Distance-Aware Join for Indoor Moving Objects

Xie, Xike; Lu, Hua; Pedersen, Torben Bach

Published in:
IEEE Transactions on Knowledge and Data Engineering

DOI (link to publication from Publisher):
10.1109/TKDE.2014.2330834

Creative Commons License
Unspecified

Publication date:
2015

Document Version
Accepted author manuscript, peer reviewed version

Citation for published version (APA):
Xie, X., Lu, H., & Pedersen, T. B. (2015). Distance-Aware Join for Indoor Moving Objects. *IEEE Transactions on Knowledge and Data Engineering*, *27*(2), 428-442. https://doi.org/10.1109/TKDE.2014.2330834

Distance-aware Join For Indoor Moving Objects

Xike Xie, *Member, IEEE,* Hua Lu, *Member, IEEE,* and Torben Bach Pedersen, *Senior Member, IEEE*

Abstract—Indoor spaces accommodate large parts of people's lives. Relevant techniques are thus needed to efficiently manage indoor moving objects, whose positions are detected by technologies, such as Assisted GPS, Wi-Fi, RFID, and Bluetooth. Among such techniques, the distance-aware join processing is of importance in practice for indoor spatial databases. Such join operators leverage a series of applications, such as object tracking, sensor fusion, and clustering. However, joining over indoor moving objects is challenging because: (1) indoor spaces are characterized by many special entities and thus render distance calculation very complex; (2) the limitations of indoor positioning technologies create inherent uncertainties in indoor moving objects data.

In this paper, we study two representative join predicates, *semi-range join* and *semi-neighborhood join*. To implement them, we define and categorize the indoor distances between indoor uncertain objects, and derive different distance bounds that can facilitate the join processing. We design a composite index scheme that integrates indoor geometries, indoor topologies, as well as indoor uncertain objects, and thus supports the join processing efficiently. The results of extensive experimental studies demonstrate that our proposals are efficient and scalable in evaluating distance-aware join over indoor moving objects.

Index Terms—Indoor Space, R-tree, Spatial Join, Moving objects.

1 Introduction

People spend large part of their lives in indoor spaces such as office buildings, shopping malls, conference venues, and transportation facilities, e.g., metro systems and airports. In such indoor spaces, positioning is becoming increasingly available due to different underlying technologies including Assisted GPS (A-GPS), Wi-Fi, RFID and Bluetooth. Indoor positioning provides localization for people and other moving objects in indoor spaces, and thus enables a variety of indoor location-based services (LBSs).

In the emerging indoor applications, distance-aware joins constitute a series of expressive operators that are indispensable for indoor spatial data management.

Example 1: **Indoor Distance-Based Alert.** A kindergarten teacher or a retirement home volunteer would like to be alerted in case that the distance from their supervised children or elders are beyond a distance threshold ϵ. In an airport, upon receiving a report of suspicious criminals or virus carriers, it would be important to monitor their k closest individuals to protect the public safety. In these cases, we need a spatial join on relevant moving objects with an appropriate distance based predicate.

Example 2: **Indoor Sensor Fusion.** In order to provide a reliable location estimation in indoor environments, it is important to combine different and complementary positioning sources like wearable sources (e.g., A-GPS and mobile gyroscope) and stationary sources (e.g., Wi-Fi hotspots and RFID readers). However, all such positioning technologies contains measurement errors. It is beneficial to consider join over moving objects equipped with multiple positioning technologies with accuracy guarantees and thus to derive their overlapping regions [15].

Example 3: **Indoor Data Analysis.** Many algorithms related to similarity search [28] and data mining [5] [4] can be constructed on top of a join query. It can either be a distance similarity join [28] [5] or a k nearest neighbor based join [4]. Thus, for indoor spatial databases, the join operator is an important primitive that allows efficient distance-aware analysis, such as indoor clustering and classification.

1.1 Problem Definition

Given two indoor objects Q and O, let $|Q,O|_I$ denote the indoor distance from Q to O. We formally define the two join predicates studied in this paper as follows.

Definition 1: **Semi-range Join.** Given two sets of indoor objects $\mathbb{Q}$ and $\mathbb{O}$, and a distance threshold ϵ, the semi-range join of the two sets returns all pairs $\{\langle Q,O\rangle\}$ of objects, such that the distance from Q to O are within ϵ. Formally:

$$\mathbb{Q} \underset{\epsilon}{\rhd\!\!<} \mathbb{O} = \{(Q,O) \in \mathbb{Q} \times \mathbb{O} \big| |Q,O|_I \leq \epsilon\} \quad (1)$$

Definition 2: **Semi-neighborhood Join.** Given two sets of indoor objects $\mathbb{Q}$ and $\mathbb{O}$, and an integer k, the neighborhood-join returns all object pairs as follows

$$\mathbb{Q} \underset{k}{\rhd\!\!<} \mathbb{O} = \{(Q,O) \in \mathbb{Q} \times \mathbb{O} \big| O \in kNN(Q)\} \quad (2)$$

Here, kNN(Q) returns Q's k nearest neighbors in $\mathbb{O}$ in terms of the indoor distance.

- *Xike Xie, Hua Lu, and Torben Bach Pedersen are with the Department of Computer Science, Aalborg University, Denmark E-mail: $\{xkxie, luhua, tbp\}$@cs.aau.dk*

In this paper, we study semi-joins instead of full joins, e.g., $\underset{\epsilon}{\bowtie}$ and $\underset{k}{\bowtie}$. The reason, as to be detailed in Section 1.2, is that the indoor space is a quasimetric space where distances are not symmetric. The semi-join is a reflection of the asymmetric property of indoor distances. Therefore, swapping the two arguments of the semi-join will yield different results. Also, full joins can be easily implemented by semi-joins. For example, the full range join can be defined as:

$$\mathbb{Q} \underset{\epsilon}{\bowtie} \mathbb{O} = \{(Q, O) \in \mathbb{Q} \times \mathbb{O} \big| |Q, O|_I \leq \epsilon \wedge |Q, O|_I \leq \epsilon\}$$

We can have $Q \underset{\epsilon}{\bowtie} O = Q \underset{\epsilon}{\mathrel{\triangleright\!\!<}} O \cap O \underset{\epsilon}{\mathrel{\triangleright\!\!<}} Q$.

For ease of presentation, we use joins to refer to semi-joins defined above throughout this paper when the context is clear. We call $\mathbb{Q}$ the *query objects*, and $\mathbb{O}$ the j*target objects*. For both $\underset{\epsilon}{\mathrel{\triangleright\!\!<}}$ and $\underset{k}{\mathrel{\triangleright\!\!<}}$, appropriate handling of distances between indoor objects is of critical importance, which faces several technical challenges in indoor spaces.

1.2 Challenges in Indoor Spaces

First of all, indoor spaces are characterized by entities such as walls, doors, rooms, etc., which render Euclidean distance and spatial network distance unsuitable [22], [33]. Such entities imply topological constraints that enable and/or disable movements. Actually, the indoor space ($\mathbb{I}$) is a *quasimetric* space. Specifically, given two points $p, q \in \mathbb{I}$, the distance $|p, q|_I$ satisfies:

1) $|p, q|_I \geq 0$ (*non-negativity*);
2) $|p, q|_I \neq |q, p|_I$ (*non-symmetry*);
3) $|p, q|_I \leq |p, e|_I + |e, q|_I$ (*triangle inequality, Lemma 10*).

We show a floor plan example in Figure 1. The Euclidean distance between two points p and q does not make sense because it is blocked by a wall. To reach p from q, one has to go through doors d_{13} and d_{15} sequentially to enter room 12. One can not reach room 12 by d_{12} because the door is one-directional, as indicated by the arrow. Note that one-directional doors are often seen in many scenarios, e.g., security controls in airports.

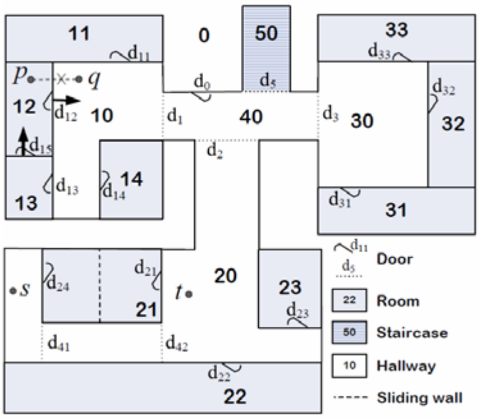


Fig. 1. Floor Plan Example

Second, indoor entities can also be associated with temporal variations. For example, a room may be only temporarily available due to its opening hours, or being blocked in a fire emergency. Also, a large room, e.g., a conference hall, may be partitioned into several smaller rooms to accommodate different events. Such reorganizations can render pre-computed indoor distances [22], [33] volatile. Refer to the example of Figure 1. Room 21 can be a single partition in banquet style if the sliding wall indicated by the dashed line is dismounted. It can also be split into two partitions in meeting style if the sliding wall is mounted. Consequently, point s cannot reach t through room 21, and the distance between s and t needs recalculating by involving doors d_{41} and d_{42}.

Third, the accuracy of indoor positioning is limited, typically varying from a few to about 100 meters [1]. For example, under RFID-based indoor positioning, the location of an object is reported as a region when it is in the detection range of an RFID reader. Due to economic reasons, an indoor space is not fully covered by such readers. As a result, indoor moving objects do not get continuous location updates as their outdoor counterparts do in GPS positioning. Consequently, the location uncertainties in indoor moving objects data make it more complex to calculate object-related indoor distances.

To address these challenges, we need to support indoor distances that take into account topological constraints, temporal variations, and location uncertainties. Recent research [22], [33] only considers part of these important points. In this paper, we propose a complete set of techniques for efficient distance-aware join processing on moving objects in realistic and dynamic indoor spaces.

1.3 Contributions

Overall, our technical contributions in this paper falls into two important aspects.

First, we define the indoor distance between two moving objects Q and O, whose locations are obtained through the aforementioned limited indoor positioning. We choose the *expected distance* as it is both interpretative and semantically comprehensive [9]. By referring to the indoor topology, we divide O's imprecise location into disjoint subregions each falling into one indoor partition (e.g., a room). Subsequently, we classify the distances from Q and the subregions based on the topological properties, and derive various distance bounds that can remove non-qualifying join pairs without calculating detailed expected indoor distances.

