A Semantic-Based Indexing for Indoor Moving Objects

Tingting Ben, Xiaolin Qin, and Ning Wang

College of Computer Science & Technology, Nanjing University of Aeronautics & Astronautics, Nanjing 210016, China

Correspondence should be addressed to Xiaolin Qin; qinxcs@nuaa.edu.cn

Received 19 March 2014; Revised 3 May 2014; Accepted 17 May 2014; Published 25 June 2014

Academic Editor: Zhangbing Zhou

Copyright © 2014 Tingting Ben et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The increasing availability of indoor positioning, driven by techniques like RFID, Bluetooth, and smart phones, enables a variety of indoor location-based services (LBSs). Efficient queries based on semantic-constraint in indoor spaces play an important role in supporting and boosting LBSs. However, the existing indoor index techniques cannot support these semantic constraints-based queries. To solve this problem, this paper addresses the challenge of indexing moving objects in indoor spaces, in which both the moving objects and the indoor environments include semantic meanings. We present an indoor semantic-based model, which gives the formal description of semantics of indoor cells and moving objects. Then, a new semantic-based index is proposed for indoor environment, which can support queries under semantic constraint. In addition, we develop efficient algorithms for two new queries, which are the nearest neighbor query based on semantic constraints for static indoor cells and moving objects, respectively. The conducted experiment demonstrates that the proposed index structure is effective, robust, and efficient.

1. Introduction

With the development of sensing technology, more and more mobile devices, such as RFID, Wi-Fi, and smart phones, possess functions of collecting sensory data in indoor spaces. As large parts of people’s daily life are accommodated in various indoor environments, such as office buildings, shopping malls and transportation facilities, data management of indoor moving objects has become a hot research topic with a lot of LBSs emerged in recent years. With further research on this work, more and more methods have been proposed to solve the problem.

Nowadays, as the indoor environments become more complex, there have been more and more semantic-constraint related applications. For example, in an office building, a department manager wants to query a meeting room which meets his requirements, he can query from the servers by sending all his requirements, like the capacity, the distance to him, the hardware configuration, or a combination of several requirements. Similarly, a manager with an urgent task may want to find a suitable employee in marketing department who is as close as possible to him. For another situation, one customer in a large shopping mall needs to know the path to the appropriate clothing shop, which may have an attractive discount. Most indoor situations reflect a significant fact that semantic constraints are common problems existing in LBS queries. However, previous studies involving indoor index techniques cannot solve this problem. They focused on spatial and temporal range queries, trajectory queries, and distance-aware queries, without considering semantic constraints based queries. Further study is still necessary and essential.

Compared with previous studies, research about semantics in indoor scenes will face new challenges and difficulties. Firstly, previous processing technologies [1–3] of indoor moving objects treated room, corridor, stairway, and other indoor entities as a same cell and interpreted each of the moving objects as a point or a region. All of them failed to notice the differences between them. Thus, these previous results were inconclusive. Secondly, the semantic meanings of the cells might affect their connected properties to a certain degree. For example, when a meeting room was being used, irrelevant persons would not be allowed to enter, even though the door was open. Thirdly, the existing query types merely focused on location, object, and time, while ignoring the semantic meanings of the cells and moving objects.

In this paper, we extend the existing indoor index techniques to an innovative one supporting semantics. We
put forward a semantic-based indoor index for the first time. Firstly, we introduce an indoor semantic-based model, which formally defines the semantic information of indoor cells and moving objects. Secondly, we propose a new semantic-based index structure for indoor scenes, called SI (semantic-based indoor Index), which composes of semantic layer, object layer, and topological layer. Thirdly, we define the nearest neighbor queries by semantic constraints and design efficient query processing algorithms that exploit SI. Finally, experiments are conducted compared with the existing indoor index techniques in order to testify the effectiveness of our proposed method. We study the update and query performances of them and test the effects of the number of moving objects and semantic constraints on query performance. The experimental results show that SI is an effective, efficient, and robust index for semantic constraints based queries in indoor spaces.

The rest of this paper is organized as follows. In Section 2, we summarize the related work of indoor moving objects index techniques. A description of the semantic-based model is presented in Section 3. In Section 4, the semantic-based index structure for indoor space is proposed. In Section 5, the nearest neighbor queries by semantic constraints are defined and an efficient query processing algorithms based on SI is designed. Experiments results and analysis are shown in Section 6. Finally, the conclusion is reached in Section 7.

