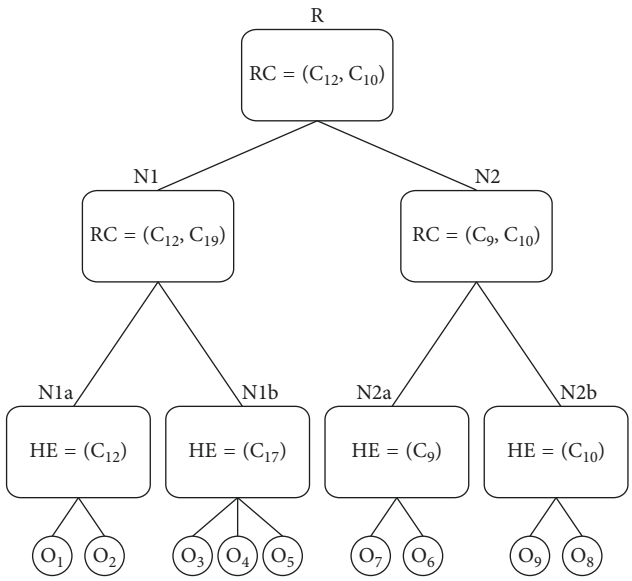

Figure 8: C-tree after inserting objects O_1, O_2, O_3, O_4,..., O_9.

(3) The expanding cells of those cells are C_9 (the highest) and C_{10} (the lowest).

(4) Hence, RC = (C_9, C_{10}).

A *leaf node* of the C-tree has two parts: (a) PTRs, which are pointers to the objects it contains, and (b) HE, which stands for "highest expanding cell." Consider, for example, the N1b leaf node in Figure 8. Its HE is determined as follows:

(1) This node contains objects O_3, O_4, and O_5.

(2) They are in cells C_{17} (the highest) and C_{19} (the lowest).

(3) Hence, HE = C_{17}.

We now illustrate how the C-tree is constructed. Consider, as an example, Figure 9, which shows a number of objects occupying the cells in the indoor space. We will be inserting the objects O_1, O_2, O_3, O_5, O_4, O_6, O_7, O_8, and O_9 into the C-tree in that listed order. We also assume that the C-tree is of order 3; that is, each node can have a maximum of three child nodes or objects.

Initially, the C-tree is empty. When inserting objects O_1 and O_2, the C-tree is as shown in Figure 10(a). It has one node, represented by a rectangle. The object, represented by a circle, is considered to be part of the node. The node has HE = (C_{12}).

The insertion of objects O_3 and O_5 requires splitting the node. The splitting is carried out as follows. First, we insert O_3 and O_5 into the root node (conceptually). This causes an overflow: there are more than the maximum of three objects. The node has to be split. The new objects are grouped together. Our strategy is to group the inserted object with the *nearest* object. In this case, O_3 and O_5 are the objects nearest to each other since they are in the same cell. With this grouping choice and using the splitting procedure of B+tree, we get the C-tree in Figure 10(b).

The insertion of object O_4, which is in cell C_{19}, requires us to move down the tree because the root has two children. For this purpose, we examine the HEs of the children which are the expanding nodes C_{12} and C_{17}. The strategy is to go with the *nearest* expanding node. In this example, we chose C_{17} since it is the nearest (two hops away). Note that the RC of the root node is now updated to be C_{12} which is the highest and C_{19} which is the lowest (see Figure 10(c)). This procedure will continue to insert the remaining objects to produce the final C-tree as shown in Figure 8.

4. Multidimensional Index: GMI-Trees

The importance of the position of moving objects can differ depending on building structures. Some buildings (such as airports and shopping centers) contain wings/sections that determine the success or otherwise of the positioning queries. From observation, in some buildings, the positioning of mobile objects can be based on their section [13]. For example, "Where is object O_9?" The accuracy of the position here is not important, because the query can obtain the answer "east wing." Moreover, here we note that the accuracy of the floor is not important in this case. Not only is the exact room/cell not needed but it could also be useless to the user. For instance, for the query "Where is object O_9?", answers such as "cell/room 15" could not be useful to the user who has no information about where cell 15 is located. However, answering with the wing location (e.g., it is in the south wing) might be more useful to the user.

Furthermore, the main reason for building a multidimensional indoor index is to obtain an index that can efficiently support queries regarding cells but at the same time can support the multifloor queries such as the wing queries and the multifloor RNN queries and pool queries. Wing queries have been explained in detail previously. The multifloor RNN queries, usually in indoor spaces, take the interests of a certain item then locate the objects that are nearest to it. For example, we may want the location of a moving object that is the nearest to Lift A. Here, the floor is not important, since a lift is basically located on all floors. Therefore, grouping the moving objects on both the same floor and on the floor above can obtain an efficient result for this type of query (the multifloor RNN queries). Another important query that can take advantage of our approach is a pool query. This type of query usually occurs in a multifloor indoor space; hence, it is a range query that intersects with many cells on different floors in order to retrieve the moving objects within that range. For example, in Figure 5, the dash lines show the retrieval of objects that are located within that range, which is shaped like a pool. Vertical and horizontal grouping is appropriate for all these queries and others that have moving objects in a multifloor environment, regardless of the floor(s) on which they are located.

4.1. Multidimensional Indoor Spaces. Our task, in this context, can be stated as follows: in addition to the primary queries that are based on cells as locations, how can we support multifloor queries such as wing queries, or multifloor RNN (for simplicity, we will use the term "wing" to include "section" and similar entities).