Second, we design a composite index for indoor spaces as well as indoor moving objects, as illustrated in Figure 2. The *geometric layer* consists of a tree structure that adapts the R*-tree [3] to index all indoor partitions, as well as a skeleton tier that maintains a small number of distances between staircases. In addition, the *topological layer* maintains the connectivity information between indoor partitions, and it is implicitly integrated into the tree structure through inter-partition links. Last, the *object layer* stores all indoor moving objects and is associated with the tree through partitions at its leaf level. By integrating the distance bounds at corresponding layers, the index supports fast distance based pruning in query evaluation.

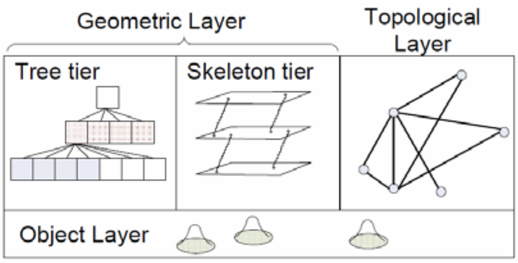


Fig. 2. Composite Index for Indoor Space

This paper substantially extends our previous work [31]. First, we formalize the problem of join operations over indoor objects (Section 1). Second, we define the indoor distance between imprecise objects (Sections 2.2 and 2.3) and extend the upper-/lower bounds from supporting "point-to-object" distance to "object-to-object" distance (Section 3). Third, we propose efficient algorithms for computing the semi-join operators on the previously proposed indoor index (Section 5). Fourth, we conduct extensive experiments to evaluate the new proposals (Section 6).

Remaining sections are organized as follows. Section 2.1 studies preliminaries for indoor databases. Section 4 designs a composite index for indoor spaces and moving objects. Section 7 reviews the related work. Section 8 concludes the paper and discusses future directions.

2 Indoor Distances for Uncertain Objects

In this section, we study indoor distances in detail. Section 2.1 presents preliminaries on indoor space and indoor distance. Section 2.2 defines the expected indoor distance for uncertain moving objects. Section 2.3 discusses categories of indoor distances. Table 1 lists all notations used throughout this paper.

2.1 Preliminaries on Indoor Space and Indoor Distance

Given an indoor space, we use *partitions* to refer to rooms, staircases, or hallways. They are connected by doors or staircase entrances. For simplicity, we regard hallways and staircases as rooms. The two entrances of a staircase can be represented by doors located on the staircase's two ends. Partitions, including their associated doors, are basic elements in indoor spaces. An indoor partition's characteristics lie in two major aspects: geometry and topology. In terms of geometry, they are 3D spatial entities in Euclidean space. Meanwhile, they are aligned to floors inside a building. For topology, partitions are separated by walls etc., and interconnected by doors or staircase entrances.

TABLE 1
Notations

Notation	Meaning
$\mathbb{I}$, $\mathbb{E}$	Indoor space, Euclidean space
$\mathbb{Q}$, $\mathbb{O}$	a set of query objects, a set of target objects
Q, O	a query object, a target object
$\lvert Q, O\rvert_I$	Indoor distance between p and q
$\lvert Q, O\rvert_E$	Euclidean distance between p and q
$\lvert Q, O\rvert_K$	Skeleton distance between p and q
$a.l$ or $a.u$	lower or upper bound of the value a
$\uparrow A$	the link/pointer to the entity A
$[R_i^-, R_i^+]$	the range for R on dimension i
$D(p)$	doors of partition p
$P(d)$	partitions connected to door d
$P(q)$	the partition containing point q
$P(O)$	partitions overlapping with object O
$\lvert O\rvert$	the number of instances belonging to object O
$a \overset{*d}{\rightsquigarrow} b$	a path from a to b with d as the last door
$a \overset{*}{\to} b$	the shortest path from a to b
$\odot(c, r)$	a circle centered at c with radius r
$\underset{\epsilon}{\ltimes}$	semi-range join
$\underset{k}{\ltimes}$	semi-neighborhood join
$\oplus$	Minkovski sum

The *doors graph* [33] has been proposed to represent the connectivity of indoor partitions as well as door-to-door distances. Formally, the doors graph is defined as a weighted graph $G_d = \langle D, E\rangle$, where:

(1) D is the set of vertices, each corresponding to a door.
(2) E is the set of edges. An edge (d_i, d_j) exists if these two doors are associated with the same partition.
(3) Each edge (d_i, d_j) has a weight that is the distance from door d_i to door d_j through their common partition.[1]

Specifically, if a door is unidirectional, i.e., allowing one-way movement only, its graph vertex's associated edges acquire directionality accordingly and are in- or out-edges. If an edge does not involve unidirectional doors, the edge is bidirectional. More details can be found in the previous work [33].

Figure 3(a) is the doors graph for the floor plan in Figure 1. One-way door d_{12}'s adjacent edges, e.g., (d_{15}, d_{12}) and (d_{12}, d_{11}), are unidirectional in the doors graph, whereas other edges that do not involve doors d_{12} or d_{15} are bidirectional, as amplified in Figure 3 (b).

In this paper, we do not create a separate doors graph. Instead, in our composite index for indoor space, we add extra links to the leaf-level tree nodes if their corresponding partitions are connected by a door. This design yields a de facto doors graph that is integrated in the index. More details are to be presented in Section 4.

Unlike the previous works [22], [33], we do not pre-compute and store the shortest indoor distances for all door pairs before query processing. Pre-computing all such distances is expensive especially when a given

[1] The door midpoints are used for calculating door-related distances.

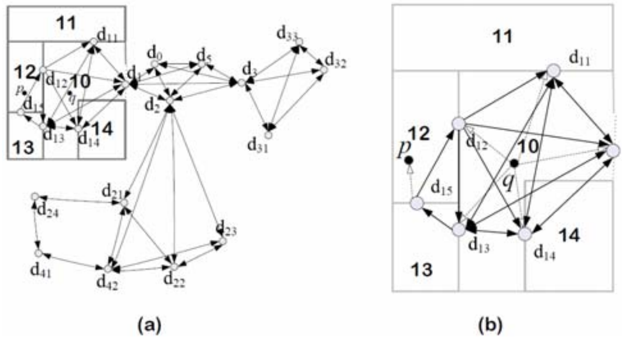


Fig. 3. Example of Doors Graph

indoor space has many partitions and doors. On the other hand, our decision is also justified by the temporal indoor space variations we consider in this paper. As explained in Section 1, partitions can be split or merged. Partitions can also be blocked in emergence or booked by sudden events, thus some doors are closed and/or temporary doors are opened accordingly. Such changes inevitably invalidate the indoor distance computing, and a considerable part of the shortest indoor distances can be affected if the temporal change happens on a pivot door or partition.

Given two indoor positions p and q, we use $q \overset{\delta}{\rightsquigarrow} p$ to denote a path from q to p where δ is the sequence of doors on that path. Referring to Figure 3(b), a path from q to p is $q \overset{d_{13},d_{15}}{\rightsquigarrow} p$ that means one can reach p from q through door d_{13} followed by d_{15}. We call the length of the shortest path as the *indoor distance* from q to p, and denote it as $|q,p|_I$. Formally, $|q,p|_I = min_{\delta}(|q \overset{\delta}{\rightsquigarrow} p\ |)$. In the example, $q \overset{d_{13},d_{15}}{\rightsquigarrow} p$ is also the shortest path as it is the only possible path. We use $q \overset{\delta}{\rightarrow} p$ to denote the shortest path from q to p.

Indoor distance $|q,p|_I$ consists of two parts: door-door distance and intra-partition object-door distance. In Figure 3(b), door-door paths (e.g., $d_{13} \rightarrow d_{15}$) are represented by solid arrows; object-door paths (e.g., $d_{15} \rightarrow p$) are represented by dashed arrows. Let $D(p)$ be the set of doors of p's partition. In general, the indoor distance $|q,p|_I =$

$$min_{d_q \in D(q), d_p \in D(p)}(|q,d_q|_E + |d_q,d_p|_I + |d_p,p|_E) \quad (3)$$

Previous works [22], [33] assume that all possible $|d_p,d_q|_I$s are known beforehand. In this paper, we lift this assumption and investigate how to process queries without pre-computing $|d_q,d_p|_I$s. As a remark, strictly speaking, $|q,d_q|_E$ should consider the possible obstacles in q's partition. Our proposals in this paper can incorporate such obstructed distances [34] at a low level for indoor partitions. As this is not the focus of this paper, we omit the details. Reversely, the concept and the computation of obstructed distances are insufficient for modeling complex indoor topologies and distances.

2.2 Indoor Moving Objects and Expected Indoor Distance

Existing proposals [10], [26] model a moving object by an *uncertainty region*, where the exact location is considered as a random variable inside. The possibility of its appearance can be collected by objects' velocities [33], parameters of positioning devices [10], or analysis of historical records and thus represented by a *probability density function* (**pdf**). The pdf can be described by either a closed form equation [7], [8], or a set of discrete instances [16], [19]. In this paper, we adopt the instance representation, as it is general for arbitrary distributions. Thus, an indoor moving object O is represented by a set $\{(o, o.\rho)\}$, where o is an instance and $o.\rho$ is its *existential probability*, satisfying $\sum_{o \in O} o.\rho = 1$. Based on such probabilities, we define the expected indoor distance to measure the distance from one object to the other.

Definition 3: **(Expected Indoor Distance for Uncertain Object)** Given two uncertain objects Q and O, the indoor distance between them is:

$$|Q,O|_I = E_{q \in Q, o \in O}(|q,o|) = \sum_{q \in Q} \sum_{o \in O} |q,o|_I \cdot q.\rho \cdot o.\rho$$

In an indoor setting, an object O's uncertainty region may overlap with multiple partitions. An example is shown in Figure 5. Object O's uncertainty region overlaps with three different rooms. Accordingly, all the instances in O are divided into subsets. Generally speaking, we have $O = \cup_{1 \le j \le m} O[j]$ $(1 \le m \le |O|)$ where each $O[j]$ corresponds to a different partition and contains all those instances in that particular partition. We also call such an $O[j]$ as O's *uncertainty subregion*. We proceed to study all cases on all possible $|Q,O|_I$s.

2.3 Cases of Indoor Distance $|Q,O|_I$

We consider how many uncertainty subregions, i.e., $Q[i]$s or $O[j]$s, objects Q or O have, and how many indoor paths exist from $Q[i]$ to $O[j]$. Accordingly, there are three cases for $|Q,O|_I$.