2. Related Work

Previous models of indoor spaces can be mainly classified into two categories: geometric and symbolic [4]. Geometric models are based on geometric coordinate and Euclidean distance. Each object of the geometric models is described by the set of coordinates defined in the Euclidean space. While in the symbolic location model, all objects are represented as symbols and referenced by names. Symbolic space models are often preferred over geometric models in the modeling of indoor spaces because they are able to capture the movement-related semantics associated with indoor entities better [5].

Researching on indoor moving objects often assumes symbolic indoor space modeling and indoor positioning [6]. However, both of them have complementary pros and cons [7]. A semantic-based model for indoor spaces is proposed to support efficient semantic constraints-based queries, which is served as the basis for the data management in this paper.

Over the past decade, many researches have been devoted to moving objects in outdoor environments [8, 9]. However, the achievements of this research are not easily applicable in indoor scenarios [10]. In recent years, with the development of indoor positioning based on technologies such as RFID [11] and Bluetooth [12], some index structures have been proposed for indoor moving objects.

Jensen et al. proposed RTR-tree and TP2R-tree [1] to index offline trajectories of moving objects in symbolic indoor spaces. By differentiating object states in terms of positioning detection, a hash table indexing method [13] has been designed to index the online positions of indoor moving objects. Later, his group proposed techniques for distance-aware queries on indoor moving objects [3, 14]. An index scheme was designed that integrated indoor geometries, indoor topologies, as well as indoor uncertain objects, and, thus, supported indoor distance-aware queries efficiently without time-consuming and volatile distance computation.

ACII [15], an adaptive cell-based index for indoor moving objects, adopted a cell-based index structure instead of outdoor index access methods such as R-trees. It employed an augmented R-tree on time intervals to retrieve relevant historical indoor. The ACII index consisted of two main structures: MC (Memory Cell) and MEMO (Memory). The MC was a cell-based adaptive index structure to handle moving objects searches. The MEMO was a trajectory list in which the time-varying positions of moving objects were stored. This two-body structure used both management of indoor moving objects and overlapping techniques to produce time and space efficient index structures and can process spatiotemporal queries.

Alamri et al. proposed a new indoor index structure [2, 16, 17] for moving objects in indoor spaces that took both spatial and temporal properties into account. The index was based on the connectivity (adjacency) between the indoor environment cells and can effectively respond to the spatial indoor queries and enable efficient updates of the location of a moving object in indoor space. The adjacency or connection between the indoor cells was stored in an adjacency distance lookup table which assisted in the choice of a leaf node algorithm and the grouping of the index structures. The distance was not metric but was based on the number of hops.

However, all these previous works involving indoor moving objects were not concerned with the actual semantic meanings of indoor cells and moving objects. When dealing with semantic constraints-based queries, their efficiency is very low. This is because they need to repeatedly check the semantic information into the database. But such semantic constrains-based queries as the LBSs queries mentioned in Section 1 are widely applied in our real life, which highlight the importance of our study.

This paper differs from these works in several aspects. Conventional index methods for indoor space focus on three dimensions, such as the localization, the object, and the time. In addition to considering these properties, we also pay attention to the semantics. We define the semantic meanings of indoor cells and moving objects. Besides, to the best of our knowledge, we are the first to construct an index structure for the indoor environment based on semantics. What is more, we define the nearest neighbor query with semantic constraints and do experiments to validate the effectiveness of the corresponding query algorithms.

3. Indoor Semantic-Based Model

In this section, we propose a new indoor semantic-based model and give some definitions and notations. “Symbols and Meanings” Section lists all symbols used throughout this paper.
3.1. Preliminaries on Indoor Space. There are numerous indoor scenes, such as office buildings, shopping malls, residential buildings, and airports. The semantic meanings of both indoor cells and moving objects vary with different indoor scenes. Given an indoor scene, we should use semantic meanings to show different rooms. In this paper, we take the office building as an example and give a formal description of the semantic meanings of the scene. The rest of the indoor scenes are similar. The indoor floor plan of a typical office building is shown in Figure 1.

From Figure 1, we can see that the whole indoor space can be divided into four parts: Marketing Department (MD), Human Resources Department (HRD), Treasury Department (TD), and Public Region (PR). Each part consists of many different types of cells, which are office room (OR), meeting room (MR), resting room (RR), washing room (WR), corridor (CR), and hallway (HA).

Figure 2(a) illustrates a detailed diagram of the Marketing Department in Figure 1. For simplicity, to handle some special partitions, such as a hallway, we decompose a long and irregular partition into smaller but regular cells. For a nonconvex partition, we define the points at which the internal angle is greater than 180° as turning points. Then, we decompose the partition into several smaller, convex cells. For a more irregularly shaped partition, we use polygons to approximate it and find the turning points over the polygonal region. Besides, in order to facilitate and describe the problem, the entrances of the smaller cells can be represented by doors located on the edges. Thus, all of the cells are connected by doors and each door is identified by a corresponding $d_i$.