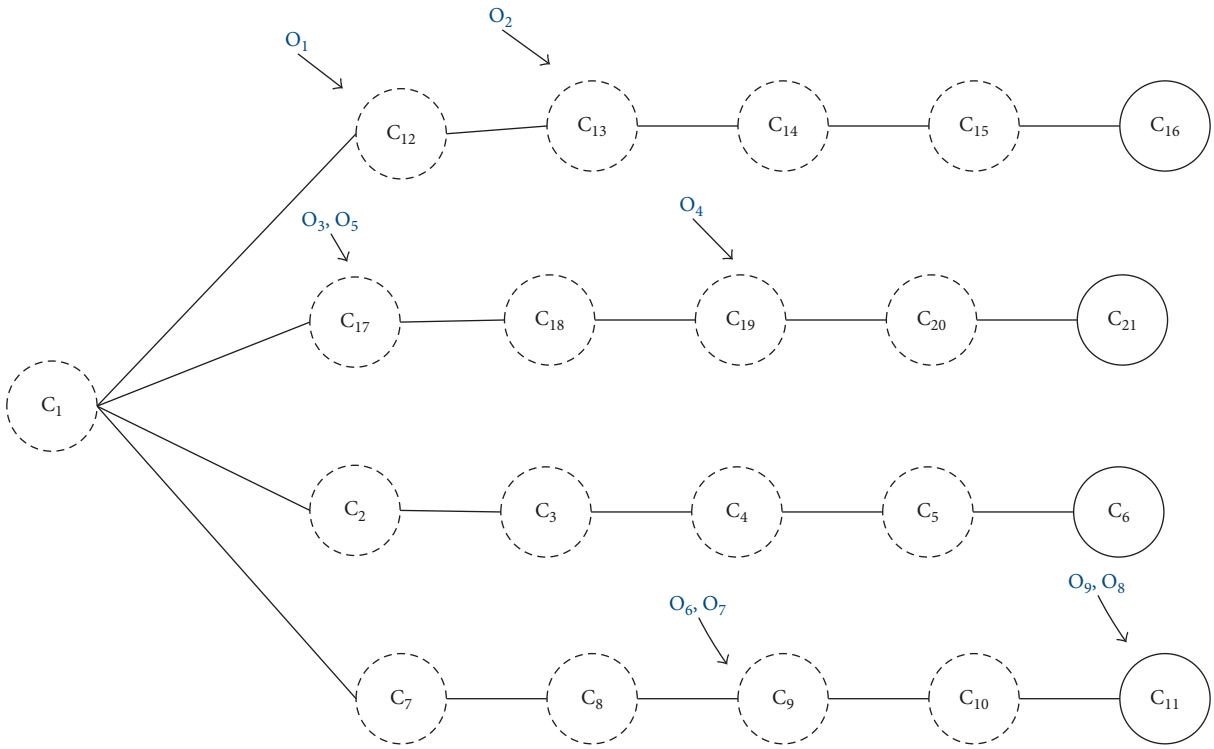

Figure 9: Occupancy of objects in the indoor space cells (shown on the connectivity tree).

Considering wing queries as an example, several issues arise. How are we to represent, or capture, the concept of "wing" in our model?

We will address this issue first. In doing this, we will consider, as an example, an indoor space which is clearly applicable to the queries.

Consider the indoor space in Figure 5. Observe that it has two wings connected by a single node and each of the wings has rooms on more than one floor.

For a multifloor building with wings, our proposed solution is introduced as an additional edge. Thus, the cells can be connected by primary edges and secondary edges. The primary edges are the same as those in the previous section: two cells (rooms) are connected by a primary edge if there is a door or staircase between them. The secondary edges, in this case, represent the "is_above" relationship. If two cells are on two adjacent floors and one is partly above another, we say that they are connected by a secondary edge. In our example (see Figure 5), cell C_{20} is directly above cell C_5; hence, there is a secondary edge between them. Recall that we will use the term *"multidimensional"* to refer to the existence of more than one type of edge.

4.2. Indoor Multidimensional Connectivity Graph. Given a multidimensional indoor space, that is, one for which we can conceive of two or more types of edges, our first step is to convert it into a multidimensional (MD) connectivity graph. Figure 11 shows the MD connectivity graph that represents the indoor space in Figure 5. The graph has primary edges, shown by solid lines between the cells, and secondary edges, shown by dotted lines. The primary edges are derived from the indoor space (Figure 5) as described in the previous section (see Table 1). The secondary edges between two cells are obtained by virtue of one cell being directly above the other. Note that the secondary "is_above" connection refers to the cells that are located above each other. This includes the case where a cell is connected vertically with more than one cell (vertically overlapped); the "is_above" connectivity will be made to all the overlapped cells regardless of the extent of the overlap.

4.3. MD Connectivity Tree. The MD connectivity tree is constructed from the MD connectivity graph by *ignoring* the secondary edges and applying the same procedure as described in Section 3 for constructing the basic (or 1D) connectivity tree. Figure 7 shows a connectivity tree for the MD connectivity graph shown in Figure 11. This connectivity tree on the surface looks exactly like a one-dimensional connectivity tree. Furthermore, it will play a similar role in the construction of the MD index tree. However, when using that connectivity tree in the index tree construction, we will take into account the secondary edges, which will have an effect on what we regard as the overall distance between pairs of cells (as given in Definition 2 below).

4.4. Constructing the Multidimensional Index Tree. The construction of an MD-tree is essentially the process of inserting an object into an existing MD-tree. This task has two major subtasks: (i) in which the new object is to be inserted and (ii) in the case where we need to split a node,

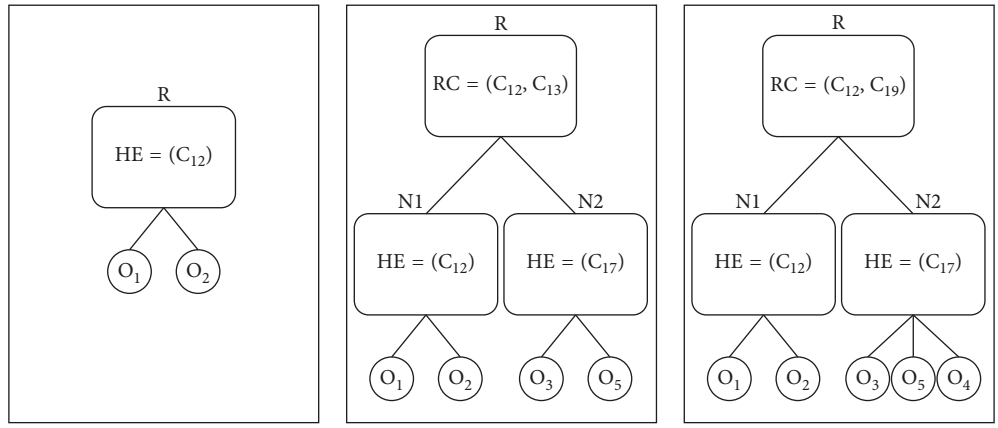

Figure 10: Index tree steps for inserting (O_1, O_2), (O_3, O_5), and O_4. (a) Inserting O_1 and O_2. (b) Inserting O_3 and O_5. (c) Inserting O_4.

choosing which objects (in the node) are to be grouped with the new object.