2.3.1 Single-Partition Single-Path Distance

In this case, Q's uncertainty regions fall into a single partition, and so does O's. Let P_Q be the partition containing Q, and P_O be the one containing O. For an arbitrary pair $(q,o)_{q \in Q, o \in O}$, the shortest path $q \overset{d_Q * d_O}{\rightarrow} o$ shares the same door sequence starting with d_Q and ending with d_O, through which the path reaches o from q. As a result, we calculate the indoor distance as follows

$$\begin{aligned} &|Q,O|_I = \\ &\sum_{q \in Q} \sum_{o \in O} (|q,d_Q|_E + |d_Q,d_O|_I + |d_O,o|_E) \cdot q.\rho \cdot o.\rho \\ &= \sum_{q \in Q} |q,d_Q|_E + |d_Q,d_O|_I + \sum_{o \in O} |d_O,o|_E \quad (4) \end{aligned}$$

2.3.2 *Single-Partition Multi-Path Distance*

In this case, Q and O's uncertainty regions still falls into one single partition, respectively. However, for different instances, say o_i and o_j, shortest paths $q \xrightarrow{*} o_i$ and $q \xrightarrow{*} o_j$ do not share the same door sequence. Here, q is an instance of Q. As a result, the indoor distance is calculated as follows.

$$|Q, O|_I = \sum_{q \in Q} \sum_{o \in O} |q, o|_I \cdot q.\rho \cdot o.\rho \tag{5}$$

An example of this case is shown in Figure 4, where O has two instances o_1 and o_2. The shortest path from q to them are: $q \overset{d_3,d_1}{\rightsquigarrow} o_1$ and $q \overset{d_2}{\rightsquigarrow} o_2$.

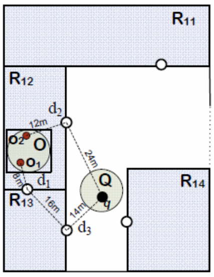


Fig. 4. Single-Partition Multiple-Path Distance

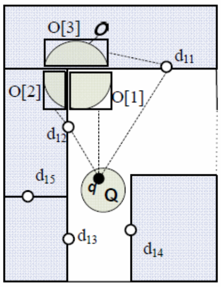


Fig. 5. Multi-Partition Path Distance

2.3.3 *Multi-partition Path Distances*

In this case, either object Q or O's uncertainty region overlaps with more than one partitions, and thus $O = \cup_{1 \le j \le m} O[j]$ $(1 < m \le |O|)$. We calculate the indoor distance as follows

$$|Q, O|_I = \sum_i \sum_j (|Q[i], O[j]|_I \cdot \sum_{q \in Q[i]} q.\rho \cdot \sum_{o \in O[j]} o.\rho) \tag{6}$$

In the above equation, $|Q[i], O[j]|_I$ is calculated according to either Equation 4 or 5.

An example of this case is shown in Figure 5, where object O has three uncertainty subregions $O[1]$, $O[2]$ and $O[3]$. Accordingly, we have $|Q, O|_I = E(\sum_{1 \le j \le 3}(|q, O[j]|_I)$.

In summary, to calculate the indoor distance $|Q, O|_I$, we need to find shortest paths for every instance pair of Q and O. Suppose objects Q and O contain $|Q|$ and $|O|$ instances, respectively. According to Definition 3, $|Q, O|_I$ requires $O(|Q| \cdot |O|)$ shortest path calculation. Next, we derive effective upper and lower bounds to alleviate the extensive computation.

3 Upper-/lower Bounds for Indoor Distances

In this section, we derive the upper and lower bounds (*ULBounds in short*) of $|Q, O|_I$ for each of the layers mentioned in Section 1 (see Figure 2 also). Specifically, they are *Euclidean Lower Bounds* for the geometric layer, *Topological Layer ULBounds* for the topological layer, and *Object Layer ULBounds* for the object layer where location probabilities of uncertain objects are known.

3.1 Geometric Layer Lower Bounds

For two uncertain objects Q and O in an indoor space, the (virtual) Euclidean distance between them is the lower bound of their distance in the indoor space. Therefore, we have $|Q, O|_{minE} \le |Q, O|_{minI}$, where $|Q, O|_{minE} = min_{q \in Q, o \in O} |q, o|_E$.

In the Euclidean space, the uncertainty region is a connected region. We bound an indoor object Q's instances by a circle $\odot(c_Q, r_Q)$, centered at the *centroid* c_Q, and with the *radius* r_Q which is defined as the maximum distance between the centroid and samples.

Lemma 1: Given an indoor object Q, denoted by $\odot(c_Q, r_Q)$, and another object O, denoted by $\odot(c_O, r_O)$, the geometric lower bound property can be rewritten as:

$$|c_Q, c_O|_E - r_Q - r_O \le |Q, O|_{minI} \tag{7}$$

Note that it is impossible to derive the indoor upper bounds by using Euclidean distances only. However, indoor distances can be upper bounded by a mixture of Euclidean distances and topological constraints.

3.2 Topological Layer ULBounds

For two uncertain objects $Q = \cup_{i=1}^{m} Q[i]$ and $O = \cup_{j=1}^{n} O[j]$, suppose that $P(Q[i])$ is the partition containing subregion $Q[i]$, and $P(Q)$ are the partitions overlapping with Q.

Lemma 2: **(Topological LBound)** Let $t_{min}(Q[i], O[j])$ be:

$$\underset{d_q \in D(P(Q[i])), d_s \in D(P(O[j]))}{min} |Q[i], d_q|_{minE} + |d_q \xrightarrow{*} d_s| + |d_s, O[j]|_{minE} \tag{8}$$

Then, $|Q, O|_I \ge \underset{i,j}{min}\{t_{min}(Q[i], O[j])\}$.

Lemma 3: **(Topological UBound)** Let $t_{max}(Q[i], O[j])$ be:

$$\underset{d_q \in D(P(Q[i])), d_s \in D(P(O[j]))}{min} |Q[i], d_q|_{maxE} + |d_q \xrightarrow{*} d_s| + |d_s, O[j]|_{maxE} \tag{9}$$

Then, $|Q, O|_I \ge \underset{i,j}{min}\{t_{max}(Q[i], O[j])\}$.

Suppose Q and O overlap with m and n partitions, respectively. Lemmas 2 and 3 involve O(mn) shortest paths. However, if Q and O's uncertainty regions both overlap with one partition, the above two lemmas can be rewritten as Lemma 4:

Lemma 4: Given an indoor object Q, denoted by $\odot(c_Q, r_Q)$, and another object O, denoted by $\odot(c_O, r_O)$, the topological ULBounds can be rewritten as:

$$|c_Q, c_O|_I - r_Q - r_O \le |Q, O|_I \le |c_Q, c_O|_I + r_Q + r_O \tag{10}$$

Proof:

$$\begin{aligned} |Q, O|_{minI} &= min_{\forall q \in Q}(|q, O|_I) \\ &\ge min_{\forall q \in Q}(|q, c_O|_I - r_O) \quad \text{(Lemma 10)} \\ = min_{\forall q \in Q}(|q, c_O|_I) - r_O &\ge |c_Q, c_O|_I - r_Q - r_O \quad \text{(Lemma 11)} \\ &\Rightarrow |Q, O|_{minI} \ge |c_Q, c_O|_I - r_Q - r_O \\ &\Rightarrow |Q, O|_I \ge |c_Q, c_O|_I - r_Q - r_O \end{aligned}$$

$|Q, O|_I \leq |c_Q, c_O|_I + r_Q + r_O$ can be proved likewise. So the lemma is proved. □

According to Lemmas 2 and 3, it requires shortest path (e.g., $|d_q \overset{*}{\to} d_s|$) computation on the doors graph in order to derive the *ULBounds*. To reduce the computation, we design a looser topological upper bound. It is not as tight as *Topological UBound*, but it is more economic to be derived. Instead of getting the shortest paths, it only requires some paths that connect objects Q and O. We call it *Topological Looser UBound*.

Lemma 5: **(Topological Looser UBound, TLU)** Let $t_{max}(Q[i], O[j])$ be:

$$\min_{d_q \in D(P(Q[i])), d_s \in D(P(O[j]))} |Q[i], d_q|_{maxE} + |d_q \overset{*}{\rightsquigarrow} d_s| + |d_s, O[j]|_{maxE} \quad (11)$$

Then, $|Q, O|_I \leq \max_{i,j}\{t_{max}(Q[i], O[j])\}$.

As to be detailed in Section 5, we use the looser bounds to prune doors and partitions in query processing. Afterwards, the shortest paths are only evaluated on the remaining doors and partitions for the topological ULBounds. For the case that both Q and O overlap with one partition, Lemma 5 can be simplified as:

$$|Q, O|_I \leq \min_{d_q \in D(P(Q[i])), d_s \in D(P(O[j]))} |d_q \overset{*}{\rightsquigarrow} d_s| + |d_q, c_Q|_E + |d_O, c_O|_E + r_Q + r_O \quad (12)$$

In contrast to TLU (Lemma 5) and topological ULBounds (Lemmas 2 and 3) which consume $O(mn)$ in terms of the number of shortest paths, their simplified versions (i.e., Lemma 4 and Equation 12) only take one shortest path to derive the ULBounds, which is certainly more efficient. To generalize the single-partition case to multiple-partition scenarios, we define the *star-connected region*.

Definition 4: (**star-connected regions**) Let $O = \odot(c, r)$ be an indoor object overlapping with more than one partition, i.e., $O = \cup_{i=1}^{n} O[i]$. Let the subregion containing c be the *central region* C. If all other subregions are connected to C by doors, we call O's region a *star-connected region*. Formally,

$$\forall O[i] \neq C, \exists \text{door } d, \text{ such that } d \in C \text{ and } d \in O[i].$$

Notice that a star-connected region is a connected region in both Euclidean space and indoor space. Then, we can define O by $\odot(c, r_I)$, where r_I is the maximum indoor distance from centroid c to all subregions, $r_I = \max_i |c, O[i]|_{maxI}$. By defining star-connected regions, we can benefit from the simplifications in topological ULBounds by substituting r_I into Equations 10 or 12. In summary, a star-connected region is a special case in that it is a tight ($r_I \leq 3r$) and connected region in the indoor space.