As shown in Figure 2(a), the long corridor is divided into three smaller cells and adds three doors for connectivity. Figure 2(b) shows the doors graph [13] for the floor plan in Figure 2(a). The distance between different doors is the basis of the calculation of the indoor distance. In this paper, Floyd algorithm [18] was used to calculate the shortest distances.

3.2. Indoor Cells. Indoor space can be seen as a collection of cells. Previous studies regarded rooms, corridors, stairways, and other indoor entities as the same cells, which did not distinguish different semantic meanings between them. In order to be more practical, this study takes the semantics into account. Next, we propose a formal description of the cells.
Cell, as the smallest unit of the indoor space, is represented as follows:

\[
\{ (cp,(ct,cr)) \mid ct \in C\text{Type}, cr \in \text{Region} \}, \tag{1}
\]

where \( cp \) represents the physical property set of the cell and \((ct, cr)\) represents the logical property of the cell. The cell's physical property set is represented as follows:

\[
cp = \{ (cn, s, \max, \now, dp) \mid cn \in D_{\text{string}}, s, dp \in \{0,1\}, \max, \now \in D_{\text{int}} \}, \tag{2}
\]

where \( s \) indicates the status of the cell. When \( s = 1 \), it means that the cell is active and objects can access to the cell freely. Conversely, \( s = 0 \) indicates the cell is inactive, which means that objects cannot enter into the cell freely. Besides, \( s \) is set by the actual semantics and properties of the cell. The \( dp \) refers to the hardware facilities equipped with the cell. One means equipment, zero means no equipment. In detail, the hardware facility is printer in OR. In MR, the facility maps to video. In RR, the facility corresponds to free drinks. The remaining parameters are shown in “Symbols and Meanings” Section.

Moreover, \((cr, ct)\) represents the logical property of the cell, which is generated by the concept of division. The cells can be divided into different types. Different indoor scenes correspond to different classification. For example, in office building, the types of cells are represented as follows:

\[
C\text{Type} = \{ \text{OR, MR, RR, WC, CR, HA} \}. \tag{3}
\]

The types should satisfy the condition \( C\text{Type} = \bigcup ct \). Through the concept of division, we divide the whole indoor space into several regions, which corresponds to a set of compact cells. We define a set of symbols to denote the regions of the office building, which meets the requirement

\[
\text{Region} = \bigcup cr, \tag{4}
\]

Furthermore, according to the spatial attributes, the relationships between these parameters should satisfy the constraint conditions as follows, which means that after the semantic classification, the union of all cells is equal to the whole indoor space:

\[
\text{MBR}(cr_i) = \{ c \mid c \cdot cr = cr_i, c \in C \} \quad (i = 1 \sim m),
\]

\[
\text{MBR}(ct_j) = \{ c \mid c \cdot ct = ct_j, c \in C \} \quad (j = 1 \sim n),
\]

\[
C = \bigcup_{i=1}^{m} \text{MBR}(cr_i) = \bigcup_{j=1}^{n} \text{MBR}(ct_j). \tag{5}
\]

3.3. Indoor Moving Objects. Existing proposals [1–3, 14–17] model a moving object by a moving point or an uncertainty region. They only focus on the location of the object, while ignoring the semantics of the object itself.

When considering the semantics, the moving object \( o \) in indoor space is represented as follows:

\[
\{ (op, ((c, t), (x, y))) \mid c \in C, t \in D_{\text{time}}, x, y \in D_{\text{real}} \}, \tag{6}
\]

where \( o \) (moving object) is located in cell \( c \) at the current time \( t \), and its location is \((x, y)\). Besides, \( op \) is the physical property set of the moving object, which is represented as follows:

\[
op = \{ (on, ot, or) \mid on \in D_{\text{string}}, ot \in O\text{Type}, or \in \text{Region} \} , \tag{7}
\]
where $on$ is the name of the moving object and $or$ indicates the region that object $o$ belongs to. And $ot$ denotes the object's type which satisfies the requirement $OType = \bigcup ot$. There are usually five types of objects in office buildings, which are General Manager (GM), Chief Executive Officer (CEO), Branch Manager (BM), Ordinary Employee (OE), and Visitor (V). The set of types is represented as follows:

$$OType = \{GM, CEO, BM, OE, V\}. \quad (8)$$

### 4. Indoor Semantic-Based Index

In this section, we propose an index structure which is based on the indoor semantic-based model in Section 3. The index, which is called SI, consists of three layers, namely, semantic layer, object layer, and topological layer. Section 4.1 details the composite index structure. Section 4.2 briefly discusses index updates.