For navigation down the tree, we use exactly the same strategy as we did in Section 3. That is, suppose we need to insert an object occupying the cell C_{new}, we examine the RC and HE of the nodes, and at any particular stage,

(i) if possible, we choose the nonleaf node whose RC covers C_{new} according to the MD connectivity tree and which has an RC cell (either a lower bound or the upper bound) that is closest to C_{new};

(ii) otherwise, we examine the RC and HE of all the candidate nodes and choose the node that has an RC or HE cell which is closest to the cell C_{new}.

However, the distance between any pair of cells is now *modified* according to the definition below.

4.4.1. Definition 2. Let $d_P(x, y)$ and $d_S(x, y)$ denote the primary and secondary distances between cells x and y based on the number of primary edges and secondary edges between them. The distance between x and y is defined to be

$$d(x, y) = \begin{cases} d_P(x, y), & if \quad d_P(x, y) \leq d_S(x, y) \\ d_S(x, y) & \text{otherwise.} \end{cases} \quad (4)$$

Figure 12 shows the adjacency order of C_1 horizontally and vertically.

Apart from the above modification for calculating the distance between cells, the rest of the algorithm is the same as for the basic 1-dimensional case.

The algorithm for constructing the GMI-tree is given in Algorithm 1 and Algorithm 2.

4.5. MD Indexing-Applicable Indoor Spaces. In principle, we can apply the MD index construction procedure to any indoor space that is represented by an MD connectivity; that is, it has two types of edges. However, in practice, as will be seen in Section 5, the types of indoor spaces to which the MD procedure can be applied with desirable benefits are those with certain characteristics. To describe those characteristics, the notion of *cut node* defined below will be useful.

Consider a graph G of nodes (cells) and edges. Suppose we have a subset S of node G such that when we remove the nodes in S from the graph, what we get is a number of unconnected subgraphs. Then, we say that S is a set of cut nodes and each element of S is called a cut node. Formally, we have the following definition.

4.5.1. Definition 3. (Graph with Cut Nodes): Let G(Cells, Edges) be a connected undirected graph. Let S be a subset of nodes in cells. Then, we say that S is a set of cut nodes if the graph $G' = (\text{Cells}', \text{Edges}')$ is an unconnected graph, where

(i) $\text{Cells}' = \text{Cells} \backslash S$;

(ii) $\text{Edges}' = \{e \in \text{Edges} : \forall x \in S, x \notin e\}$.

(Note that an edge e is an unordered pair (i.e., a set) of two nodes.)

In addition, in practice, we also choose the set S of cut nodes that is minimal in the sense that if we remove any node from S, we would get a smaller number of islands.

Intuitively, and as will be seen in Section 5, the types of indoor spaces to which the MD procedure can be applied with desirable benefits are those with the following properties:

(1) It is a graph with cut nodes.

(2) The graph after the removal of cut nodes forms a number of islands (isolated subgraphs) of reasonable sizes.

(3) There are no secondary edges among the cut nodes (this is reasonable because cut nodes should not be part of a wing: they separate wings from one another).

(4) There are no secondary edges between nodes that belong to different islands (secondary edges are confined to separate islands).

Under those conditions, we can formalize the concept of a wing as follows.

Insert_Object /∗ To Insert an object into the current index tree. The tree is of order M = maximum number of child nodes ∗/
Data: The connectivity tree CONNECT_TREE; the Current index tree
INDEX_TREE: the object to be inserted OBJECT; the cell that is occupied by the object,
OBJECT_CELL **Result:** The updated index tree.
1: /∗ Find the leaf node to insert the new object ∗/
2: Let p = root of INDEX_TREE;
3: **while** p is not a leaf node **do**
4: | Construct the set EXPANDING_CELLS from all the RC and LE of the children of p;
5: | Choose from EXPANDING_CELLS, a cell NEAREST_CELL that is nearest to the OBJECT_CELL, according to $d(x, y)$ = IF $d_P(x, y) \leq d_S(x, y)$ THEN $d_P(x, y)$ ELSE $d_S(x, y)$;
6: | Choose a node Node that contains NEAREST_CELL as a bound of RC or LE;
7: | Let p = Node;
8: **endwhile**
9: /∗Insert the object into the leaf node p ∗/
10: **if** Node p is not full, i.e., number of objects is less than m **then**
11: | /∗ NOTE: Add this condition to definition of Index Tree ∗/
12: | Insert OBJECT into node p;
13: **else**
14: | Choose a set of $\lceil \frac{M}{2} \rceil$ existing objects in the node p, that are nearest to OBJECT_CELL.
15: | Call this set GROUPED_CELLS;
16: | Insert OBJECT and GROUPED_CELLS into a sibling node of p;
17: | Split the node, using the standard node-splitting procedure for B+Tree, and insert OBJECT and GROUPED_CELLS into a sibling node of p;
18: **endif**
19: For nodes that contain new objects, update RC and LE

Algorithm 1

CONSTRUCT_INDEX_TREE /∗ To construct an index tree for a set of objects ∗/
Data: The connectivity tree CONNECT_TREE; set of objects OBJECTS; function OCCUPIES : OBJECTS®CELLS **Result:** The index tree.
1: Let Index_Tree = ∅;
2: each object O_i of OBJECTS
3: | Call algorithm INSERT_OBJECT to insert O_i into INDEX_TREE;
4: **endfor**
5: **return** *Index_Tree;*

Algorithm 2

4.5.2. Definition 4. Assuming the context of the conditions above, two cells X and Y are in the same wing if and only if they belong to the same island.

Examples: Figure 11, which is the connectivity graph of an indoor space (Figure 5), shows that C_1 is the cut node: by removing C_1, we will have two unconnected subgraphs. Other examples of the cut nodes are shown in Figure 13.

As another example, Figure 14 shows a set of moving objects O_1, O_2,. . ., O_{10} in a multifloor indoor space. The GMI-tree will group the objects based on their section/wing as follows: The GMI-tree begins by grouping the moving objects that are located in the same cell and then starts to evaluate the adjacency levels in case of overflow or underflow. Therefore, O_4 is grouped with the above cell entity which is O_3. Figure 14 shows the GMI-tree of objects O_1, O_2,. . ., O_{10}. Note that the objects in the west wing are grouped together (O_1, O_2, O_3, and O_4) and east wing has the objects (O_5, O_6, O_7, O_8, O_9, and O_{10}). In this grouping technique, the distance will be based on the number of the hops between the cells (on both horizontal and vertical adjacent levels). Consequently, with this resulting index, queries that concern the objects' wing locations are processed easily with a lower access cost.