3.3 Object Layer ULBounds

Nevertheless, the topological ULBounds can still be very loose in the following two cases:

- Case 1: If object Q or O's uncertainty region is relatively big compared to their indoor distance;
- Case 2: If object Q or O overlaps with multiple partitions that are not interconnected (i.e., they are not star-connected regions).

Referring to the example shown in Figure 5, where object $O = \cup_{i=1}^{3} O[i]$, the distance from q to $O[1]$ is short, while the distance to $O[3]$ is long. If the gap between topological upper and lower bounds is large, the expected distance is only constrained by a loose range and thus not well approximated. Notice that the gap can be even bigger, if q is also an uncertain object in Figure 5. To tackle this problem, we design *Object Layer ULBounds* by using location probability information associated with objects.

For two indoor objects, their expected indoor distance derives the expectation of the distances between all pairs of samples. Geometric and topological ULBounds bound the distance by the minimum/maximum distance between sample pairs. The object layer ULBounds make a difference by considering the probability distributions among sample points. We proceed to define the concept of β-region.

Definition 5: **(β-region [6], [20], [21])** Given an indoor object O, the β-region is a closed region such that the probability of O being located inside the region is greater than β.

The β-region can be constructed in different manners:

- For Case 1: Given a predefined β value, the β-region can be constructed by first sorting an object's samples according to their distances from the centroid. Then, we count and summarize their probabilities until β is reached. The distance between the last counted sample point and the centroid is r^{β}. Thus, O's β-region O^{β} is determined by a circle $\odot(c, r^{\beta})$.
- For Case 2: We randomly select a subregion $O[i]$ as the β-region. Here, the value of β equals to the summation of probabilities for samples inside $O[i]$, i.e., $\beta = \sum_{s \in O[i]} s.\rho$. The shape of the β-region is a rectangle, which is the intersection of O's MBR and the partition containing $O[i]$.

In both cases, we bound the possible instances of an object by a region with the probability higher than β. In the remainder of this section, we first consider a simplified case where the distance is between a point q to an object O (Lemma 6). Based on that, we extend point q to object Q and derive the object layer ULBounds (Lemma 7).

Lemma 6: Given a point q and an object O, we have:

$$(1-\beta) \cdot |q, O|_{minI} + \beta \cdot |q, O^{\beta}|_{minI} \leq |q, O|_I \leq \beta \cdot |q, O^{\beta}|_{maxI} + (1-\beta) \cdot |q, O|_{maxI}$$

Proof: We first prove $|q, O|_I \leq \beta \cdot |q, O^{\beta}|_{maxI} + (1-\beta) \cdot |q, O|_{maxI}$.

$$\begin{gathered}|q,O|_I = E(|q,O|_I) = \\ E_{s\in O^\beta}(|q,s|_I)\cdot Pr\{s\in O^\beta\} + E_{s\in O-O^\beta}(|q,s|_I)\cdot Pr\{s\in O-O^\beta\} \\ E_{s\in O^\beta}(|q,s|_I)\cdot \beta + E_{s\in O-O^\beta}(|q,s|_I)\cdot(1-\beta) \\ \le |q,O^\beta|_{maxI}\cdot\beta + |q,O|_{maxI}\cdot(1-\beta) \\ (\forall s\in O^\beta, |q,s|_I \le |q,O^\beta|_{maxI})\end{gathered}$$

Likewise, we can prove $(1-\beta)\cdot|q,O|_{minI} + \beta\cdot|q,O^\beta|_{minI} \le |q,O|_I$. Thus, the lemma is proved. □

Lemma 7: (**Object Layer ULBounds**) Given two objects O_i and O_j, we have:

$$\begin{gathered}(1-\beta_Q)\beta_O|Q^\beta,O|_{minI} + \beta_Q\beta_O|Q^\beta,O^\beta|_{minI} + \\ (1-\beta_Q)(1-\beta_O)|Q,O|_{minI} + \beta_Q(1-\beta_O)|Q,O^\beta|_{minI} \\ \le |Q,O|_I \\ \le \beta_Q\beta_O|Q^\beta,O^\beta|_{maxI} + (1-\beta_Q)\beta_O|Q^\beta,O|_{maxI} + \\ \beta_Q(1-\beta_O)|Q,O^\beta|_{maxI} + (1-\beta_Q)(1-\beta_O)|Q,O|_{maxI}\end{gathered}$$

Proof: Assume q is a point inside Q, we have:

$$\begin{gathered}E(|Q,O|) = \\ E(|q,O|_I|q\in Q^\beta)\cdot Pr\{q\in Q^\beta\} \\ +E(|q,O|_I|q\in Q-Q^\beta)\cdot Pr\{q\notin Q^\beta\}\end{gathered} \tag{13}$$

$$|q,O|_I = \begin{cases}|Q^\beta,O|_I & q\in Q^\beta \\ |Q,O|_I & q\notin Q^\beta\end{cases} \tag{14}$$

Substituting Equations 13 and 14 into Lemma 7, the lemma is proved. □

3.4 Summary

To summarize, we use geometric and topological ULBounds for the case that an object overlaps with a single partition; we use probabilistic ULBounds for the case that an object overlaps with multiple partitions. A summary is given in Table 2. With the ULBounds as well as the approximate indoor distances, we avoid computing shortest paths for all existential instances of an uncertain object. However, we still need to find shortest paths for other objects and instances when using these bounds. To accelerate such shortest path computing, we design a composite index scheme to enable search space pruning.

TABLE 2
Indoor Upper / Lower Bounds for Different Cases

Cases	Bounds
single-partitioned region star-connected region	Geometric Layer ULBounds (Lemma 8) Topological Layer ULBounds (Lemma 5)
multi-partitioned region big uncertainty region	Object Layer ULBounds (Lemma 7)

4 Composite Index for Indoor Spaces

Our composite index consists of three layers, namely *Geometric layer*, *Topological layer*, and *Object layer*. The geometric layer consists of tree tier and skeleton tier. Section 4.1 details the composite index structure. Section 4.2 presents the *Geometric Lower Bound* property which is useful in the query phase.

4.1 Composite Index Structure

4.1.1 Overview

For the floor plan shown in Figure 1, its composite index is shown in Figure 6(c). Figure 6(a) is a planar view of the index and Figure 6(b) is an amplified view of the floor plan part covered by tree node R_1.

Indoor partitions are indexed by the *Tree Tier*, called *ind*R-tree, that adapts an R-tree and treats the floor plan as an Euclidean space. Large partitions may be decomposed into small ones, each of which corresponds to a leaf node entry. Each leaf node, which represents a (sub)partition, is associated with a bucket of objects in that partition. The set of all object buckets form the *Object Layer*. This way, the object can be easily located to one or more indoor partitions given its positioning information (either a location or an uncertainty region) via the tree. Since the Euclidean distance is a lower bound of the indoor distance, the hierarchical tree structure supports indoor distance-aware queries efficiently, by pruning away non-qualifying candidates at higher levels.

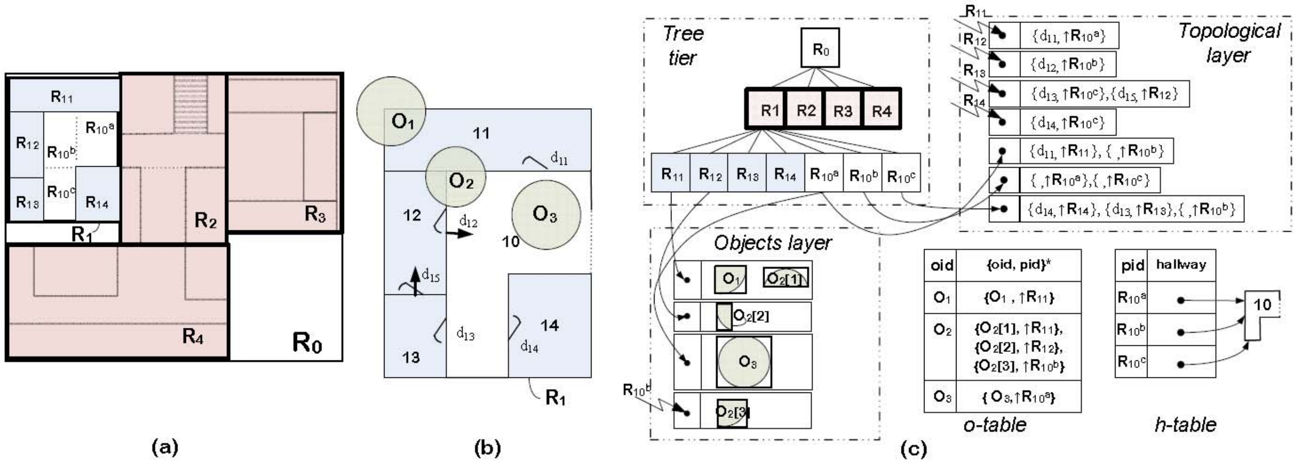
The indoor topology information is covered by the *Topological Layer*. To support indoor distance calculation, especially for the door-to-door distance, we can traverse the topological layer in the way of traversing a graph. In addition, the *o-table* maps an object to the tree leaf nodes it overlaps with, while the *h-table* stores the mappings from a leaf node entry to an indoor partition to which it belongs.

4.1.2 Tree Tier

Indoor partitions like rooms and hallways are special spatial entities. They occupy 3D regions, spanning two horizontal dimensions and one vertical dimension. Considering a building consisting of many floors, the closest facility (e.g., a restroom) might be the one upstairs. Therefore, the distance of the vertical dimension should be considered.

On the other hand, for the entities on the same floor, we care more about their planar distances. If a partition is represented by a 3D *Minimum Bounding Rectangle* (MBR in short) in *ind*R-tree, the maximum 3D distance will surely surpass its planar counterpart. This would degrade the tree's pruning performance while handling queries.