#### 4.1. Composite Index Structure

To support semantic constraints-based queries, we need to introduce semantics into SI index. The three layers are structured in a hierarchy, in which semantic layer is the base layer. The topological layer can communicate with object layer through semantic layer. For the floor plan shown in Figures 1 and 2(a), its index is shown in Figure 3. In order to facilitate understanding, we give the pointer three different colors. Next, we will introduce the index structure in detail.

#### 4.1.1. Semantic Layer

The semantic layer consists of a tree structure based on the traditional R*-tree [19] to index all indoor partitions. A nonleaf node stores $sem$ and $P$, where $sem$ indicates the semantics from cells' concepts of division and $P$ is an array of several children pointers. The structure of each leaf node entry is in the form of $(cp, Po, and Pr)$. Here, the specific semantic properties are stored in $cp$. $Po$ is a pointer that points to the object layer, while $Pr$ is a pointer that points to the topological layer. Each leaf node stores a cell.

In order to reduce the storage cost of the index, most information of the cells are only stored in leaf nodes. Bottom-up approach is used for constructing this layer. According to the semantics of the cells, we aggregate the cells having the same type together, and then semis equivalent to $ct$. On this basis, we put the cells together based on the principle $cr_i = [c | c \cdot cr = cr_i, c \in Ci]$, which means that they have the same region.

For the same indoor space, the semantic layer structure is stable and easy to expand. Changing the information in object layer has little effect on it. Only when a specific parameter of the cell changes, SI index needs to adjust some properties located in leaf nodes of the semantic layer. This structure effectively reduces the cost of dynamic adjustment of SI, which may be caused by the frequently update of the moving objects.

#### 4.1.2. Object Layer

The object layer stores all indoor moving objects and is associated with the semantic layer through cells at its leaf level. Each leaf node in the semantic layer has a $Po$ pointer, which corresponds to a linked list of buckets. The meta of the bucket is described by the semantic types of the objects. The data for an object in the bucket is stored as a three-tuple: $(o, (c, t), (x, y))$, where it means that object $o$ is at the cell $c$ at time $t$ and its location is $(x, y)$.

Note that in order to accelerate the speed of finding objects, there is a $o$-table, which is structured in the form of $(op, ptr)$. Here, $op$ is the physical properties set of the moving object. And $ptr$ is a pointer that points to the storage location of $o$ in the bucket list. Besides, we only save the object semantics in $o$-table.

#### 4.1.3. Topological Layer

In topological layer, we maintain the connectivity information between indoor cells. Each
leaf node in semantic layer stores a pointer that points to topological layer. For accessibility, the structure is in the form of \((P_i, d_i)\), where \(P_i\) is a pointer that points to the leaf node of semantic layer and \(d_i\) represents the door that connect the two adjacent cells. We store all doors of the cell and the links to accessible cells through each door. Referring to the example shown in Figure 3, for cell \(R_1\), we store \(d_1\) and its adjacent cell's link \(\uparrow R_8\).

The connectivity of a cell is also determined by its own state. When \(s = 0\), all doors of the cell can be seen as closed.

4.2. Update Processing. Algorithm 1 shows the pseudocode of SI's update process. If the object is updated in the same cell space, SI index does not need to update. However, when a moving object is observed in a new cell, the index needs to perform update operation. This technology can improve the efficiency of index updates.

In fact, the moving object can only move to the adjacent cell. According to this rule, we can use the topological layer of SI to improve the efficiency of update; it can exclude the time overhead of searching the cells meeting semantic constraints in semantic layer, which use the top-down method.

The algorithm takes the tuple: \(new = (o, (c, t, (x, y)))\) as an input. First, based on the object's \(o\), the algorithm finds the old location of the object in \(o\)-table (Line (2)-(3)). After finding the old cell where the new object is located, the index checks whether the new cell is identical with the old cell (Line (4)-(6)). If they are not equal, object \(o\) has to be deleted from the bucket list of the old cell. Besides, the old cell also needs to modify its property \(now\) (Line (7)-(8)). Then, the topological layer is acquired to look for the adjacent cells until successful matching with the new cell \(c\) (Line (9)). Next, the new tuple is inserted into the newly cell and the value of the new cell's \(now\) needs to add 1 (Line (10)-(11)). Finally, the \(o\)-table is updated and the \(ptr\) of the updated object is set to the current location (Line (12)).

Algorithm 1: Update.

5. Efficient Query Evaluation

The index we proposed supports indoor queries under semantic constraints. In this section, we take semantics into consideration and study two representative queries in indoor applications: semantic constraints based nearest neighbor query for static indoor cells and moving objects, which are called \(scNNQ-c\) and \(scNNQ-o\), respectively. Note that other types of queries by semantic constraints can also be supported. For example, \(k\)-nearest neighbor queries can be derived.