Regarding the search operation efficiency, the GMI-tree is very similar to the R-tree [1, 6] in three crucial aspects: (a) *the tree structure*, which is B-tree like; (b) *the tree construction process* in which the nodes are split in a similar manner to those of the B-tree; and (c) *the search procedure* (from the root, the search examines the children of a node N if the searched object is "contained" in the RC of N (for GMI-tree) or the MBB (for R-tree)). Consequently, similar to the R-tree, the worst case complexity of a GMI-tree search is O(n) due to overlapping RCs, and the average case complexity can be expected to be O(log(n)). In addition, for a given connectivity graph, different choices of the default cell can result in connectivity trees of different shapes. This presents the possibility of performing a pre-tree-construction exploration to find a connectivity tree that is wide and shallow

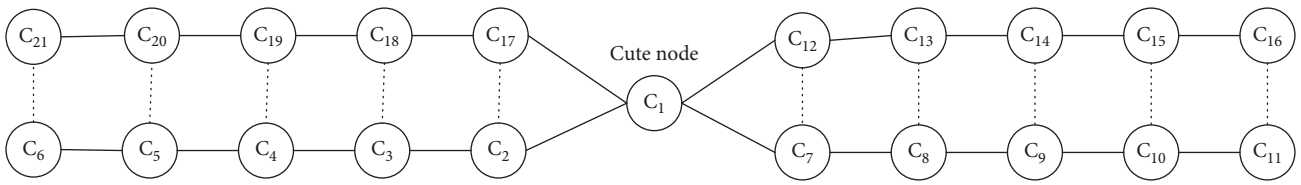

Figure 11: MD connectivity graph for the indoor space in Figure 5.

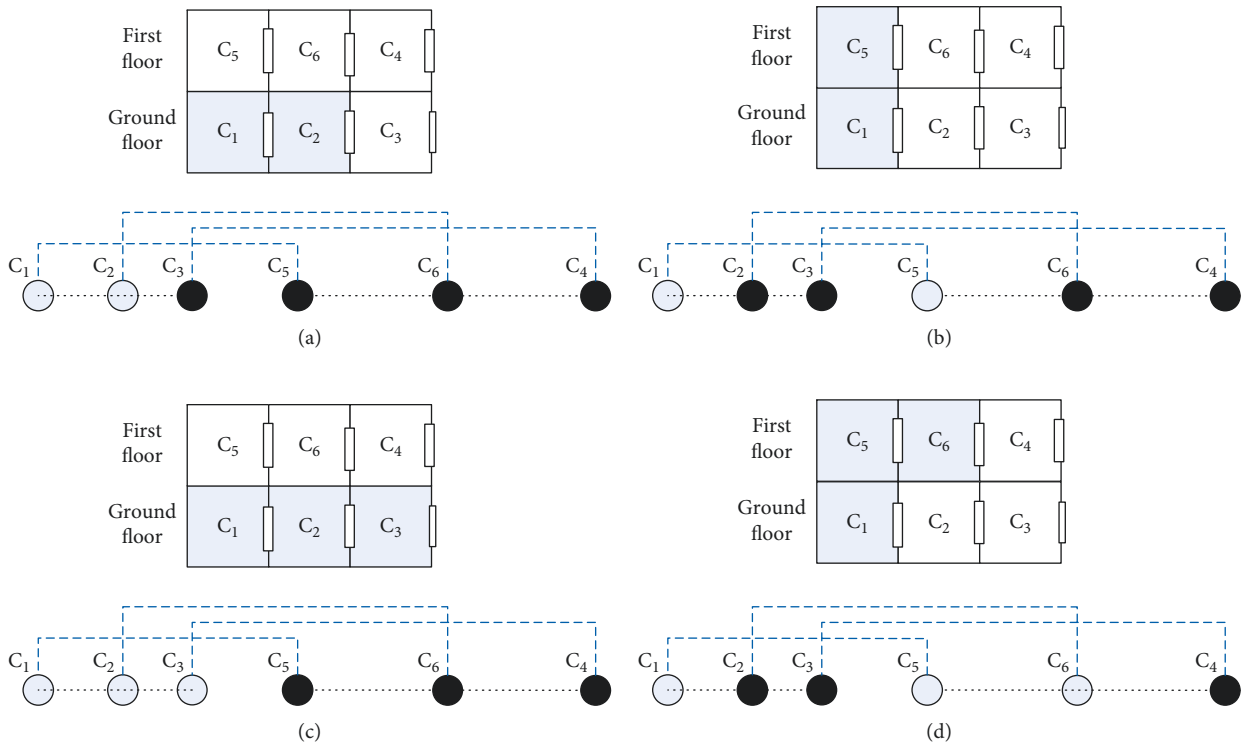

Figure 12: The adjacency levels of C_1. (a) First level adjacent of C_1. (b) Vertical level adjacent of C_1. (c) Second level adjacent of C_1. (d) Second vertical level adjacent of C_1.

because such a connectivity tree will tend to produce a GMI-tree with fewer overlapping RCs. We can also note that the cell connections, which are fixed for an indoor space, can be stored in an adjacency matrix and can be looked up very quickly for tree construction and querying.

5. Experimental Results and Performance Analysis

In this section, under a simulation environment, we present the results of experiments that have been conducted in order to evaluate the proposed index structure, the Graph-based Multidimensional Indoor-tree (GMI-tree). We compared this new structure with the Basic Index (C-tree) (which only groups the moving objects horizontally based on a single type of edge) [24]. The experiment was carried out on an Intel Core i5-2400S processor 2.50 GHz PC, with 4 GB of RAM running on 64-bit Windows 7 Professional. The maximum number of entries per node is $M = 80$ and the minimum is $m = 40$. The data structure was implemented in Java. The data set size ranged from 20 to 5000 moving objects in multifloor indoor spaces. We used synthetic datasets of moving objects due to the lack of real data for indoor environments. In the experiment, we used a real case of a 20-cell indoor space.

We will focus on the following features. First, we aim to compare the capacities of the GMI-tree and the basic C-tree in grouping the moving objects according to their primary distance and secondary distance. The grouping on primary distance is referred to as "horizontal grouping," and the grouping on secondary distance, "vertical grouping." Second, we compare the number of the crossover nodes between the two index trees. A crossover node is the one that groups together objects from different wings of the indoor space. Crossovers are regarded as undesirable "false hits." Third, we compare the costs of construction of the GMI-tree and the C-tree. And finally, we compare their insert and search performances. For all operations, the execution time was measured. Each operation was performed five times, and the average was calculated.