However, if the MBRs are planar rectangles, the splitting strategy for R-tree fails as the 3D volume of a tree node, expected to be minimized in R-tree construction, is always 0. To this end, when creating the tree we set the vertical length for one partition to 1 centimeter, which is very small compared to its horizontal length. Let the vertical dimension be the third dimension. We set an MBR R's vertical range to be $[R_3^-, R_3^+]$, where R_3^+ is larger than R_3^- by 1 centimeter. In the query phase, while calculating distances, we consider R's vertical range to be $[R_3^-, R_3^-]$, i.e., the vertical length is neglected. In other words, the partition is treated as a 2D rectangle distributed in the 3D space in the query phase. This design gives two advantages: 1) it reduces the distance



oid	{oid, pid}*
O1	{O1 , ↑R11}
O2	{O2[1], ↑R11}, {O2[2], ↑R12}, {O2[3], ↑R10b}
O3	{ O3,↑R10a}

pid	hallway
R10a	●
R10b	●
R10c	●

Fig. 6. An Example of the Composite Indoor Index (without the skeleton tier)

calculation workload; 2) it makes the distance reflected in the tree more accurate without the disturbance from the vertical dimension.

Some special partitions, such as a hallway, may be very imbalanced: long in one dimension but short in the other in the planar space. It may also be a non-convex region, e.g., hallway 10 in Figure 6(b). Such irregularities cause much dead space in a tree node, and thus degrade the tree's query performance. To handle them, we decompose [2] an irregular partition into smaller but regular regions. We call the resulting regions, as well as undecomposed regular partitions, *index units*.

For example, in the tree shown in Figure 6(a), the root node is R_0 and the hallway 10 is decomposed into three index units: R_{10}^a, R_{10}^b, and R_{10}^c. The mapping between such an index unit and its original indoor partition is recorded in a hash table *h-table* when the tree is constructed. Formally,

$$h-table: \{\text{index unit}\} \rightarrow \{\text{indoor partition}\}$$

In the tree tier, each leaf node represents an index unit that corresponds to either a regular, undecomposed partition or a smaller region obtained from decomposing an irregular partition. In addition to the MBRs, a leaf node also stores two types of information: 1) a linked bucket for all objects inside it; 2) links to its connected partitions. These two kinds of information belong to the *Object layer* and *Topological layer*, respectively. We proceed to introduce these two layers.

Augmented Tree Tier. The basic idea of performing a spatial join is to use the property that the MBR of an index node covers the MBRs of its subtree. We denote it as the *partial order property*. However, an object might only partially overlap with a partition (or a leaf node). So the MBR of an index node might not cover objects of its subtree. To maintain the partial order property that eases join processing, we augment each tree node t with two attributes, $\{t.r_{max}, t.count\}$. We measure an object's size by the length of its MBR's longest dimension. Further, we use $t.r_{max}$ to represent the largest object size of t's subtree, and $t.count$ to represent the number of objects associated with t's subtree. The update of the tree tier can be handled in an aggregated R-tree manner. Consequently, the *augmented area* of t is the Minkovski sum of t's MBR and its r_{max}, denoted by $t \oplus t.r_{max}$. The augmented area of a tree node covers all those of its subtree. We discuss how to use the property for queries in Section 5.

4.1.3 Object Layer

Due to uncertainty, an object may overlap with multiple indoor partitions. For example, object O_2 overlaps with three partitions in Figure 6(b), namely 10, 11 and 12. In each of the three leaf-nodes' buckets, we store O_2. Meanwhile, we maintain a hash table *o-table* as follows.

$$o-table: \{O\} \rightarrow 2^{\{\text{index unit}\}}$$

Note that o-table maps an object to all the index units it overlaps with, and it is tightly tied up with the tree tier. When an object update occurs, o-table needs to be updated accordingly [3].

4.1.4 Topological Layer

We maintain the connectivity between partitions in this layer. Here, to simplify the discussion, we assume each door always connects two partitions. As introduced in Section 4.1.1, each leaf node stores a (sub)partition. For accessibility, we also store the doors belonging to the partition and the links to accessible partitions through each door. Referring to the running example shown in Figure 6(c), for partition R_{12}, we store door d_{12} together with its accessible partition's link $\uparrow R_{10}^b$.

4.1.5 Skeleton Tier

In our preliminary experiments we found that the Euclidean lower bound is too loose to be effective for indoor space queries. Although it applies to road networks [27] that are modeled as planar graphs, it falls

[2] The detail of decomposition can be found elsewhere [31].

[3] Detail is available elsewhere [31].

short in indoor spaces that are more complex than planar graphs. Usually, an indoor floor's horizontal extent is much larger than its height. Consider a 20-floor building where each floor is of size 600 m × 600 m × 4 m and has four staircases each on one corner. Suppose a range query is issued for the center of the ground floor and asks for objects within 300 meters. Over 90% of the building space is covered if the Euclidean lower bound is used to constrain the search. As a matter of fact, only objects on the ground floor qualify since any path to upper floors is longer than 300 meters due to the staircase positions.

Staircases can be critical in deciding whether to expand the search to other floors or not. This motivates us to design the *Skeleton Tier* that captures all staircases in a concise way to help distance-based pruning in query processing. This tier is a graph. Each staircase entrance is captured as a graph node, and an edge connects two nodes if their entrances are on the same floor *or* their entrances belong to the same staircase. The weight of an edge is the indoor distance between the two staircase entrances. For the staircase plan example in Figure 7(a), its skeleton tier is shown in Figure 7(b).

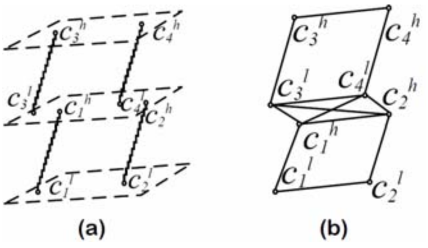


Fig. 7. Skeleton Tier Example

Let M be the total number of staircase entrances in a building, which is much smaller than that of doors in the building. We compute the indoor distance for each pair of staircase entrances and store such distances in an M by M matrix M_{s2s}. Let s_i and s_j be two staircase entrance identifiers. Matrix M_{s2s} satisfies the following properties:

(1) $M_{s2s}[s_i, s_i]$ = 0;
(2) $M_{s2s}[s_i, s_j] = |s_i, s_j|_E$ if s_i and s_j are on the same floor;
(3) if s_i and s_j are of the same staircase, $M_{s2s}[s_i, s_j]$ is the shortest distance from s_i to s_j within that staircase;
(4) $M_{s2s}[s_i, s_j]$ is calculated as the shortest path distance from s_i to s_j in the skeleton layer for other cases.

4.2 Indoor Distance Bounds in the Geometric Layer

Within the geometric layer of the composite index, we can derive tighter indoor distance bounds than the Euclidean distance bounds. Let q be a fixed indoor point, $q.f$ the floor of q, and $S(q.f)$ all the staircases on floor $q.f$. We define the skeleton distance from two points q to p as follows.

Definition 6: **(Skeleton Distance)** Given two points p and q, their skeleton distance $|q, p|_K = |q, p|_E$ if they are on the same floor; otherwise, $|q, p|_K = \underset{s_q \in S(q.f), s_p \in S(p.f)}{min} (|q, s_q|_E + M_{s2s}[s_q, s_p] + |s_p, p|_E)$.

If q and p are on different floors, reaching p from q has to go through one staircase entrance on q's floor and another on p's floor. Therefore, the skeleton distance sums up the Euclidean distance and the indoor distance. Hence, we define the skeleton distance as the alternative *Geometric Distance*. Now we design the *Geometric Lower Bound Property* based on that.

Lemma 8: **(Geometric Lower Bound Property)** Given two points p and q, their skeleton distance lower bounds their indoor distance, i.e., $|q, p|_K \leq |q, p|_I$.

Proof: If q and p are on the same floor, $|q, p|_K = |q, p|_E \leq |q, p|_I$. Otherwise, suppose $s_q^* \in S(q.f)$ and $s_p^* \in S(p.f)$ are on the shortest path from q to p, denoted by $q \overset{*s_q^* * s_p^* *}{\rightarrow} p$. Since $|q, p|_K = \underset{s_q \in S(q.f), s_p \in S(p.f)}{min} (|q, s_q|_E + M_{s2s}[s_q, s_p] + |s_p, p|_E) \leq |q, s_q^*|_E + M_{s2s}[s_q^*, s_p^*] + |s_p^*, p|_E = |q, p|_I$, the lemma is proved. □

Consider an entity e that is either an object or an *ind*R-tree node. If e spans multiple floors, we use the interval $[e.lf, e.uf]$ to represent all those floors. Note those floors must be consecutive. We define the mininum skeleton distance $|q, e|_{minK}$:

$$|q, e|_{minK} = \begin{cases} |q, e|_{minE}, \text{ if } q.f \in [e.lf, e.uf]; \\ min \\ \{ \underset{s_q \in S(q.f), s_e \in S(e.lf)}{min} (|q, s_q|_E + M_{s2s}[s_q, s_e] + |s_e, e|_{minE}), \\ \quad \underset{s_q \in S(q.f), s_e \in S(e.uf)}{min} (|q, s_q|_E + M_{s2s}[s_q, s_e] + |s_e, e|_{minE})\}, \\ otherwise. \end{cases} \tag{15}$$

With $|q, e|_{minK}$ [4], we can constrain the search via the *ind*R-tree to a much smaller range compared to if we use the Euclidean distance bounds. We design an algorithm called `RangeSearch`, as shown in Algorithm 4 in the appendix.The algorithm takes a query point q and a distance r as input, and returns the objects and partitions within the specified range. When r=0, the query degenerates to a point-location query that returns the partition containing q.

5 Efficient Distance-Aware Join Evaluation

We make use of the indoor distances (Section 2 and 3) and the index (Section 4) to efficiently evaluate the semi-range and semi-neighborhood joins. Our query evaluation consists of 4 phases. The first phase, *filtering*, retrieves candidate partitions as well as candidate objects for join pairs. The second phase, *subgraph*, constructs a subgraph based on candidate partitions, and uses the doors of the partition containing query objects as sources

[4] Note that if e is a descendant of E ($e \subseteq E$), we have $|q, E|_{minK} \leq |q, e|_{minK}$, because one has to go through some parts of E to reach e.

to compute the shortest indoor paths that are to be used in the subsequent two phases. In the third phase, *pruning*, upper/lower distance bounds for objects are calculated to further reduce the number of candidate join pairs. In the fourth phase, *refinement*, the indoor distances for the remaining pairs are computed and the qualifying results are returned as the query results.