5.1. Query Semantics

Definition 1 (semantic constraints based nearest neighbor query for static indoor cells (\(scNNQ-c\))). Given a query point \(q \in O\), and a set of semantic constraints \(cs\), the \(scNNQ-c\) returns the cell whose indoor distance to \(q\) is the smallest among all cells meeting the semantic constraints.

Formally, \(scNNQ-c(q,\cdot) = \{c \mid \forall c_i, Dist(p, c) < Dist(p, c_i), c \cdot cs = true, c_i \in C, cs \in CS\}\).

Here, the constraints are derived from the semantic information of the units. Consider:

\[
CS = \{cs_1, cs_2, cs_3 \mid cs_1 \in cr, cs_2 \in ct, cs_3 \in cp\}. \tag{9}
\]

Definition 2 (semantic constraints based nearest neighbor query for moving objects (\(scNNQ-o\))). Given a query point \(q \in O\), and a set of semantic constraints \(os\), the \(scNNQ-o\) returns the object whose indoor distance to \(q\) is the smallest among all objects meeting the semantic constraints.

Formally, \(scNNQ-o(q,\cdot) = \{o \mid \forall o_j, Dist(p, o) \leq Dist(p, o_j), o \cdot os = true, o_j \in O, os \in OS\}\).

Here, the constraints are derived from the semantic information of the objects. Consider:

\[
OS = \{os_1, os_2, os_3 \mid os_1 \in or, os_2 \in ot, os_3 \in op\}. \tag{10}
\]
5.2. Query Processing. We make use of the new index (Section 4) to efficiently evaluate the scNNQ-c and scNNQ-o queries.

When an object wants to find a specific cell, it tends to put forward some related constraint information that the cell needs to be met, such as the type, the region, and the name. In the query processing, we select the appropriate constraints, in turn, until we find the candidate set of the cells which meet all the constraints. This process will enhance the efficiency of the algorithm to a large extent. A large number of cells may be quickly prune so that only the candidate set of cells satisfy the semantic constraints of the query. However, the existing indoor moving object index methods cannot find a candidate set which meets the semantic constraints; they need to check the database repeatedly, which causes them to deal with the low efficiency of such queries. Above all, we can draw the semantic constraints based nearest neighbor query algorithm for static indoor cells, shown in Algorithm 2.

Algorithm 2 presents scNNQ-c query processing algorithm in indoor spaces. The algorithm takes the query point \( q \) and the semantic constraints \( cs \) as input and keeps the result answer and the candidate cell set in \( res \) and \( C \), respectively (Line (2)). Initially, the algorithm gets the query object's cell \( c \) utilizing \( o \)-table (Line (3)). Then, the top-down approach is used to search SI's semantic layer and we check the set of query constraints in order of \( cr, ct \), and \( cp \) (Line (4)–(10)). The set of the candidate cells is composed of the remaining cells left after pruning (Line (11)). Subsequently, in order to return the nearest cell, for each cell in \( C \), the algorithm calculates \([q, c_i]\) using Floyd algorithm (Line (12)–(15)). Lastly, the cell having the shortest distance to \( q \) is assigned to \( result \) (Line (16)–(17)).

Unlike Algorithm 2, Algorithm 3 is to find the object that satisfies the semantic constraints. Here, the semantic constraints may contain the type, the name, or the region that the object belongs to. In the query processing, topological layer of SI is used to quickly search the adjacent cells of the original cell which the query point belongs to. In addition, based on the characteristics of object layer, we can quickly rule out the objects that are not satisfied with the ot property. These operations can improve the efficiency of query processing; the former can reduce the number of cells to be checked and the latter can reduce the number of objects which need to match the semantic constraints. The semantic constraints based nearest neighbor query algorithm for moving objects is shown in Algorithm 3.

In Algorithm 3, scNNQ-o query is processed. After finding the cell \( c \) where the query point \( q \) based on the \( o \)-table, scNNQ-o first checks whether the other objects are also in the same cell. If 

\[
c_i\text{now} > 1
\]

the algorithm checks if the object is satisfied with semantic constraints \( os \) and chooses the object that has the shortest Euclidean distance (Line (1)–(8)). Afterwards, if the result set is still empty, the algorithm searches all the adjacent cells of \( c \) to find the objects that satisfy the constraints. Here, \( d \) stores the shortest distance and \( res \) stores the corresponding object (Line (10)–(13)). Besides, \( d[i] \) is calculated to contain the shortest distance between \( q \) and \( c_i \) (Line (15)). When \( d \) is greater than \( d[i] \), it means that there may be objects that exist in the adjacent cells whose distances are closer than \( d \) (Line (18)). In this case, each connected cell of \( c_i \) needs to go back to step (8) for rechecking (Line (19)–(21)). Otherwise, the \( result \) is returned.