Figure 15(a) shows the number of nodes of the GMI-tree and the C-tree that group together the objects that are in close proximity according to their primary distance

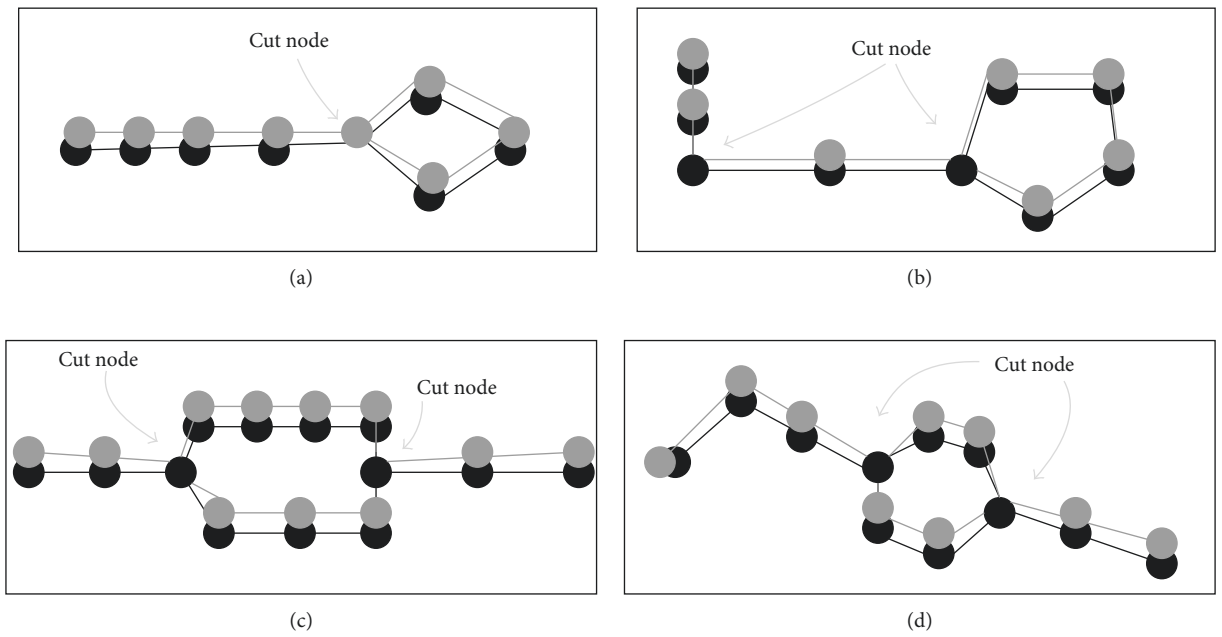

Figure 13: Cut nodes or articulation nodes.

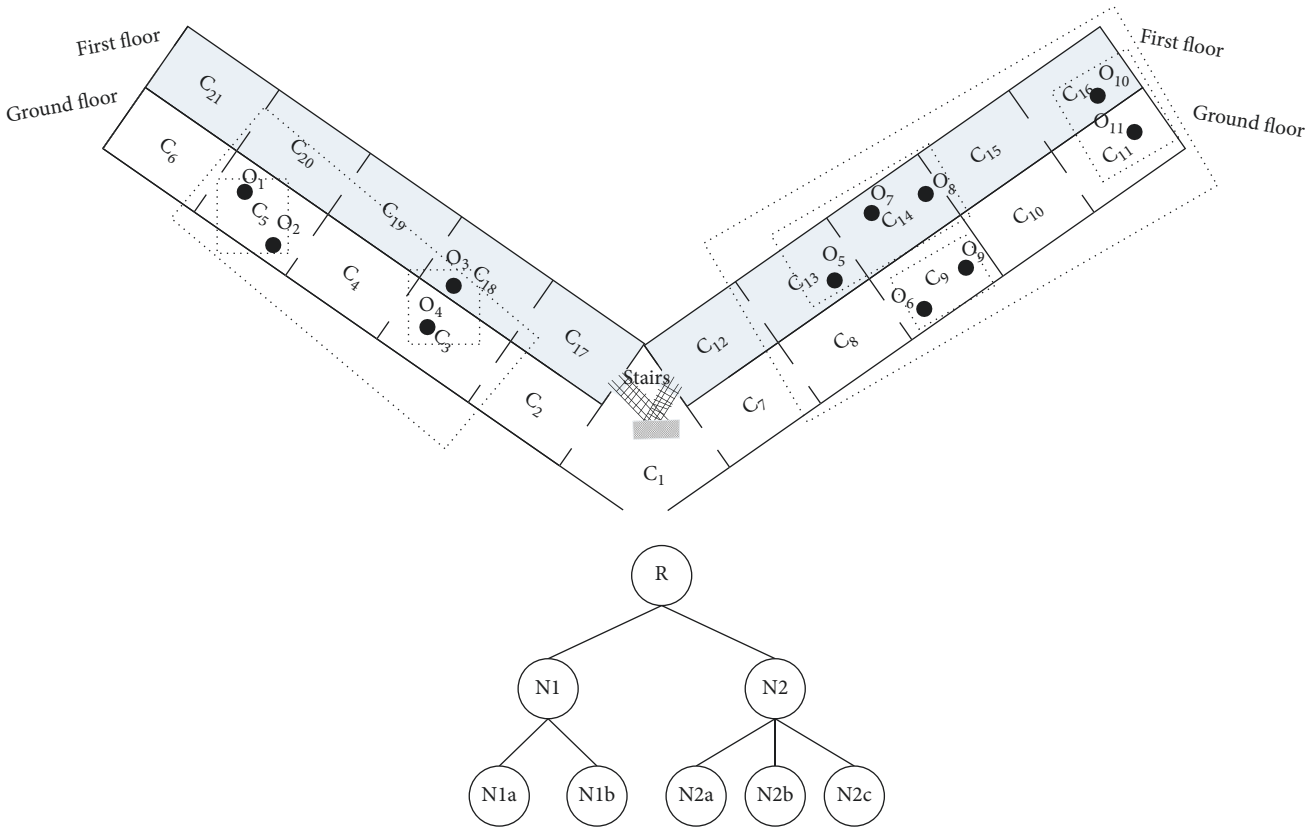

Figure 14: An example of Graph-based Multidimensional Indoor-tree.