Batch Processing. We can combine the processing of the filtering and subgraph phases for query objects belonging to the same partition Q^P. The intuition is to treat partition Q^P as a "big" object, whose augmented area $Q^P \oplus Q^P{\cdot}_{r_{max}}$ covers the regions of all the objects inside. For the filtering phase, if a target object O has $|Q^P \oplus Q^P{\cdot}_{r_{max}}, O|_{minK} > \epsilon$, then O can be filtered out. Because object Q must have a longer distance to O, we have that $|Q, O|_{minK} > \epsilon$, according to the partial order property. For the subgraph phase, objects have the same source partition for the shortest path calculation. Also, their surrounding partitions, whose distance from the source partition is indicated by a parameter, ϵ, should also be the similar. Thus, their subgraph phases can be combined. The correctness is guaranteed by Lemma 9.

Lemma 9: Suppose object Q is associated with partition P^Q. For another partition P, if $|P^Q \oplus P^Q{\cdot}_{r_{max}}, P|_{minI} \geq \epsilon$, then $|Q, P|_{minI} \geq \epsilon$.

Proof: Since region $R = P^Q \oplus P^Q{\cdot}_{r_{max}}$ must cover Q, we have $|R, P|_{minI} \leq |Q, P|_{minI}$. Thus, if $|R, P|_{minI} \geq \epsilon$, $|Q, P|_{minI}$ is definitely no shorter than ϵ. The lemma is proved. □

In the subsequent sections, we consider the two sets of objects, $\mathbb{Q}$ and $\mathbb{O}$, organized by a single index. Each index node is associated with two augmented attributes, for $\mathbb{Q}$ and $\mathbb{O}$, respectively. We use two pointers, *QNode* and *ONode*, to point to the index nodes for each object set. Then, although tree nodes are indexed by a single physical structure, they appear as two logical indexes. For example, we have two logical root nodes, QNode $\mathcal{T}_\mathcal{Q}$ and ONode $\mathcal{T}_\mathcal{O}$, which point to the same physical root node. We proceed to present the algorithms for semi-range join in Section 5.1, and semi-neighborhood join in Section 5.2.

5.1 Semi-range Join

The evaluation of the semi-range join is formalized in Algorithm 1. In the filtering step, semi-range join first calls a recursive procedure, `Semi-range-filtering`, to quickly qualify or disqualify the joining pairs. The procedure is run in the geometric layer. In particular, those tree node pairs with geometric distances within ϵ are retrieved and the corresponding object pairs are qualified as query answers. Undetermined tree node pairs are kept in the candidate set C. Then, the algorithm calls `RangeSearch` (Algorithm 4 in the appendix) to search target objects for each element in C. Given the *Geometric Lower Bound Property* (Lemma 8), R^o and R^P are guaranteed to avoid false negatives. Specifically, any discarded entity e (object or partition) satisfies $|Q, e|_I \geq |Q, e|_{minK} > \epsilon$ for any query object Q in partition Q^P.

In the subgraph phase, the *Dijkstra* Algorithm is called to calculate single-source shortest paths starting at the doors of the partition Q^P[5]. The subgraph phase for the objects in Q^P is conducted in a batch. The correctness is guaranteed by Lemma 9. The object pairs from Q^P and R^P contain false positives. So the algorithm continues to subsequent phases to verify the candidate join pairs incrementally. Specifically, the algorithms makes use of the topological and object upper/lower bounds to approximate indoor distances and compare them to ϵ (`Lines 9-12`). The exact indoor distances are only computed for those object pairs whose distance bounds cover ϵ (`Lines 13-15`).

Algorithm 1 Semi-range Join

```
 1: function SEMI-RANGE-JOIN(indoor index $\mathcal{T}_\mathcal{Q}$, indoor index $\mathcal{T}_\mathcal{O}$, distance $\epsilon$)
 2:     result set $R$; candidate partition set $C$;          ▷ Golobal variables
 3:     Semi-range-Filtering($\mathcal{T}_\mathcal{Q}$ , $T_O$, $\epsilon$);      ▷ Phase 1: filtering
 4:     for each partition $Q^P \in C$ do
 5:         Cand ← ∅   ▷ Cand is a set for candidate join pairs
 6:         $\langle R^O, R^P \rangle \leftarrow$ RangeSearch($Q^P, \epsilon + Q^P{\cdot}_{r_{max}}, T_O$);
 7:       ▷ $R^O$ is a set of objects in $\mathbb{O}$; $R^P$ is a set of partitions;
 8:         Dijkstra($R^P$);                        ▷ Phase 2: subgraph
 9:         for each object pair $\langle Q, O \rangle_{Q \in Q^P, O \in R^O}$ do
10:             $[\langle Q, O \rangle.l, \langle Q, O \rangle.u] \leftarrow [|Q, O|_{minI}, |Q, O|_{maxI}]$;(Table 2)
11:                                                  ▷ Phase 3: pruning
12:             if $\langle Q, O \rangle.u \leq \epsilon$ then $R = R \cup \langle Q, O \rangle$
13:             else
14:                 if $\langle Q, O \rangle.l \leq \epsilon$ then $Cand = Cand \cup \langle Q, O \rangle$
15:             for each $\langle Q, O \rangle \in Cand$ do      ▷ Phase 4:refinement
16:                 Calculate $|Q, O|_I$;
17:                 if $|Q, O|_I \leq \epsilon$ then $R = R \cup \langle Q, O \rangle$;
18:     return $R$.
 1: procedure SEMI-RANGE-FILTERING(QNode $\mathcal{Q}$, ONode $\mathcal{O}$, distance $\epsilon$) ▷ QNode is a node of $T_Q$; ONode is a node of $T_O$
 2:     if $\mathcal{Q}{\cdot}_{count} = 0$ or $\mathcal{O}{\cdot}_{count} = 0$ then
 3:         return;
 4:     if $|\mathcal{Q}, \mathcal{O}|_{minK} - \mathcal{Q}{\cdot}_{r_{max}} - \mathcal{O}{\cdot}_{r_{max}} \geq \epsilon$ then
 5:         Add $\mathcal{Q}$ to $C$;
 6:     else
 7:         if $|\mathcal{Q}, \mathcal{O}|_{maxK} + \mathcal{Q}{\cdot}_{r_{max}} + \mathcal{O}{\cdot}_{r_{max}} \leq \epsilon$ then
 8:             $R = R \cup \{\langle Q, O \rangle | Q \in \mathcal{Q}, O \in \mathcal{O}\}$;
 9:         else
10:             if $\mathcal{Q}$ is a leaf node then
11:                 Add $\mathcal{Q}$ to $C$;
12:             else
13:                 for each child node $Q^P \in \mathcal{Q}$ do
14:                     for each child node $O^P \in \mathcal{O}$ do
15:                         Semi-range-Filtering($Q^P$, $O^P$, $\epsilon$);
```

5.2 Semi-neighborhood Join

The evaluation of semi-neighborhood join is formalized in Algorithm 2. In the filtering phase, a semi-neighborhood join first traverses the index to retrieve the leaf nodes containing one or more query objects, and stores them in C. Then, for each partition Q^P in

[5] The weight of an edge is the Euclidean distance of two accessible doors as aforementioned in Section 2.

set C, the algorithm calls *kSeedsSelection* (Algorithm 3 in the appendix) to return an object set R_1^o and a partition set R_1^p. Specifically, R_1^o contains the k objects that are in partition Q^P or in the closest adjacent partitions, and R_1^p is the set of all the involved partitions. Then, the algorithm derives *Topological Looser Upper Bounds* for the k objects and chooses the longest one as *k-bound* = $\underset{seed_i \in R_1^o}{max}\{|Q^P, seed_i|_I.TLU\}$. Next, a range search $\odot(Q^P, kbound)$ is done on the tree tier (`Line 9`). The *Geometric Lower Bound Property* (Lemma 8) ensures zero false negatives.

The algorithm continues to apply the *Dijkstra* Algorithm and derives upper/lower bounds (`Lines 10`). The remaining target objects are sorted and O_k whose upper bound is the k-th shortest is found. Target objects with $O.u$ closer than $O_k.l$ are added to R_O as qualified pair halves for Q (`Line 18`). Target objects with $O.l$ farther than $O_k.u$ have no chances as there are already k objects closer. For undetermined objects, their indoor distances are calculated and the qualifying ones are picked (`Lines 21-22`). Then, the objects are sorted and the k objects with shortest distances are picked to form join pairs with Q. The above process is repeated for all partitions in C before the semi-neighborhood join result is finalized and returned.

Algorithm 2 Semi-neighborhood join

```
 1: function SEMI-NEIGHBORHOOD-JOIN(indoor index $\mathcal{T}_\mathcal{Q}$, indoor index $\mathcal{T}_\mathcal{O}$, parameter $k$)
 2:     result set $R$; candidate object set $C$; ▷ Global variables
 3:     for each of $T_Q$'s leaf nodes $\mathcal{Q}$ do    ▷ Phase 1: filtering
 4:         if $\mathcal{Q}._{count}$==0 then
 5:             Add $\mathcal{Q}$ to $C$;
 6:     for each partition $Q^P$ in $C$ do
 7:         $(R_1^o, R_1^p)$ ←kSeedsSelection($Q^P, k$);
 8:         kbound ← $max_{O \in R_1^o}\{|Q^P, O|_I.TLU\}$;    ▷ Lemma 5
 9:         $(R_2^o, R_2^p)$ ←RangeSearch($Q^P$,kbound +$Q^P._{r_{max}}$,$T_O$);
10:         Dijkstra($R_2^p$);    ▷ Phase 2: subgraph
11:         for each object $Q$ in partition $Q^P$ do
12:             $C_O \leftarrow \emptyset$ ▷ $C_O$ is a set for candidate objects in $\mathbb{O}$
13:             $R_O \leftarrow \emptyset$    ▷ $R_O = kNN(Q)$
14:             for each object $O$ in $R_2^o$ do    ▷ Phase 3: pruning
15:                 $[O.l, O.u] \leftarrow [|Q,O|_{minI}, |Q,O|_{maxI}]$;▷ Table2
16:             Find object $O_k$ which has the k-th shortest $O.u$;
17:             for each $O \in R_2^o$ do
18:                 if $O.u < O_k.l$ then $R_O = R_O \cup \{O\}$
19:                 else
20:                     if $O.l \leq O_k.u$ then $C_O = C_O \cup \{O\}$
21:             for each $O \in C_O$ do    ▷ Phase 4: refinement
22:                 Calculate $|Q,O|_I$;
23:             Sort objects in $C_O$ by $|Q,O|_I$ in ascending order and add top $k - |R_O|$ objects to $R_O$;
24:             $R \leftarrow R \cup \{Q,O\}_{O \in R_O}$
25:     return $R$;
```

6 Experimental Studies

We conduct extensive experimental studies to evaluate our proposals. Section 6.1 describes the experimental settings, where default parameters are bolded. Section 6.2 reports the experimental results.