6. Experimental Evaluation

In this section, we evaluate the query and update performance of our proposed method with the existing indoor index techniques, RTR-tree [1] and ACII [15]. In addition, we consider the impacts of the number of the moving objects and semantic constraints on SIs' query performance, respectively. Furthermore, SI, together with RTR-tree and
Input: Query point $q$, the set of query constraints $os$.
Output: Moving object $res$.

1. function $scNNQ_o(object\ q,\ query\ constrains\ os)$
2. ObjectSet result; candidate cell set $O$
3. $c = o$-table.lookup;
4. if ($c.now > 1$) then
5. for each $o \in c$ satisfy $os$ do
6. Calculate $|q, o|_i$;
7. end for
8. result = $\min_{os}|q, o|_i$;
9. else if (result $\neq null$) then
10. for each adjacency cell of $c$
11. for each $o_i$ in $c_i$ satisfy $os$ do
12. Calculate $|q, o_i|_j$;
13. end for
14. $d = \min \{|q, o_i|_j, d\}$, $result = o_j$;
15. end for
16. for ($j = 0; j < i; j++$) do
17. if ($d > d[j]$) then
18. for each adjacency cells of $c_i$ do
19. goto Step 8 to check the adjacency cell;
20. end for
21. else
22. return $result$;
23. end if
24. end for
25. end if
26. end if

Algorithm 3: $scNNQ_o$.

ACII, is implemented in C++ and runs on an Intel Core i3-2120 processor, 3.30 GHz CPU, with 4 GB of RAM running on 32-bit Windows 7 Professional.

In the experiment, we use MWGen generator [20] to product the data sets. The indoor scene used is a real public floor plan [21] of an office building with 35 rooms, 2 staircases, and 1 lift. To test scalability, we use the plan to generate buildings with 8 floors. All of them are connected by hallways and lifts. For the moving objects, we generate a series of data sets, containing 10 K to 100 K objects randomly distributed in the given space. Each object randomly has a life cycle and can randomly choose two rooms as the start and the destination. According to a certain speed, MWGen generates the trajectories of them. What is more, the objects can move within the cell and move to one of the adjacent cells.

All the query points are randomly generated in the given building. In all experiments, we issue 500 queries and report the average response time for each query type. The parameters are summarized in Table 1.

### 6.1. Performance Analysis

Next, we study the update and query processing costs and evaluate the efficiency of algorithms proposed in Section 5.

**Performance of Update Processing.** The results of the execution time for update operations are reported in Figure 4. We can see that SI outperforms the other indexes. As the number of moving objects increases, its performance of update processing is almost the same, reaching a plateau at 0.07 milliseconds. This means that the number of moving objects has no significant influence on update performance. Specifically, our index for indoor spaces is an update-efficient design.

**Performance of Query Processing.** Figure 5 shows the comparison of query performance of three indexes in the setting experiment conditions. From the figure, we see that no matter for $scNNQ_c$ queries or $scNNQ_o$ queries, the running time required for SI is fewer than ACII and RTR-tree under the condition of the same data sets.

For $scNNQ_c$ queries, as reported in Figure 5(a), results of all the index techniques are relatively stable, because the increase in the number of objects does not affect the number of cells needed to check. The performance of SI is high because based on the semantic layer of the index, a large number of cells can be quickly ruled out through semantic constraints. Compared with RTR-tree and ACII,
this operation can reduce the overhead by checking whether the semantics of each cell satisfy the query requirements.

For scNNQ-ο queries, increasing the object number increases the query processing time for each of them. We can draw the conclusion from Figure 5(b) that compared with RTR-tree and ACII, query performance of SI doubled. This is because the existing indoor techniques have no index based on semantics. In order to find the object meeting the semantic constraints of the queries ACII and RTR-tree need to check the database which stores semantics of the moving objects for many times, therefore their query efficiency is low.

From the above analysis, we know that SI has high performance and is very stable. Next, we will analyze the effects of the number of moving objects and semantic constraints on SI’s query performance.

6.2. Impact of Object Number. In this section, we mainly analyze the impacts of the number of moving objects on the query performance. Firstly, we randomly choose the number of semantic constraints for each query. Secondly, we vary the object number from 10 K to 100 K to see the effect on the query performance. For scNNQ-𝑐 queries and scNNQ-ο queries, the impacts of object number on query performance are reported in Figure 6.