("horizontal grouping"). The result shows that the GMI-tree (which is based on both the primary and secondary distances) performs virtually as well as the basic C-tree (which is based on the primary distance only). This result is very encouraging because we would want to maintain good efficiency for the primary horizontal grouping. Figure 15(b) shows the number of nodes that the GMI-tree and the C-tree group the moving objects vertically; that is, the number of

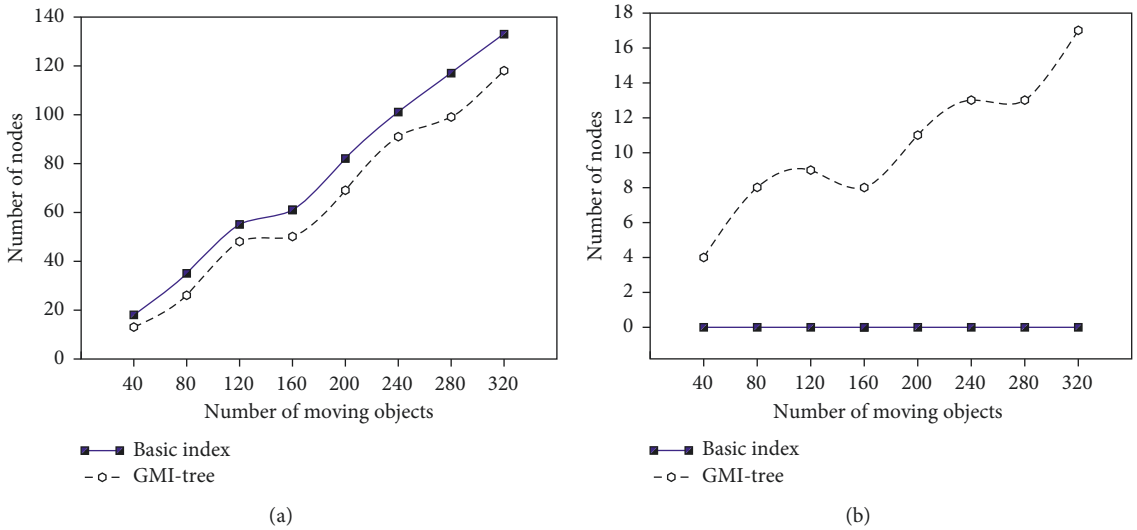

Figure 15: Horizontal and vertical grouping. (a) Number of nodes that horizontally grouped the moving objects. (b) Number of nodes that vertically grouped the moving objects.

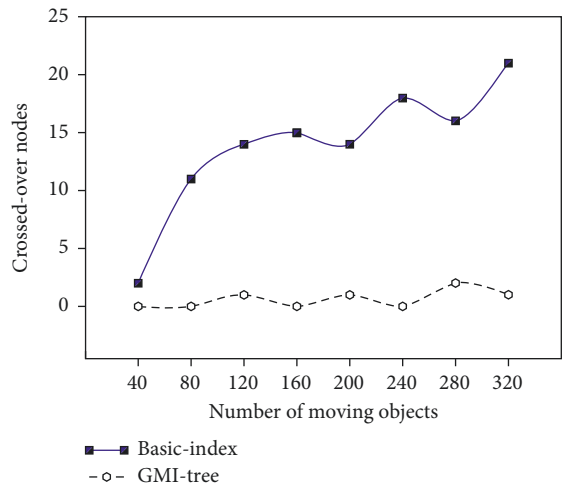

Figure 16: Crossed-over nodes between the building wings.

nodes which group together the objects that are close to one another according to their secondary (vertical) distances. The result shows that the GMI-tree performs clearly better than the basic C-tree in this respect. Though expected, this result proves the decidedly positive effect of the vertical grouping of the GMI-tree.

Taking both results together, they clearly illustrate that the GMI-tree has been successful in grouping the moving objects both horizontally and vertically. Consequently, it successfully groups objects that are close to each other in each of the wings of this indoor space with a cut node. Moreover, Figure 16 compares the crossover nodes of the GMI-tree with those of the basic C-tree. The crossover nodes are those that group together objects from the two different wings of the indoor space. The result shows that the GMI-tree groups the moving objects in each wing with a significantly very low crossover between the wings of the building.

Furthermore, here we investigate the measurement of the tree construction in both the R-tree and the proposed GMI-tree. The TP2R or RTR trees basically used R-tree through RFID readers. As explained in Section 2, the TP2R or RTR trees have a recordID on which the area of the MBR will be based. Here, the criteria are different from our prospective. They are entirely based on RFID in order to control the area of the MBRs. Therefore, we will not compare our work with TP2R or RTR trees, although we compare it with their baseline which is R-tree which similarly uses the Euclidean distance to group the moving objects. Here, we calculate the false hits of the mobile objects in an indoor environment for both the GMI-tree and the R-tree in order

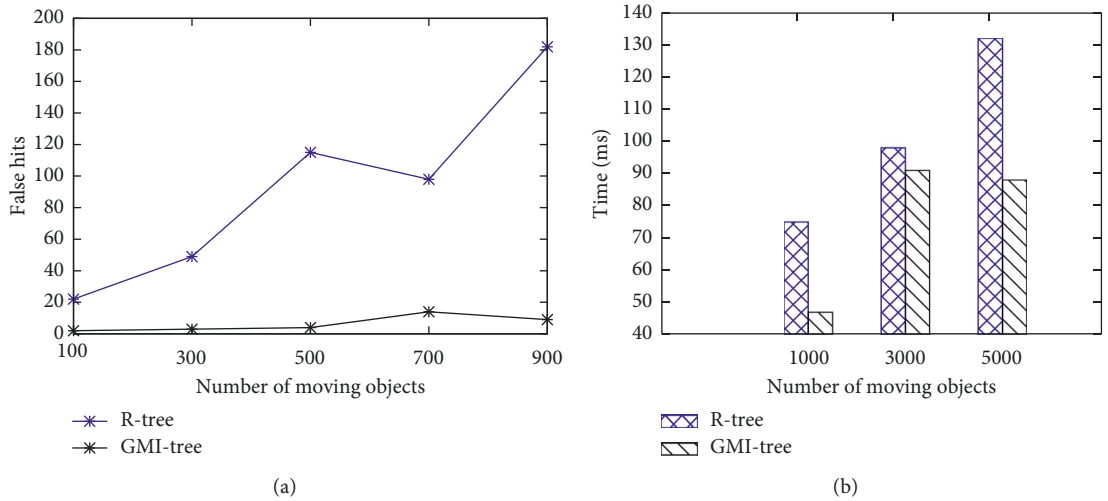

Figure 17: Metric performance (R-tree) compared with the GMI-tree. (a) False hits in the R tree and the GMI-tree. (b) Insert costs for the R-tree and the GMI-tree.