6.1 Experimental Setup

Indoor Space. We use a real floor plan of a shopping mall[6]. Each floor takes 600 m ×600 m × 4 m, with 100 rooms and 4 staircases. To test scalability, we use the plan to generate buildings with 10, **20**, and 30 floors. All of them are connected by hallways and staircases. We vary the number of floors and therefore also the number of partitions and doors.

Indoor Moving Objects. We generate a series of datasets, containing 10K, **20K**, and 30K objects for $\mathbb{O}$ and 100, **200**, 400 objects for $\mathbb{Q}$. By default, they are randomly distributed in a given building. For other types of distributions, such as Gaussian and Zipfian, we use Theodoridis et al's data generator [7]. For Gaussian distribution, we set the mean as the center of the floor and the variance as the square of 1/6 of its side length. We first obtain an object set of a plane, by setting the domain to 600 × 600 (the size of a floor). Then, we copy the set to all floors of the building. Objects' uncertainty regions are represented by circles, with the diameter 10, **30**, and 50 meters. The *pdf* is represented by a set of 100 sampling points, following a Gaussian distribution. The mean is the circle center and the variance is the square of 1/6 of its diameter.

Tree Tier. We use a packed R*-tree [24] to index all indoor partitions. The entire tree is accommodated in the main memory. We set the tree fanout to be 20, according to the results reported elsewhere [12].

Query Parameters. For semi-range join, we set the query range to 50, **100**, and 150 meters. For semi-neighborhood join, we set k to 10, **30**, and 50. In all experiments, we issue 10 queries and report the average response time for each query type.

We implement all programs in C++ and conduct experiments on a PC running MS Windows 7 Enterprise with Core2 Duo 3.40GHz CPU and 8GB main memory.

6.2 Experimental Results

Sections 6.2.1 and 6.2.2 report the query performances for semi-range join and semi-neighborhood join, respectively. For both query types, we test their efficiency and scalability with respect to the number of objects ($|\mathbb{O}|$ and $|\mathbb{Q}|$), the size of uncertainty regions, and the number of partitions. Section 6.2.3 investigates the effectiveness of our indoor distance bounds in the filtering and pruning phases. [8]

6.2.1 Performance of Semi-range Join

The results of semi-range join's execution time are reported in Figure 8. Referring to Figure 8(a), the query

[6] http://fc06.deviantart.net/fs28/f/2008/143/4/6/Floor_Plan_for _a_Shopping_Mall _by_mjponso.png

[7] http://www.rtreeportal.org/software/SpatialDataGenerator.zip

[8] Due to page limits, please refer to our previous work [31] for the performances of the composite indoor index.

time increases sublinearly with $|\mathbb{O}|$ and the area of query ranges. We show the query time break-down for default settings in Figure 8(b). The filtering and subgraph phases depend on the topologies, and thus they do not change as $|\mathbb{O}|$ increases. On the other hand, larger $|\mathbb{O}|$s make the refinement phase handle more objects that pass the filtering and pruning phases, and thus increase the query time. As the objects' uncertainty regions become larger, more objects are involved in the semi-range join execution, and therefore the query time also increases, as shown in Figure 8(c). We also fix the number of objects and vary the number of partitions to see the effect on query time. The results are shown in Figure 8(d). Since the average number of objects in one partition (i.e., object density in each partition) decreases, we see query time decreases accordingly. In Figure 8(e), we vary the number of query objects $|\mathbb{Q}|$, the query time increases sub-linearly. Intuitively, the query time meant to increase linearly w.r.t. ($|\mathbb{Q}|$). Since the batch processing reuses the calculation of filtering and subgraph phases, the performance is further improved. We examine the performance of semi-range join on different types of distributions, in Figure 8(f). The query time increases with $|\mathbb{O}|$. The skewed distribution requires more refinement efforts thus corresponds to a relatively higher execution time.

6.2.2 Performance of Semi-neighborhood Join

The results of semi-neighborhood join's execution time are reported in Figure 9. Referring to Figure 9(a), the query time increases stably as the number of objects and k increase. The query time break-down for default settings is shown in Figure 9(b). Compared to semi-range join, semi-neighborhood join needs to retrieve more indoor partitions to find sufficient candidates in the filtering phase. Consequently, the subsequent phases get higher workloads to process. The results on the effect of object uncertainty region size are shown in Figure 9(c). Larger uncertainty sizes render more objects and partitions to be retrieved in the range search step, and thus increase the query execution time. The query execution time on the effect of the number of partitions are shown in Figure 9(d). Again, query time decreases as the object density in each partition decreases. We demonstrate the performance w.r.t. $|\mathbb{Q}|$ in Figure 9(e). The query time increases sub-linearly, which is as expected because the query objects belonging to the same partition are batch processed. We compare the query performance over different distributions of $\mathbb{O}$. The Uniform distribution performs worst. The reason is that for semi-neighborhood join query results rely on the distance rankings. If objects are more equally distant from queries, the ambiguity on rankings is larger and more refinement effort is required.

6.2.3 Effectiveness of Indoor Distance Bounds

Our indoor distance bounds contribute to the efficiency of query execution through filtering and pruning phases, as indicated by the results shown in Figure 10. We define the term *pruning ratio* as the ratio of join pairs disqualified over $|\mathbb{O}| \times |\mathbb{Q}|$.

Referring to Figure 10(a), over 98% join pairs are filtered out by the skeleton distance bound in the filtering phase of semi-range join in all tested settings. The results show that the skeleton layer and the skeleton distance bound are very effective in filtering indoor partitions and objects at a high level without the search going down to the object layer. Without the filtering phase, all indoor partitions would be involved in the shortest path computation, which would be too expensive for the query execution. After the pruning phase over 99.9% join pairs in total are pruned. Specially, when $|\mathbb{O}|$=20k, the pruning ratio is as high as 99.96%. We further study the effect of the pruning phase by including and excluding it in semi-range join. The results are shown in Figure 10(b). Clearly, the topological distance bounds (Table 2) used in the pruning phase are very effective in speeding up the query processing.

The counterpart results for semi-neighborhood join are shown in Figures 10(c) and 10(d). Again, indoor distance bounds are very effective in discarding unqualified pairs. Referring to Figure 10(d), the query time would increase by at least 4 times without the pruning phase.

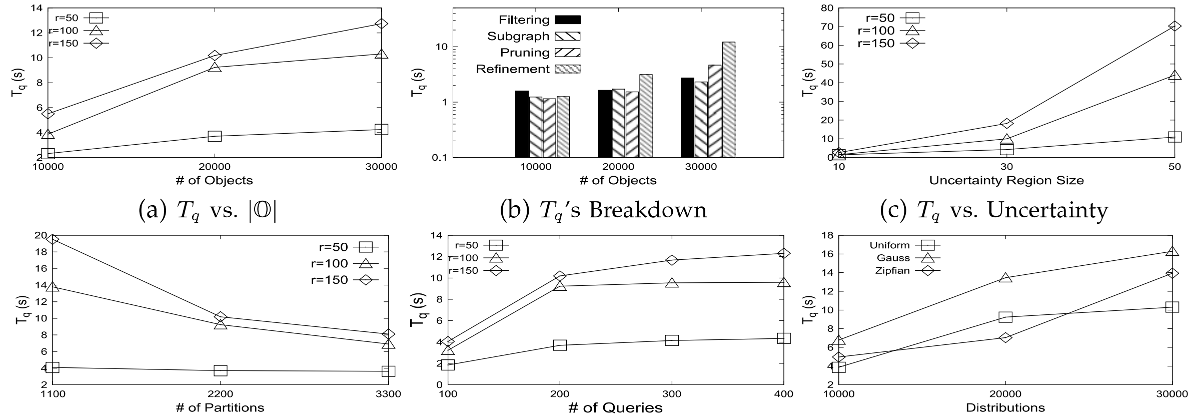
7 Related Work

Different indoor space models have been proposed. The 3D Geometric Network Model [17] treats the vertical and horizontal connectivity relationship among 3D spatial cells separately. The 3D Indoor Geo-Coding technique employs the 3D *Poincaré Duality* [25] to transform 3D spatial cells from primal space to dual space. A 3D metrical-topological model [30] describes both the shapes and connectivity of spatial cells for navigation purposes. Another 3D model [2] combines space partitions with possible events in a dual space, to enable navigation in multi-layered buildings. Focusing on topological relationships, these models do not support indoor distances and relevant queries.

A lattice-based semantic location model [18] defines the "length" of an indoor path by the number of doors on the path rather than the actual indoor distance. As a result, this model falls short in many practical scenarios [22]. Different ways of transforming a floor plan into a graph also exist [11], [13], [29], but such proposals lack support for indoor distances.

Research on indoor moving objects often assumes symbolic indoor space modeling and indoor positioning [13]. R-tree based structures [14] have been used to index offline trajectories of moving objects in symbolic indoor spaces. By differentiating object states in terms of positioning detection, a hash indexing method [32], [33] has been designed to index the online positions of indoor moving objects.