The results of scNNQ-𝑐 execution time are reported in Figure 6(a). The query time remains stable as the number of objects increases. That is to say that the number of moving objects is not the key impact factor for scNNQ-𝑐 queries. Because the purpose of scNNQ-𝑐 query is to obtain the cell meeting the query conditions, which is irrelevant to the moving objects in the cells.

Figure 6(b) shows the impacts of the object number on the performance of scNNQ-ο queries. Since the average number of objects in one cell increases, we see that query time increases accordingly. With the average number of moving objects in each cell increasing, the scNNQ-ο queries need to check more objects to confirm the correctness of the query results. What is more, referring to Figure 6, we can see that a greater increase is in the magnitude of the query time for scNNQ-ο queries compared with scNNQ-𝑐 queries.

6.3. Impact of Semantic Constraints Number. Our index and algorithms make full use of semantics. In this section, we investigate how the number of semantic constraints affects the query performance. Figure 7 shows the impact of semantic constraints.

For scNNQ-𝑐 queries, we verify the query performance in four cases, which are (1) the region and type of the cell joint constraint (𝑐𝑡 & 𝑐𝑡), (2) only the type of the cell constraint (𝑐𝑡), (3) only the region of the cell constraint (𝑐𝑟), and (4) no semantic constraints (none). Referring to Figure 7(a), we
can see that the average time needed for scNNQ-c queries is short. The \( cr \) & \( ct \) constraint perform with less cost than other situations. The more the number of semantic constraints, the fewer the number of cells in the candidate sets, and, thus, the smaller the cost for calculating the distance. As the more semantic constraints are considered, the performance of scNNQ-c queries is improved.

For scNNQ-o queries, we verify the query performance also in four cases, which are (1) the region and type of the object joint constraint (\( ot \) & \( or \)), (2) only the type of the object constraint (\( or \)), (3) only the region of the object constraint (\( ot \)), and (4) no semantic constraints (none). The results on the effect of the size of semantic constraints are shown in Figure 7(b). According to Figure 7(b), query time increases as the number of semantic constrain increases. The number of objects and cells having to be inspected is directly proportional to the size of the semantic constraints. When there are no semantic constraints, SI usually only needs to check the objects in the cell where the query point is located in. Besides, the performance of queries with \( ot \) constraint is better than the performance of the queries with \( or \) constraint. This is because that in SI’s object layer the head of the linked bucket is set to \( ot \), which can effectively exclude objects that are not satisfied with \( ot \) condition.

We can summarize the results as follows. The update overhead is very stable, which is not affected by the update frequency. SI outperforms the other indoor index techniques when processing queries based on semantic constraints.
Then, the cost of scNNQ-\(o\) queries increases with the number of moving objects increasing. However, the performance of scNNQ-\(c\) queries is independent of the number of objects. Besides, the performance of both scNNQ-\(c\) and scNNQ-\(o\) queries is affected by the number of semantic constraints. What is more, SI index is a reliable and robust index for indoor spatial and semantic constraints-based queries.

7. Conclusion

In this paper, we research an efficient evaluation of queries based on semantic-constraint in indoor space. We develop an indoor semantic-based model, which defines the semantics of both the cells and the moving objects. We also define two kinds of queries, namely, semantic constraints-based queries for indoor cells (scNNQ-\(c\)) and moving objects (scNNQ-\(o\)). They are the expansion of traditional nearest neighbor query. The most important is that an efficient index structure is proposed for the indoor space, which takes the semantic constraints-based queries for indoor cells and moving objects into account. And efficient query algorithms are proposed for scNNQ-\(c\) and scNNQ-\(o\). Moreover, several experiments are conducted on different data sets to compare our proposals with the existing indoor index techniques and to reveal the effects of two factors on the query performance of SI index that is the number of moving objects and semantic constrains. The experimental results show that our methods are effective and robust. We will combine the access permission of the moving objects with the indoor queries in our future works.

Symbols and Meanings

\(O\): The set of indoor moving objects  
\(C\): The whole indoor space  
\(cp\): Physical properties of the cell  
\(cr\): The region of the cell  
\(s\): The status of the cell  
\(now\): Current object’s number in the cell  
\(ap\): Physical properties of the object  
\(ot\): The type of the moving object  
\((x, y)\): The coordinate of the moving object  
\(a, o\): Indoor moving object  
\(c, c\): Indoor cell (room, corridor, stair…)  
\(ct\): The type of the cell  
\(cn\): The logical name of the cell  
\(max\): Maximum capacity of the cell  
\(dp\): Deploy configuration of the cell  
\(on\): The name of the moving object  
\(or\): The region of the moving object  
\(\text{Dist}(p, q)\): Minimal indoor distance between \(p\) and \(q\).