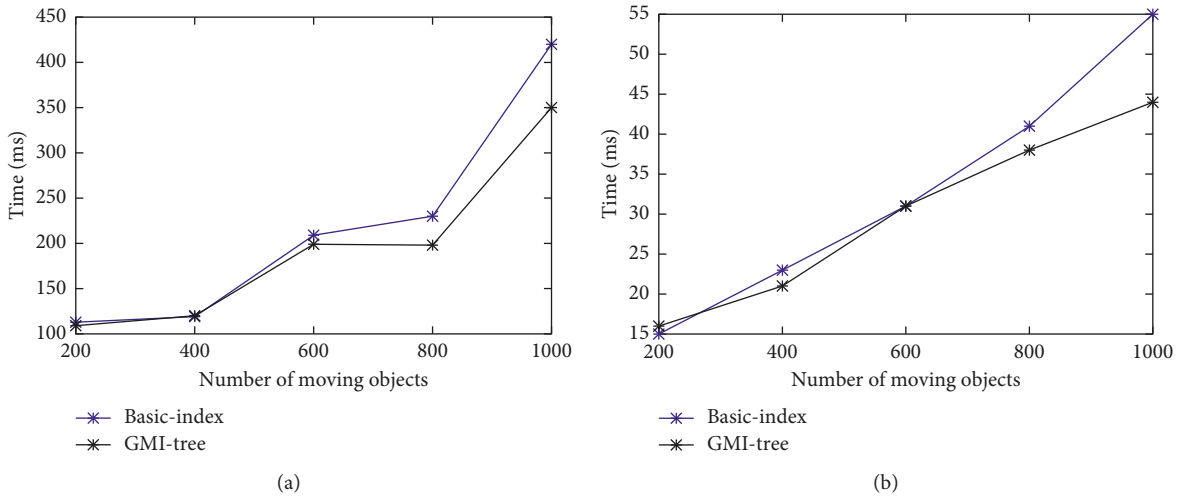

Figure 18: Construction and insert costs. (a) Construction cost. (b) Insert cost.

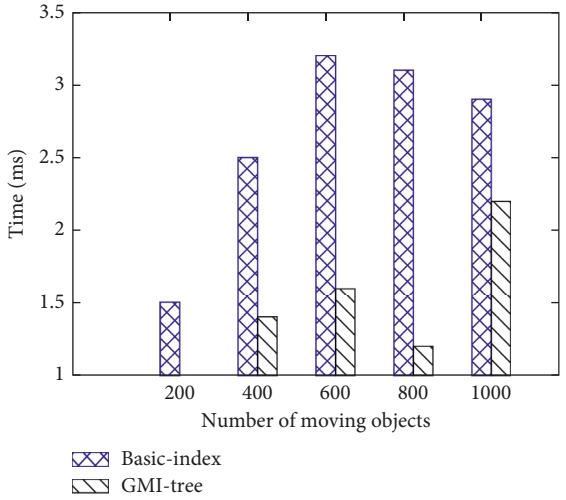

Figure 19: Search costs.

to illustrate the advantage of using the cell-based data structure in indoor space compared with any Euclidean-based data structures.

Figure 17(a) shows that many objects have been forced to be grouped based on the Euclidean distance when the R-tree data structure is used. However, in indoor environments, there are many walls and partitions which control the movement of moving objects. Therefore, using any metric index structures in indoor may lead to deficiency of the data structure through the many false hits that occurred. Moreover, the insert costs are shown in Figure 17(b). We notice in Figure 17(b) that in most cases the cost of the GMI-tree is lower than that of the R-tree insert costs. The GMI-tree takes into account the two ways of grouping (horizontally or vertically) which give the data structure more options, thereby reducing the cost significantly.

As the next objective in our experiment, we compared the construction costs and the insert costs of the GMI-tree and the C-tree. The result is shown in Figures 18(a) and 18(b). As we can note from the graph, the construction cost for the GMI-tree is no different from that of the Basic Index. The basic point here is that with the new method of indexing the mobile objects on the basis of two links of the indoor graph, we find that construction is still good. In fact, in most cases, the construction costs and the insert costs of the GMI-tree are less than those of the Basic Index. For example, in Figure 18(a), for 800 objects, the GMI-tree is decreased by around 28%.

As the last result to be reported here, we compared the search performance of the GMI-tree and the basic C-tree. The result is shown in Figure 19. From the graph, we can observe a significant improvement in the new index structure for all density cases. The improvement in search performance can be explained thus: In the GMI-tree, the moving objects are separately grouped based on their section through two links of the connectivity tree which makes the index tree more flexible and less costly. For example, Figure 19 shows that in the case of 600 densities, the search cost is reduced by over 44.

The experimental results clearly show that the newly proposed index structure, the Graph-based Multidimensional Indoor-tree, can successfully produce a reliable, robust, cost-effective, and query-efficient index for multifloor indoor spaces.

6. Conclusion and Future Work

This paper addresses the challenge of indexing moving objects as a multidimensional grouping in multifloor indoor environments. It is clear that the exact positions of objects moving about indoors are often not needed and cannot be regarded as a primary characteristic. The indoor environments are related to the symbolic notion of cellular space (or cell-based space), which contains restriction entities (e.g., walls) that play important roles in restricting an object's movements. Hence, the indoors locationing is based primarily on the notion of cellular space. In addition, it is also clear that in some interiors of buildings, the locationing of the moving objects can be based on their section/wing. Therefore, in this paper, we present a new index structure that considers the multidimensional grouping of mobile objects in a multifloor indoor space. More specifically, the indoor environment is a connectivity environment where the rooms/venues are connected with each other by doors; therefore, we take advantage of the graph that results from this primary indoor connectivity. We group the moving objects based on their adjacency on the same floor and in the same section (based on the adjacency/connectivity between the cells) and extend that to the grouping of the objects in each section as multidimensional grouping for the multilevel via a distance metric that is based on both the primary distance and secondary distance. The other basic idea is to determine the types of indoor spaces that can be advantageously considered for multidimensional grouping.