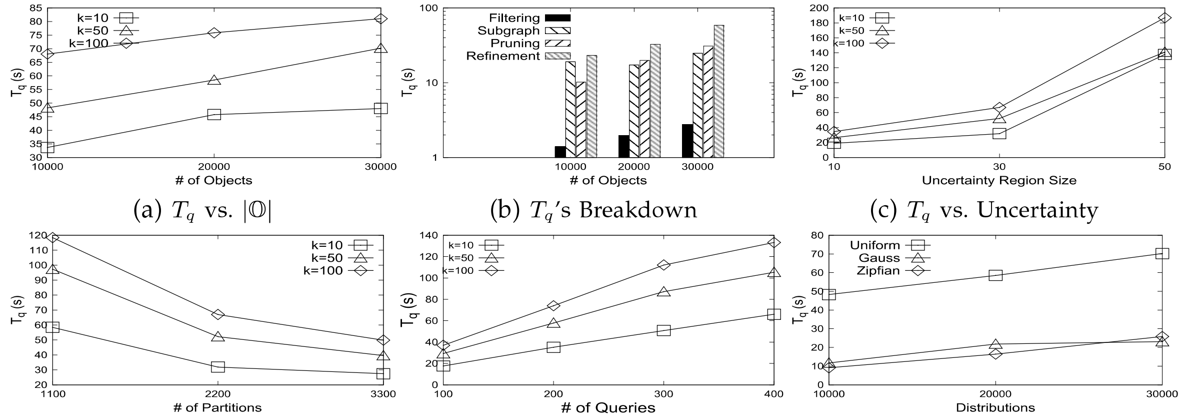
Previous works [22], [23], [33] study spatial queries on online indoor moving objects. This paper differs from these works in several aspects. First, previous works [22],



(a) T_q vs. $|\mathbb{O}|$ (b) T_q's Breakdown (c) T_q vs. Uncertainty

(d) T_q vs. # of Partitions (e) T_q vs. $|\mathbb{Q}|$ (f) T_q vs. Distributions

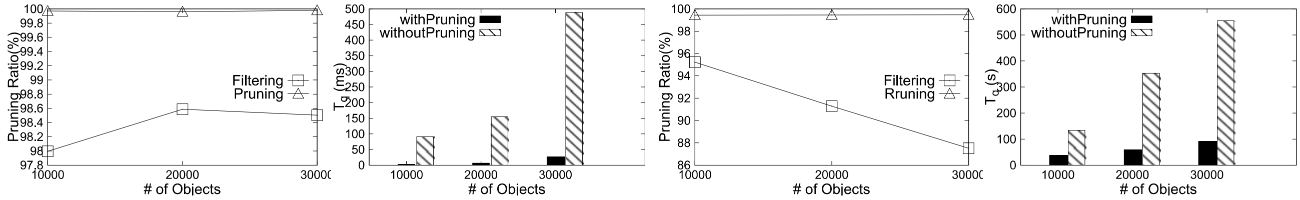
Fig. 8. Semi-range Join (T_q)



(a) T_q vs. $|\mathbb{O}|$ (b) T_q's Breakdown (c) T_q vs. Uncertainty

(d) T_q vs. # of Partitions (e) T_q vs. $|\mathbb{Q}|$ (f) T_q vs. Distributions

Fig. 9. Semi-neighborhood Join (T_q)



(a) Filtering & Pruning (Semi-range Join) (b) Pruning Effectiveness (Semi-range Join) (c) Filtering & Pruning (Semi-neighborhood Join) (d) Pruning Effectiveness (Semi-neighborhood Join)

Fig. 10. Effectiveness of Indoor Distance Bounds

[23], [33] assume that all door-to-door distances are precomputed for query processing, whereas this paper lifts this assumption and computes indoor distances on the fly in query processing. Second, the previous work [22] queries on indoor static objects (points of interest, i.e., POIs) while the queries in this paper are on indoor moving objects. Third, indoor join semantics in [23] are based on co-location in an indoor unit, whereas this paper defines distance-aware join semantics.

8 Conclusion

In this paper, we study efficient evaluation of distance-aware join operations on indoor moving objects. We study two representative join operators, semi-range join and semi-neighborhood join. We investigate the indoor distance categories regarding object location uncertainties and indoor topologies. To speed up distance based pruning in query evaluation, we propose effective indoor distance upper/lower bounds. We also design a composite index for indoor space as well as objects, which facilitates efficient indoor distance retrieval as well as query processing. Extensive experimental results demonstrate that our proposals are effective, efficient and scalable in various query settings.

Our work in this paper opens directions for future work. First, it is of interest to study other query types using the distance bounds and the composite index proposed in this paper. Second, it is useful to estimate the selectivity for indoor distance aware queries and make use of it in further optimizing queries over uncertain objects. Third, it is beneficial to reuse computational efforts on indoor distances when multiple, related queries are issued within a short period of time.

References

[1] Accurate indoor positioning and navigation at the threshold. http://www.loctronix.com/news/insider/i1-1-a3-scpdain.html (accessed July 2012).

[2] T. Becker, C. Nagel, and T. H. Becker. A multilayered space-event model for navigation in indoor spaces. *Proc. 3rd International Workshop on 3D Geo-Info*, 2008.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD*, 1990.

[4] C. Böhm and F. Krebs. The k-nearest neighbour join: Turbo charging the kdd process. *Knowl. Inf. Syst.*, pages 728–749, 2004.

[5] b. . C. y. . . Böhm, Christian and Braunmüller, Bernhard and Breunig, Markus and Kriegel, Hans-Peter, title = High performance clustering based on the similarity join.

[6] J. Chen and R. Cheng. Efficient evaluation of imprecise location-dependent queries. In *ICDE*, 2007.

[7] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, 2003.

[8] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *TKDE*, 16(9), 2004.

[9] G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, pages 305–316, 2009.

[10] D.Pfoser and C. S.Jensen. Capturing the uncertainty of moving-objects representations. In *SSDBM*, 1999.

[11] G. Franz, H. Mallot, J. Wiener, and K. Neurowissenschaft. Graph-based Models of Space in Architecture and Cognitive Science-a Comparative Analysis. In *IIAS InterSymp*, pages 30–38, 2005.

[12] S. Hwang, K. Kwon, S. Cha, and B. Lee. Performance evaluation of main-memory r-tree variants. In *SSTD*. 2003.

[13] C. S. Jensen, H. Lu, and B. Yang. Graph model based indoor tracking. In *MDM*, 2009.

[14] C. S. Jensen, H. Lu, and B. Yang. Indexing the trajectories of moving objects in symbolic indoor space. In *SSTD*, pages 208–227, 2009.

[15] L. Klingbeil, R. Reiner, M. Romanovas, M. Traechtler, and Y. Manoli. Multi-modal sensor data and information fusion for localization in indoor environments. In *WPNC*, pages 187–192, 2010.

[16] H.-P. Kriegel, P. Kunath, and M. Renz. Probabilistic nearest-neighbor query on uncertain objects. *DASFAA*, 2007.

[17] J. Lee. A spatial access-oriented implementation of a 3-d gis topological data model for urban entities. *GeoInformatica*, 8(3):237–264, 2004.

[18] D. Li and D. L. Lee. A lattice-based semantic location model for indoor navigation. In *MDM*, pages 17–24, 2008.

[19] X. Lian and L. Chen. Monochromatic and bichromatic reverse skyline search over uncertain databases. In *SIGMOD*, 2008.

[20] X. Lian and L. Chen. Probabilistic group nearest neighbor queries in uncertain databases. In *TKDE*. 2008.

[21] X. Lian and L. Chen. Similarity join processing on uncertain data streams. In *TKDE*. 2011.

[22] H. Lu, X. Cao, and C. S. Jensen. A foundation for efficient indoor distance-aware query processing. In *ICDE*, 2012.

[23] H. Lu, B. Yang, and C. S. Jensen. Spatio-temporal joins on symbolic indoor tracking data. In *ICDE*, 2011.

[24] M.Hadjieleftheriou. Spatial index library version 0.44.2b.

[25] J. Munkres. *Elements of algebraic topology*. Addison Wesley Publishing Company, 1993.

[26] P. Sistla et al. Querying the uncertain position of moving objects. In *Temporal Databases: Research and Practice*. 1998.

[27] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *VLDB*, 2003.

[28] A. R., L. K., S. H., and S. K. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *VLDB*, 1995.

[29] C. van Treeck and E. Rank. Analysis of building structure and topology based on graph theory. In *ICCCBE*, 2004.

[30] E. Whiting, J. Battat, and S. Teller. Topology of urban environments. In *CAAD Futures*, pages 115–128, 2007.

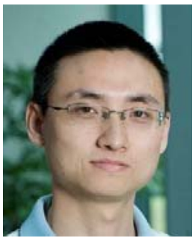
[31] X. Xie, H. Lu, and T. B. Pedersen. Efficient distance-aware query evaluation on indoor moving objects. In *ICDE*, 2013.

[32] B. Yang, H. Lu, and C. S. Jensen. Scalable continuous range monitoring of moving objects in symbolic indoor space. In *CIKM*, 2009.

[33] B. Yang, H. Lu, and C. S. Jensen. Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space. In *EDBT*, 2010.

[34] J. Zhang, D. Papadias, K. Mouratidis, and M. Zhu. Spatial queries in the presence of obstacles. In *EDBT*, pages 366–384, 2004.

Xike Xie received the BSc and MSc degrees from Xi'an Jiaotong University, China, in 2003 and 2006, respectively, and the PhD degree in computer science from the University of Hong Kong in 2012. He is currently an assistant professor in the Department of Computer Science, Aalborg University, Denmark. His research interests include data uncertainty, spatiotemporal databases, and mobile computing. He is a member of the IEEE and ACM.



Hua Lu received the BSc and MSc degrees from Peking University, China, in 1998 and 2001, respectively, and the PhD degree in computer science from National University of Singapore, in 2007. He is an associate professor in the Department of Computer Science, Aalborg University, Denmark. His research interests include databases, geographic information systems, as well as mobile computing. Recently, he has been working on indoor spatial awareness, complex queries on spatial data with heterogeneous attributes, and social media data management. He has served on the program committees for conferences and workshops including ICDE, SSTD, MDM, PAKDD, APWeb, and MobiDE. He is PC cochair or vice chair for ISA 2011, MUE 2011 and MDM 2012. He is a member of the IEEE.

Torben Bach Pedersen is a full professor of computer science at Aalborg University (AAU), Denmark, where he heads the Center for Data-Intensive Systems. He received his Ph.D. in computer science from AAU in 2000 and his M.Sc. in computer science and mathematics from Aarhus University in 1994. Before joining AAU in 2000, he worked as a business intelligence specialist in industry for six years. His research interests span business intelligence topics such as data warehousing, OLAP, and data mining, with a focus on non-traditional and complex types of data such as web-related data, spatial and spatio-temporal data, music data, energy data, and medical data. He has published more than 110 peer-reviewed papers on these topics. He has served as PC Chair for DaWaK 2009 and 2010 and DOLAP 2010, General Chair for SSTD 2009, Co-Chair of EnDM 2012 and Cloud-I 2012, and on numerous program committees, including VLDB, ICDE, and EDBT. He is a member of the SSTD Endowment and the DOLAP Steering Committee.