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61373015, 61300052), the National High Technology Research and Development Program of China (863 Program) (no. 2007AA01Z404), the Research Fund for the Doctoral Program of High Education of China (no. 2010321110017), a project funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD), and the Fundamental Research Funds for the Central Universities, NUAA (no. NP2013307 and no. NZ2013306).

References

[1] C. Jensen, S. H. Lu, and B. Yang, “Indexing the trajectories of moving objects in symbolic indoor space,” in Advances in Spatial and Temporal Databases, vol. 5644 of Lecture Notes in Computer Science, pp. 208–227, Springer, Berlin, Germany, 2009.

[2] S. Alamri, D. Taniar, M. Safar, and H. Al-Khalidi, “Spatiotemporal indexing for moving objects in an indoor cellular space,” Neurocomputing, vol. 122, pp. 70–78, 2013.

[3] X. Xie, H. Lu, and T. B. Pedersen, “Efficient distance-aware query evaluation on indoor moving objects,” in Proceedings of the 29th International Conference on Data Engineering (ICDE '13), pp. 434–445, Brisbane, Australia, April 2013.

[4] U. Leonhardt, Supporting location-awareness in open distributed systems [Ph.D. thesis], Imperial College, 1998.

[5] C. Becker and F. Dürre, “On location models for ubiquitous computing,” Personal and Ubiquitous Computing, vol. 9, no. 1, pp. 20–31, 2005.

[6] C. S. Jensen, H. Lu, and B. Yang, “Graph model based indoor tracking,” in Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware (MDM ’09), pp. 122–131, May 2009.

[7] C. Jiang and P. Steenkiste, “A hybrid location model with a computable location identifier for ubiquitous computing,” in Ubicomp 2002: Ubiquitous Computing, pp. 246–263, Springer, Berlin, Germany, 2002.

[8] R. H. Gütting, M. H. Böhlen, M. Erwig et al., “A foundation for representing and querying moving objects,” ACM Transactions on Database Systems, vol. 25, no. 1, pp. 1–42, 2000.

[9] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, “Moving objects databases: issues and solutions,” in Proceedings of the 10th International Conference on Scientific and Statistical Database Management, pp. 111–122, July 1998.

[10] S. Alamri, D. Taniar, and M. Safar, “Indexing moving objects in indoor cellular space,” in Proceedings of the 15th International Conference on Network-Based Information Systems (NBIS ’12), pp. 38–44, September 2012.

[11] R. Want, “RFID explained: a primer on radio frequency identification technologies,” Synthesis Lectures on Mobile and Pervasive Computing, vol. 1, no. 1, pp. 1–94, 2006.

[12] S. Feldmann, K. Kyamakya, A. Zapter, and Z. Lue, “An indoor Bluetooth-based positioning system: concept, implementation and experimental evaluation,” in Proceedings of the International Conference on Wireless Networks (ICWN ’03), pp. 109–113, June 2003.

[13] B. Yang, H. Lu, and C. S. Jensen, “Probabilistic threshold k nearest neighbor queries over moving objects in symbolic
indoor space,” in Proceedings of the 13th International Conference on Extending Database Technology: Advances in Database Technology (EDBT ’10), pp. 335–346, ACM, March 2010.

[14] H. Lu, X. Cao, and C. S. Jensen, “A foundation for efficient indoor distance-aware query processing,” in Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE ’12), pp. 438–449, Washington, DC, USA, April 2012.

[15] S. S. Shin, G. Kim, and H. Bae, “Adaptive cell-based index for moving objects in indoor,” KSII Transactions on Internet and Information Systems, vol. 6, no. 7, pp. 1815–1830, 2012.

[16] S. Alamri, “Indexing and querying moving objects in indoor spaces,” in Proceedings of the IEEE 29th International Conference on Data Engineering Workshops (ICDEW ’13), pp. 318–321, April 2013.

[17] 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. 18, no. 2, pp. 287–301, 2014.

[18] R. W. Floyd, “Algorithm 97: shortest path,” Communications of the ACM, vol. 5, no. 6, p. 345, 1962.

[19] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, “R∗-an efficient and robust access method for points and rectangles,” in Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 322–331, May 1990.

[20] J. Xu and R. H. Guting, “MWGen: a mini world generator,” in Proceedings of the IEEE 13th International Conference on Mobile Data Management (MDM ’12), pp. 258–267, IEEE, Bengaluru, India, July 2012.

[21] 2011, http://www.modulargenius.com/.
Submit your manuscripts at http://www.hindawi.com