This work can be extended in several directions. First, the temporal aspects of objects moving in indoors are an important factor. The moving objects are expected to be more dynamic in corridors or on stairs, but more stabilized in a room. Therefore, the indexing of moving objects based on their temporal stabilization is an interesting area of study (considering vertical transits such as stairs). Another avenue of future work is the consideration of the data structure of moving objects with certain pattern movements in the indoor environment buildings. Thus, we aim to extend our index structure to involve certain movement patterns of moving objects in indoor environments.

Conflicts of Interest

Sultan Alamri now works at the College of Computing and Informatics, SEU, Saudi Arabia. The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] S. Alamri, D. Taniar, M. Safar, and H. Al-Khalidi, "A connectivity index for moving objects in an indoor cellular space," *Personal and Ubiquitous Computing*, vol. 15, no. 2, pp. 287–301, 2013.

[2] W. Yuan and M. Schneider, "Supporting continuous range queries in indoor space," in *Proceedings of the 2010 Eleventh International Conference on Mobile Data Management, MDM'10*, pp. 209–214, IEEE Computer Society, Washington, DC, USA, 2010.

[3] S. Alamri, "Indexing and querying moving objects in indoor spaces," in *Proceedings of the 2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW)*, pp. 318–321, IEEE Computer Society, Los Alamitos, CA, USA, 2013.

[4] T. Abeywickrama, M. Aamir Cheema, and D. Taniar, "k-nearest neighbors on road networks: a journey in experimentation and in-memory implementation," *Proceedings of the VLDB Endowment (PVLDB)*, vol. 9, no. 6, pp. 492–503, 2016.

[5] D. Lin, *Indexing and querying moving objects databases [Ph.D. Thesis]*, National University of Singapore, Singapore, 2006.

[6] S. Alamri, D. Taniar, M. Safar, and H. Al-Khalidi, "Spatio-temporal indexing for moving objects in an indoor cellular space," *Neurocomputing*, vol. 122, pp. 70–78, 2013.

[7] C. Jensen, H. Lu, and B. Yang, "Indexing the trajectories of moving objects in symbolic indoor space," in *Advances in Spatial and Temporal Databases Lecture Notes in Computer Science*, N. Mamoulis, T. Seidl, T. Pedersen, K. Torp, and I. Assent, Eds., vol. 5644, pp. 208–227, Springer, Berlin, Germany, 2009.

[8] M. Muñoz-Organero, P. J. Muñoz Merino, and C. Delgado Kloos, "Using bluetooth to implement a pervasive indoor positioning system with minimal requirements at the application level," *Mobile Information Systems*, vol. 8, no. 1, pp. 73–82, 2012.

[9] D. Sidlauskas, S. Saltenis, and C. S. Jensen, "Processing of extreme moving-object update and query workloads in main memory," *The VLDB Journal: The International Journal on Very Large Data Bases*, vol. 23, no. 5, pp. 817–841, 2014.

[10] Y. Li, H. Chen, R. Xie, and J. Z. Wang, "Bgn: a novel scatternet formation algorithm for bluetooth-based sensor networks," *Mobile Information Systems*, vol. 7, no. 2, pp. 93–106, 2011.

[11] H. Zhou, S. Xu, D. Ren, C. Huang, and H. Zhang, "Analysis of event-driven warning message propagation in vehicular ad hoc networks," *Ad Hoc Networks*, vol. 55, pp. 87–96, 2017.

[12] T. A. Dionti, K. M. Adhinugraha, and S. M. Alamri, "Inter-building routing approach for indoor environment," in *Proceedings of the International Conference on Computational Science and Its Applications*, pp. 247–260, Springer, Cham, Switzerland, 2017.

[13] S. Alamri, D. Taniar, and M. Safar, "A taxonomy for moving object queries in spatial databases," *Future Generation Computer Systems*, vol. 37, pp. 232–242, 2014.

[14] K. Raptopoulou, A. N. Papadopoulos, and Y. Manolopoulos, "Fast nearest-neighbor query processing in moving-object databases," *Geoinformatica*, vol. 7, no. 2, pp. 113–137, 2003.

[15] S. Alamri, D. Taniar, M. Safarb, and H. Al-Khalidi, "Tracking moving objects using topographical indexing," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 8, pp. 1951–1965, 2015.

[16] Q. Zhang, L. T. Yang, X. Liu, Z. Chen, and P. Li, "A tucker deep computation model for mobile multimedia feature learning," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 13, no. 3s, pp. 1–18, 2017.

[17] J.-W Chang, J.-H. Um, and W.-C. LeeP, "A new trajectory indexing scheme for moving objects on road networks," *Flexible and Efficient Information Handling,* vol. 4042 of Lecture Notes in Computer Science, pp. 291–294, Springer, Berlin, Germany, 2006.

[18] D. Wu, B. Choi, J. Xu, and C. S. Jensen, "Authentication of moving top-k spatial keyword queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 922–935, 2015.

[19] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexing the positions of continuously moving objects," *ACM SIGMOD–SIGMOD Record*, vol. 29, no. 2, pp. 331–342, 2000.

[20] Y. Tao, D. Papadias, and J. Sun, "The TPR∗-tree: an optimized spatio-temporal access method for predictive queries," in *Proceedings 2003 VLDB Conference*, pp. 790–801, Berlin, Germany, September 2003.

[21] E. Frentzos, "Indexing objects moving on fixed networks," in *Proceedings of the 8th International Symposium on Spatial and Temporal Databases (SSTD)*, pp. 289–305, Springer, Santorini Island, Greece, July 2003.

[22] S. Alamri, D. Taniar, and M. Safar, "Indexing moving objects for directions and velocities queries," *Information Systems Frontiers*, vol. 15, no. 2, pp. 235–248, 2012.

[23] C. S. Jensen, D. Lin, B. Chin Ooi, and R. Zhang, "Effective density queries on continuously moving objects," in *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, p. 71, IEEE Computer Society, Atlanta, GA, USA, April 2006.

[24] S. Alamri, D. Taniar, and K. Nguyen, "Efficient cell-based indexing of indoor mobile objects," *Information Systems Frontiers*, 2017.

[25] H. Lu, X. Cao, and C. S. Jensen, "A foundation for efficient indoor distance-aware query processing," in *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering (ICDE)*, pp. 438–449, IEEE Computer Society, Washington, DC, USA, 2012.